Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Propuesta de mejora en el departamento de servicio al cliente en una empresa desarrolladora de software

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Pablo Sánchez Iniesta

Tutor: Andrés Boza García

Curso Académico: 2016-2017

A mi tutor por su ayuda y tiempo dedicado,

a mi familia por su apoyo incondicional,

a la empresa en la que se ha realizado el

proyecto por todo su soporte y recursos.

Gracias.

Resumen

El objetivo principal de este trabajo es estudiar las necesidades, objetivos y responsabilidades de los diferentes roles del departamento de soporte de una gran multinacional del software y proponer las posibles soluciones de mejora. Se pueden identificar cuatro partes significativas en este trabajo. Una primera sección en la que se realiza un estudio de los modelos de proceso, de las metodologías tradicionales así como de las metodologías ágiles. Una segunda parte en la que se estudia la metodología de solución de problemas Design Thinking así como sus diferentes fases y métodos. En tercer lugar, se presenta la aplicación de la metodología de Design Thinking llevada a cabo en un proyecto real, delimitando y estudiando los problemas del departamento. Finalmente, se explican las propuestas de mejoras en forma de prototipos (Wireframes) de los problemas identificados en el Espacio del Problema.

La propuesta de mejora se centra principalmente en identificar los requerimientos establecidos en el Espacio de la Solución y de la posible solución de diseño mediante la creación de prototipos de las diferentes ventanas que forman el Documento de Investigación.

Palabras clave: Design Thinking, prototipos, metodologías agiles, metodologías tradicionales, Wireframe, modelos de proceso.

Resum

L'objectiu principal d'aquest treball és estudiar les necessitats, objectius i responsabilitats dels diferents rols del departament de suport d'una multinacional del software i proposar solucions de millora. Es poden identificar quatre parts significatives en aquest treball. Una primera secció en la qual es realitza un estudi dels models de procés, de les metodologies tradicionals i també de les metodologies àgils. Una segona part en la qual s'estudia la metodologia de solució de problemes Design Thinking, així com les seues diferents fases i mètodes. En tercer lloc es presenta l'aplicació de la metodologia de Design Thinking, duta a terme en un projecte real, delimitant i estudiant els problemes del departament. Finalment, s'expliquen les propostes de millora en forma de prototips (Wireframes) dels problemes identificats a l'espai del problema.

La proposta de millora es centra principalment a identificar els requeriments identificats a l'espai de la solució i de la possible solució de disseny mitjançant la creació de prototips dels diferents marcs que formen el document d'investigació.=

Paraules clau: Design Thinking, prototips, metodologies àgils, metodologies tradicionals, Wireframe, models de procés.

The principal objective of this academic work is to study the needs, objectives and responsibilities of the different roles in a global department of support in a large multinational company of the business software and to suggest some solutions of improvement. We can distinguish within four parts in this academic work. The first section is about a study of the process modelling, the traditional methodologies and the agile methodologies. In the second part of the document it's studied the Design Thinking Methodology studying the different methods and phases. In third place it's about carrying out the Design Thinking Methodology in a real project, delimiting and studying the problems of the department. Finally, it's explained the different proposal of improvement in form of prototypes (Wireframes) of the different problems identifying in the Problem Space.

The proposal of improvement is mainly centred in identifying the requirements established in the Solution Space and the possible design solution by the creation of prototypes of the different windows that form the Investigation Document.

Keywords: Design Thinking, prototypes, agile methodologies, traditional methodologies, Wireframe, process modelling.

Tabla de contenidos

Índice de tablas y figuras 8
1. Introducción 9
2. Contexto del problema 11
3. Base Teórica 13
3.1 Historia de la GUI 13
3.1.1 1980 13
3.1.2 1990 14
3.2 Software 15
3.3 Modelos de Proceso 18
3.3.1 Metodologías Tradicionales 18
3.3.2 Metodologías Ágiles 22
3.4 Ingeniería de Requerimientos 26
3.4.1 Requerimientos funcionales 27
3.4.2 Requerimientos no funcionales 28
3.5 Design Thinking 29
3.5.1 Fases 31
3.5.2 Métodos 35
3.6 Principios Universales del Diseño 40
3.7 Interacción Persona Computador 42
3.8 Human Interface Guidelines 46
3.8.1 Google Material Design Guideline 46
3.8.2 Apple Human Interface Guideline 47
3.9 Diseño de la interfaz del usuario 49
3.9.1 Balsamiq 49
4. Propuesta de mejora 50
4.1 Contexto del Problema 50
4.2 Primera toma de contacto 53
4.2.2 Herramientas Existentes 54
4.3 Design Thinking 55
4.3.1 Espacio del Problema 55
4.3.2 Espacio de la Solución 62
4.3.3 Espacio de la Implementación 69

5. Conclusión 71
6. Bibliografía 73
Bibliografía utilizada: 73
7. Anexos 75
7.1 Anexo 1 - Storyboard Amy 75
7.2 Anexo 2 - Cartas de Presentación 77
7.3 Anexo 3 - Requerimientos 81
7.4 Anexo 4 - Wireframes 85

Índice de tablas y figuras

Figura 3.1 Tasa de fallos del software a lo largo del tiempo. 15
Figura 3.2 Ciclo de vida del modelo en espiral 21
Figura 3.3 Fases del Design Thinking 29
Figura 3.4 Agrupación de fases del Design Thinking en Espacios 30
Figura 3.5 Combinación de las intenciones del Design Thinking 30
Figura 3.6 Comparativa de diseño entre Radio de Bolsillo Braun (1959) y el iPod Classic 1st Generation (2001) 41
Figura 3.7 Distribución en capas de Material Design 46
Figura 3.8 Distribución y elevación entre los diferentes componentes de Material Design 47
Figura 3.9 Interfaz de Balsamiq junto con Mockup de la pantalla de inicio de la UPV 49

Tabla 3. 1 Tabla comparativa Metodologías Tradicionales contra Metodologías Agiles 23

Figura 4.1 Carta de presentación de la Persona Amy, la Asistente de Soporte y Manager del Éxito 57
Figura 4.2 Knowledge Workspace ejemplo 63
Figura 4.3 Ejemplo primer Wireframe basado en Knowledge Workspace 65
Figura 4.4 Wireframe basado en Fiori Guideline 66
Figura 4.5 Wireframe basado en estilo web tradicional con barra superior 67
Figura 4.6 Wireframe basado en pestañas con barra superior 67
Figura 4.7 Conexión entre los diferentes Wireframes 68

Figura 7.1 Carta de presentación de Tara la Líder de Equipo 78
Figura 7.2 Carta de Presentación Marc el Manager 79
Figura 7.3 Carta de Presentación de Frank, el Líder Funcional 80
Figura 7.4 Requerimientos de la Persona Amy 1-17 81
Figura 7.5 Requerimientos de la Persona Amy 18-34 82
Figura 7.6 Requerimientos de la Persona Amy 35-52 83
Figura 7.7 Requerimientos de la Persona Amy 53-65 84
Figura 7.8 Wireframe Material Design, Dificultad de la Relación 85
Figura 7.9 Wireframe Material Design, Stakeholders detalle Director de IT 85
Figura 7. 10 Wireframe Material Design, Stakeholders 86
Figura 7.11 Wireframe Material Design, Indicadores de la relación 86
Figura 7.12 Wireframe Material Design, Indicadores de la Relación 87
Figura 7.13 Wireframe Material Design, Dificultad de la Relación 87
Figura 7.14 Wireframe Material Design, Stakeholders 88
Figura 7.15 Wireframe Material Design, Stakeholders detalle en Director IT 88
Figura 7.16 Wireframe Material Design Stakeholders detalle en ES 89
Figura 7.17 Wireframe Material Design, Tabla de los sistemas del cliente 89

1. Introducción

Los sistemas informáticos llevan evolucionando desde su aparición. Esto es algo que va intrínseco a la disciplina. El software debe evolucionar, adaptarse a las expectativas de los usuarios actuales, debe generar confianza en los usuarios que lo utilizan, en definitiva debe estar bien diseñado.

Nos situamos a comienzos del año 2016, Departamento de Soporte de una gran multinacional del software empresarial llamada de aquí en adelante **XYZ** por cuestiones de confidencialidad. En este contexto surge una idea. Viendo el pobre rendimiento, la empinada curva de aprendizaje para los nuevos trabajadores (sobre todo para nativos digitales), nace la idea de cambiar ese software por otro adaptado a las necesidades y expectativas de los usuarios actuales.

El objetivo del proyecto, del que se ha formado parte a un alto nivel, se podría resumir en:

> "**Rediseñar la experiencia del soporte, cambiando y simplificando la ejecución de los objetivos del departamento. A través de la mejora de las herramientas existentes y de mejora de la documentación**"

En el Departamento de Soporte de XYZ se da asistencia a más de 86000 clientes de 190 países. En este departamento los trabajadores actúan como consultores que ofrecen y recomiendan servicios, sugieren cursos y formación relevante para el cliente así como eventos que podrían hacer que el cliente obtuviera el mayor valor añadido posible de todo lo que ofrece el soporte.

Los clientes a los que se da soporte suelen ser grandes corporaciones, las cuales han realizado inversiones millonarias en el software. Por esos sistemas pasan diariamente cientos de miles de transacciones además de contener valiosísima información sobre el negocio. Del funcionamiento de estos sistemas depende en gran medida el funcionamiento de la propia compañía. Los clientes más grandes pueden llegar a tener cientos de miles de usuarios en sus sistemas por lo que un soporte al que estamos acostumbrados, como usuarios finales, no es suficiente. El soporte que da en este Departamento es un soporte Premium con un elevado coste anual para el cliente. La gran ventaja que ofrece es que se centra en la proactividad, es decir, ofrece la solución al problema antes de que ocurra. Para que el soporte sea un éxito se necesita de la creación de relación con el cliente basada en la confianza mutua.

Este TFG ha sido realizado dentro del Departamento de Soporte de la compañía XYZ por lo que la interacción con los usuarios finales ha sido realmente intensa, al ser compañeros de trabajo.

Los objetivos más relevantes que persigue este trabajo de fin de grado son:

1. Analizar y comprender la problemática existente en los sistemas actuales del Departamento de Soporte.
2. Analizar las necesidades, objetivos, responsabilidades y problemas de los diferentes perfiles (Personas) que interactúan con los sistemas.
3. Analizar los requerimientos del nuevo sistema, basándose en las necesidades del usuario final.
4. Analizar, comprender y aplicar la metodología Design Thinking a un caso real.
5. Diseñar una nueva Experiencia del Usuario, basada en los últimos estilos de diseño junto con los principios más comprobados en la industria.

2. Contexto del problema

Diariamente los trabajadores del Departamento de Soporte de XYZ utilizan un software que dentro del mundo de la Ingeniería del Software se conoce como **software heredado**. Este fue desarrollado hace años, pensado para realizar otra tarea y sobre este se fueron añadiendo funcionalidades a fin de satisfacer las nuevas necesidades y creando como resultado un conjunto de sistemas los cuales cumplen con las necesidades del departamento pero sacrificando el rendimiento y una buena experiencia del usuario.

Como indica Ian Sommervile (Sommerville, 2005) sobre el software heredado, muchas veces debido al tiempo y esfuerzo requeridos para desarrollar un sistema complejo, los grandes sistemas informáticos normalmente tienen un tiempo de vida largo. Por ejemplo, los sistemas militares son diseñados normalmente para un tiempo de vida de 20 años y muchos de los sistemas de control del tráfico aéreo del mundo todavía dependen de software y procesos operativos que fueron desarrollados en un principio en los años 60 y 70. Algunas veces es demasiado caro y peligroso descartar tales sistemas de negocio críticos después de unos pocos años de uso. Su desarrollo continúa durante toda su vida se realizan cambios para satisfacer nuevos requerimientos, nuevas plataformas operativas, y así sucesivamente.

Los sistemas heredados son sistemas informáticos socio-técnicos que han sido desarrollados en el pasado, a menudo usando una tecnología antigua y obsoleta. Estos sistemas no solamente incluyen hardware y software sino también procesos y procedimientos heredados, antiguas formas de hacer cosas que son difíciles de cambiar porque dependen de software heredado. Cambios en una parte del sistema inevitablemente implican cambios en otros componentes. Los sistemas heredados son a menudo sistemas de negocio críticos. Se mantienen porque es demasiado arriesgado reemplazarlos.

En el caso del problema que intenta cubrir este TFG, al igual que se menciona anteriormente, el sistema de soporte es demasiado crítico como para ser sustituido de la noche a la mañana dado que de él depende el soporte de cerca de 86000 clientes por lo que el proyecto tiene pensado estar proyectado en el tiempo hasta principios de 2018.

La empresa XYZ enfoca todos sus esfuerzos en crear soluciones software para todas las necesidades que puedan tener las empresas, indistintamente de la industria a la que pertenece. Actualmente esos negocios, infraestructuras civiles, grandes puertos de carga, grandes líneas de ensamblaje, infraestructuras ferroviarias, todas ellas y muchas más industrias dependen de que los complejos sistemas informáticos sigan funcionando.

Tanto es el impacto que tiene XYZ sobre el funcionamiento de las compañías que en caso de venirse abajo los sistemas de XYZ, por ejemplo en una gran factoría de coches, los coches sencillamente no podrían ensamblarse y la producción se pararía.

El Departamento de Soporte busca el apoyo del cliente a fin de obtener el mayor beneficio. Se necesita de una colaboración muy estrecha entre el trabajador de soporte y del cliente para alcanzar el éxito. Diariamente por el departamento pasan miles de incidentes abiertos por los clientes en los que se pide soporte para realizar una

actualización de los sistemas, descubrir fallas en el rendimiento, buscando formación e información. A este tipo de relación se conoce como reactiva o bajo demanda, es decir, se necesita que el cliente se comunique con soporte para poder ayudarle.

Otro tipo de relación con el cliente, siendo además la más común, es la proactiva, es decir se tiene constancia de la situación del cliente, qué proyectos tiene planeados, cuál es el rendimiento del sistema, cuál es el estado de la formación de los trabajadores. Este otro tipo de relación exige que el trabajador de soporte tenga una gran empatía a fin de poder crear una buena relación con el cliente, entendiendo y posicionándose en la visión del cliente.

La compañía XYZ lleva dando soporte a las grandes corporaciones más de tres décadas. Sin embargo, ha llegado el momento de cambiar el software de soporte al no cumplir con las expectativas de los usuarios actuales cuyos marcos de referencia son las aplicaciones que llevan en sus Smartphone. Estos usuarios buscan la misma simplicidad que tienen en sus teléfonos una vez llegan a su puesto de trabajo.

3. Base Teórica

3.1 Historia de la GUI

El ordenador es uno de los grandes inventos del siglo XX y tal vez uno de los más grandes de la historia de la Humanidad. Se puede considerar a la altura de la imprenta o de la máquina de vapor, máquinas que en su momento revolucionaron la sociedad. Las necesidades que cubrían estos primeros ordenadores eran principalmente el cálculo aritmético y el tratamiento de ingentes cantidades de información (Prieto Molero & Pujol Domenech, 2016). Desde entonces, un elemento que ha ido cogiendo protagonismo es la interfaz del usuario (**GUI** - Graphical User Interface)

Jeremy Reiner (2005) resume la historia de la GUI de la siguiente manera:

A partir de 1975 es cuando el concepto de Computadora Personal aparece por primera vez. Esto se debe principalmente al abaratamiento en el precio de los componentes. Además, empiezan a aparecer diferentes sistemas operativos como pueden ser el MS-DOS 1981, Apple Macintosh 1984 y Linux en 1991.

Uno de las primeras computadoras personales fue la Xerox Alto 1973, además de ser la primera computadora en contar con una interfaz gráfica, una metáfora de escritorio y la primera en contar con un ratón. Sin embargo la primera Computadora Personal en contar con un entorno gráfico que consiguió ganarse al público fue la Apple Lisa 1983. El primer SO de Microsoft en contar con un entorno grafico fue el Windows 1.0 presentado en 1985.

3.1.1 1980

Apple es muchas veces considerada como una de las pioneras en de las GUI (Graphical User Interface). El Apple Lisa fue pionera en muchas innovaciones que ahora consideramos normales como puede ser el utilizar un icono para indicar cada uno de los diferentes documentos o aplicaciones y además contaba con el primer menú de despliegue. Otras innovaciones fueron la utilización de atajos de teclado para las acciones más utilizadas además creó algunas convecciones que aún hoy en día utilizamos, como la papelera de reciclaje para los documentos borrados.

El ratón con el que contaba Lisa tenía un solo botón. Al necesitar la interfaz dos tipos de acciones, seleccionar y ejecutar, se creó el concepto de doble-click el cual vino a solventar este problema. Posteriormente este concepto de doble-click se convirtió en un standard de la industria. El concepto grafico de "*drag-and-drop*", arrastrar y soltar en castellano, apareció por primera vez en el Apple Lisa.

Uno de las principales aportaciones de Apple en estos primeros sistemas grafico fue la posibilidad de saber qué ventana estaba activa respecto a otra a través de las sombras de la parte superior de la ventana, algo aún presente en la actualidad.

La compañía de Redmond lanzó al mercado en 1985 el Windows 1.0. Si bien en las primeras imágenes mostradas en 1983 el sistema carecía de una GUI, en su versión final contaba con una GUI a color. Esta primera versión de Windows contaba con muchas de las funcionalidades con que cuenta a día de hoy como puede ser la barra desplazamiento, el control de las ventanas y menús. Como curiosidad Microsoft tuvo acceso a las primeras versiones del Apple Lisa lo que marcó el futuro de la GUI de Windows. Microsoft presentó en 1987 la que fue la versión 2.0 de su OS plagado de polémica dado que el "*look and feel*" de su OS hizo que Apple le demandara perdiendo el caso Apple en los tribunales.

3.1.2 1990

Ya en los años 90 Microsoft presento la que fue la primera GUI altamente popular en la era PC, Windows 3.0. A pesar de que le faltaban muchas funcionalidades presentes en Macintosh, esta versión contaba con muchas otras y un aspecto novedoso para la época lo que hizo que vendieran millones de copias. La salida de Windows 95 estableció los cimientos de lo que serían los pilares de Windows GUI de la próxima década.

En esta misma década Apple presento una nueva GUI llamada Aqua para su nuevo sistema operativo, ahora llamado Mac OS X. Aqua introdujo algunos conceptos técnicos aún utilizados como son el mantener todas las ventanas en un doble buffer de memoria lo que mejoraba la experiencia del usuario final. Además, introdujo muchas mejoras visuales como las animaciones al minimizar y al maximizar una ventana así como la posibilidad de visualizar una miniatura de todas las ventanas que tenías abiertas sin tener la necesidad de ir cambiando entre ellas.

Todo sistema informático debe estar controlado por un software que actúe como soporte lógico, que le permita realizar las tareas específicas para las que fue diseñado. Esto hace introducir el concepto de Software que se cubrirá a continuación en la siguiente sección

3.2 Software

Como indica Roger S. Pressman (2010), el software puede definirse como:

1) Instrucciones (programas de cómputo) que cuando se ejecutan proporcionan las características, función y desempeño buscados; 2) estructuras de datos que permiten que los programas manipulen de forma adecuada la información, y 3) información descriptiva tanto en papel como en formas virtuales que describen la operación y uso de los programas.

A diferencia del hardware el software es un elemento lógico y por tanto muchas de sus propiedades difieren:

1) El software se desarrolla no se fabrica en el sentido clásico de la palabra.
2) El software no se desgasta. La tasa de fallos de un software es muy diferente de la tasa de fallos de un componente hardware, la del software es más parecida a una función logarítmica decreciente, en la que a lo largo del tiempo el número de errores va disminuyendo, puesto que se van solucionando errores. Sin embargo esta función idealizada es difícil que se dé, dado que esos cambios suelen hacer que otros aparezcan. Pero poco a poco esta línea de fallos va decreciendo

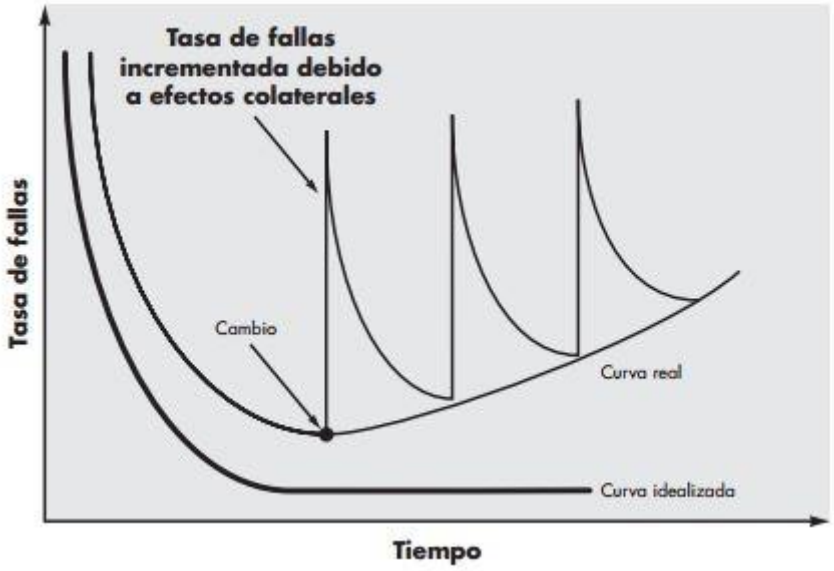


Figura 3. 1 *Tasa de fallos del software a lo largo del tiempo.*

Fuente: (Pressman, 2010)

Existen en la actualidad siete diferentes categorías en las que englobar todos los tipos de software.

- Software de sistema: Estos programas están desarrollados para dar soporte a otros programas, en este grupo englobamos los compiladores, editores, los exploradores de archivos, componentes del propio sistema operativo, controladores, etc. Lo que caracteriza a todo este grupo de programas es una alta

interacción con el hardware, uso de recursos compartidos y uso de estructuras complejas de datos.

- Software de aplicación: Solventan necesidades específicas de los negocios.
- Software científico: Normalmente siempre han sido softwares caracterizados por ser devorares de números. Pero a día de hoy el diseño asistido por ordenador, sistemas de simulación, sistemas de secuenciación genómica entran dentro de este grupo.
- Software incrustado: Se caracteriza por formar parte de un producto final y controlar las características y funciones del mismo. En este grupo entra el software de control de un lavavajillas, lavadoras, sistema de frenado de un automóvil, etc.
- Software de línea de productos: Está diseñado para solventar las necesidades de muchos consumidores finales, ya sea un mercado particular o un mercado global. En este grupo entran los procesadores de texto, administradores de Base de Datos, etc.
- Aplicaciones Web: Son software desplegado en la red que puede tener desde capacidades de computo a estar integradas con bases de datos corporativas, etc.
- Software de inteligencia artificial: En este grupo entran algunos tipos de software como Siri o Cortana de reconocimiento de voz a sistemas capaces de mantener una conversación en lenguaje natural.

Cientos de miles de programas de cómputo caen en uno de los siete dominios amplios de aplicación que se estudiaron en la subsección anterior.

Los programas antiguos -que es frecuente denominar software heredado- han sido centro de atención y preocupación continuas desde la década de 1960. Dayani-Fard (1999) y sus colegas describen el software heredado de la siguiente manera:

> *Los sistemas de software heredado [...] fueron desarrollados hace varias décadas y han sido modificados de manera continua para que satisfagan los cambios en los requerimientos de los negocios y plataformas de computación. La proliferación de tales sistemas es causa de dolores de cabeza para las organizaciones grandes, a las que resulta costoso mantenerlos y riesgoso hacerlos evolucionar.*

Liu y sus colegas (1998), amplían esta descripción al hacer notar que "muchos sistemas heredados continúan siendo un apoyo para las funciones básicas del negocio y son 'indispensables' para éste". Además, el software heredado se caracteriza por su longevidad e importancia crítica para el negocio. Desafortunadamente, en ocasiones hay otra característica presente en el software heredado: *mala calidad*. Hay veces en las que los sistemas heredados tienen diseños que no son susceptibles de extenderse, código confuso, documentación mala o inexistente, casos y resultados de pruebas que nunca se archivaron, una historia de los cambios mal administrada, la lista es prácticamente interminable.

Si el sistema es fundamental para el negocio, ¿Qué hacer con el sistema? La respuesta más razonable, aunque suene fuerte, es no hacer nada si el sistema funciona. Sin embargo, llegado el momento el software deberá adaptarse al futuro, ya sea por la llegada de una nueva tecnología de Base de Datos, una actualización del Sistema Operativo, un

simple cambio en la Arquitectura de las Redes o un cambio en las necesidades del negocio. Llegado a este punto al no poder adaptarse a esa nueva característica muchas veces lo más sensato es desarrollar una nueva aplicación, que contenga una metodología moderna que tenga en cuenta la posible evolución del software.

A fin de crear un software que sea adaptable para el futuro, hemos de aplicar un proceso de modelado de software conociéndose a esta disciplina como Ingeniería del Software.

El **IEEE** (Institute of Electrical and Electronics Engineers) desarrolló la siguiente definición de la ingeniería del software:

> 1) La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software; es decir, la aplicación de la ingeniería al software.
>
> 2) El estudio de enfoques según el punto 1.

Como menciona Pressman (2010), la ingeniería del software se asienta sobre un ***compromiso con la calidad***, además de:

- El ***proceso*** se define como la estructura que debe establecerse para la obtención eficaz de tecnología de ingeniería de software. El *proceso* de software forma la base para el control de la administración de proyectos de software, y establece el contexto en el que se aplican métodos técnicos.
- Los ***métodos*** de la ingeniería de software proporcionan la experiencia técnica para elaborar software. Incluyen un conjunto amplio de tareas, como comunicación, análisis de los requerimientos, modelación del diseño, construcción del programa, pruebas y apoyo.
- Las **h*erramientas*** de la ingeniería de software proporcionan un apoyo automatizado para el proceso y los métodos.

Cuando se está desarrollando un producto o un software es muy importante realizar una serie de pasos predecibles que, a fin de cuentas, nos ayuden a saber qué viene después. A este mapa que nos ayuda a saber qué paso viene después se le conoce como proceso del software.

3.3 Modelos de Proceso

Como indica Pressman (2010) los procesos se subdividen en cinco fases:

- **Comunicación.** Antes de que comience cualquier trabajo técnico, tiene importancia crítica comunicarse y colaborar con el cliente (y con otros participantes). Se busca entender las necesidades de este cliente, a fin de cuentas cuales son los requerimientos de este proyecto.
- **Planeación.** Se puede definir como las tareas técnicas por realizar, los riesgos probables, los recursos que se requieren, los productos del trabajo que se obtendrán y una programación de las actividades.
- **Modelado.** En este paso el ingeniero de software crea modelos arquitectónicos a fin de entender mejor los requerimientos del software y el diseño que los satisfará.
- **Construcción.** Esta actividad combina creación del código y las pruebas que se requieren para descubrir errores en éste.
- **Despliegue.** El software se entrega al consumidor que lo evalúa y que le da retroalimentación, misma que se basa en dicha evaluación.

Estas cinco fases se siguen en todos los proyectos tanto desde los más pequeños como hasta en los proyectos de las grandes compañías de software. La forma en cómo se siguen y agrupan estos pasos es en lo que difieren los distintos modelos de software. Los modelos se han agrupado normalmente en dos grandes grupos: Modelos de Proceso Prescriptivo (o Tradicionales) y las Metodologías Ágiles, aunque están empezando aparecer otras más novedosas como es el Design Thinking, que posteriormente se explicará.

Por otra parte Ian Sommerville (2005), explica sobre los modelos de procesos lo siguiente:

Un modelo de procesos del software es una descripción simplificada de un proceso del software que presenta una visión de ese proceso. Estos modelos pueden incluir actividades que son parte de los procesos y productos de software y el papel de las personas involucradas en la ingeniería del software.

3.3.1 Metodologías Tradicionales

Estas metodologías surgieron con la intención de poner orden en el desarrollo de software. Dependiendo de cómo organicemos las cinco fases mencionadas anteriormente y de cómo saltemos a la siguiente surgen las diferentes metodologías tradicionales, (Pressman, 2010) agrupa las metodologías tradicionales en:

3.3.1.1 Modelo en Cascada

El *modelo de la cascada*, a veces llamado *ciclo de vida clásico* sugiere un enfoque sistemático y secuencial para el desarrollo del software, que comienza con la

especificación de los requerimientos por parte del cliente y avanza a través de planificación, modelado, construcción y despliegue, para concluir con el apoyo del software terminado.

El modelo de la cascada es el paradigma más antiguo de la ingeniería de software. Sin embargo, en las últimas tres décadas las críticas hechas al modelo han ocasionado que incluso sus defensores más obstinados cuestionen su eficacia. Hanna (1995).

Por otro lado, Ian Sommerville (2005), establece el modelo en cascada dividido en diferentes etapas:

1. *Análisis y definición de requerimientos*. Los servicios, restricciones y metas del sistema se definen a partir de las consultas con los usuarios. Entonces, se definen en detalle y sirven como una especificación del sistema.
2. *Diseño del sistema y del software*. El proceso de diseño del sistema divide los requerimientos en sistemas de hardware o software. Establece una arquitectura completa del sistema. El diseño del software identifica y describe las abstracciones fundamentales de los sistemas software y sus relaciones.
3. *Implementación y prueba de unidades*. Durante esta etapa, el diseño del software se lleva a cabo como un conjunto de unidades de programas. La prueba de unidades implica verificar que cada una cumpla su especificación
4. *Integración y prueba del sistema*. Los programas o las unidades individuales de programas se integran y prueban como un sistema completo para asegurar que cumpla los requerimientos del software. Después de las pruebas, sistemas es entregado al cliente
5. *Funcionamiento y mantenimiento*. Esta fase es la más larga del ciclo de vida del software. El sistema es instalado y se pone en funcionamiento práctico. El mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida.

3.3.1.2 Modelo Incremental

Si los requerimientos iniciales del software están razonablemente bien definidos pero el alcance general del esfuerzo de desarrollo imposibilita un proceso lineal; si, además, tal vez hay una necesidad imperiosa de dar rápidamente cierta funcionalidad limitada de software a los usuarios y aumentarla en las entregas posteriores de software, en estas ocasiones un modelo incremental puede ayudar a solucionar estas situaciones.

Cuando se utiliza un modelo incremental, es frecuente que el primer incremento sea el *producto fundamental*. Es decir, se abordan los requerimientos básicos pero no se proporcionan muchas características extras. Un vez que el cliente está utilizando esta primera versión puede servir como testeo para futuras actualizaciones.

El desarrollo incremental es útil en particular cuando no se dispone de personal para la implementación completa del proyecto en el plazo establecido por el negocio. Los primeros incrementos se desarrollan con pocos trabajadores. Si el producto básico es bien recibido, se tiene suficiente peso como para poder incorporar más gente al proyecto.

3.3.1.3 Modelo de proceso evolutivo

Dado que nos encontramos en un mundo complejo y evolutivo en el que los modelos de negocio cambian constantemente y por tanto las necesidades de este software también necesitan cambiar. Para adaptar esta tendencia al desarrollo del software, en la que una trayectoria rectilínea como en el modelo en cascada no resulta realista. Se necesita un modelo de proceso diseñado explícitamente para adaptarse a un producto evolutivo a lo largo del tiempo

Los modelos evolutivos son iterativos. Se caracterizan por la manera en la que permiten desarrollar versiones cada vez más completas del software. Existen dos subtipos de modelos evolutivos:

Hacer prototipos. Es frecuente que un cliente defina un conjunto de objetivos generales para el software pero que no identifique los requerimientos detallados para las funciones y características. El ideal es que el prototipo sirva como mecanismo para identificar los requerimientos del software. El prototipo sirve como "la primera vista" y los expertos recomiendan una vez creado que sea desechado.

El modelo espiral. Este modelo fue originalmente propuesto por Boehn (1988). Este es un modelo evolutivo del proceso del software y se acopla con la naturaleza iterativa de hacer prototipos con los aspectos controlados y sistémicos del modelo de cascada. Tiene el potencial para hacer un desarrollo rápido de versiones cada vez más completas. El modelo de flujo sigue el de una espiral que, con lo que el software se desarrolla en una serie de entregas evolutivas. Durante las primeras iteraciones, lo que se entrega puede ser un modelo o prototipo. En las iteraciones posteriores se producen versiones cada vez más completas del sistema.

Por otra parte, Ian Sommerville (2005) señala sobre el **modelo incremental**, que más que representar el proceso del software como una secuencia de actividades con la retrospectiva de una actividad a otra es un modelo que se representa en una espiral y, cada clico de la espiral representa una fase del proceso del software, a su vez cada ciclo de la espiral se divide en cuatro sectores:

1. *Definición de objetivos*. En esta fase se definen los objetivos específicos, se identifican las restricciones del proceso y el producto, y se traza un plan detallado de gestión.
2. *Evaluación y reducción de riesgo*. Se lleva a cabo un análisis detallado para cada uno de los riesgos del proyecto identificados. Se definen los pasos para reducir dichos riesgo.
3. *Desarrollo y validación*. Después de la evaluación de riegos, se elige un modelo para el desarrollo del sistema
4. *Planificación*. El proyecto se revisa y se toma la decisión de si se debe continuar con un ciclo posterior de la espiral. Si se decide continuar se desarrollan los planes de la siguiente fase del proyecto.

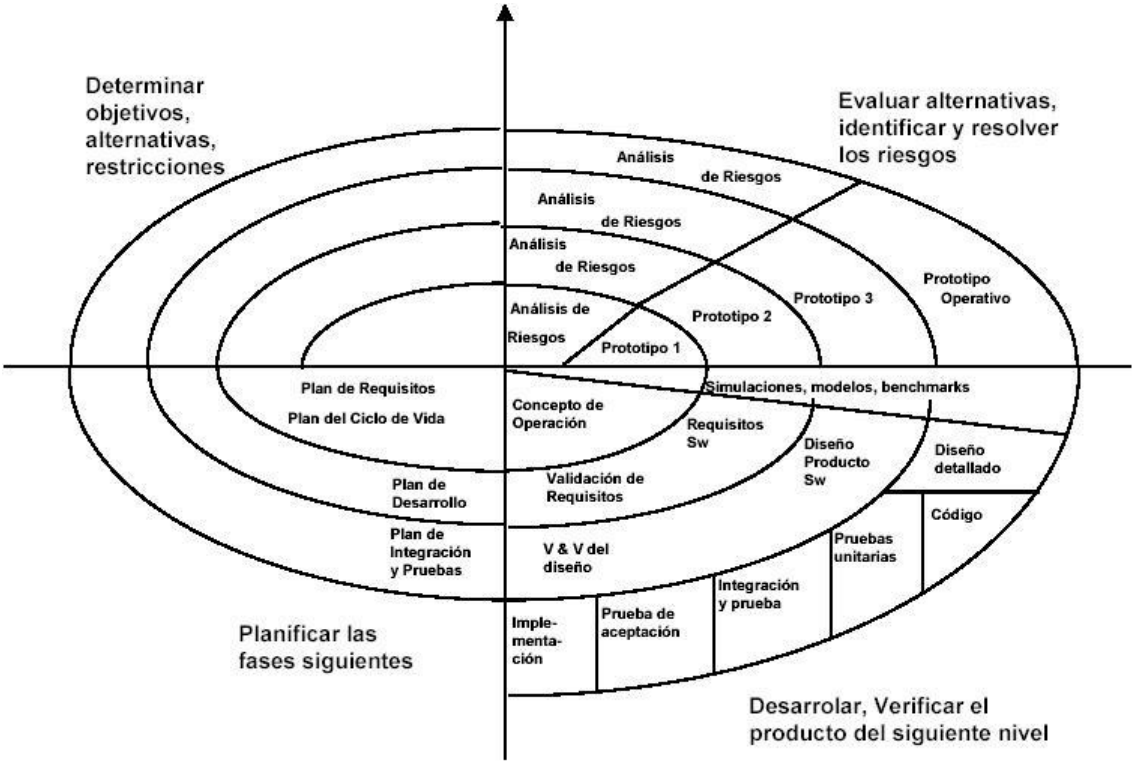


Figura 3. 2 *Ciclo de vida del modelo en espiral*

Fuente: (Sommerville, 2005)

3.3.1.4 Modelo Concurrente

El modelo de desarrollo concurrente permite que un equipo de software represente elementos iterativos y concurrentes de cualquiera de los modelos de proceso descritos anteriormente.

El modelado concurrente define una serie de eventos que desencadenan transiciones de un estado a otro para cada una de las actividades, acciones o tareas de la ingeniería de software. El modelado concurrente es aplicable a todos los tipos de desarrollo de software y proporciona un panorama apropiado del estado actual del proyecto. En lugar de confinar las actividades, acciones y tareas de la ingeniería de software a una secuencia de eventos, define una red del proceso. Cada actividad, acción o tarea de la red existe simultáneamente con otras actividades, acciones o tareas. Los eventos generados en cierto punto de la red del proceso desencadenan transiciones entre los estados.

3.3.1.5 Modelo unificado de Rational

Ian Sommerville (2005) resume sobre el Proceso Unificado de Rational (**RUP**) que este es un ejemplo de un modelo de proceso que proviene del trabajo en el UML y el trabajo asociado al Proceso Unificado de Desarrollo de Software, Rumbaugh (1999). Reúne elementos de todos los modelos de procesos genéricos e ilustra buenas prácticas en la especificación y el diseño.

El RUP se describe normalmente desde tres perspectivas:

1. *Una perspectiva dinámica*, que muestra las fases del modelo sobre el tiempo.
2. *Una perspectiva estática,* que muestra las actividades del proceso que se representan.
3. *Una perspectiva práctica,* que sugiere buenas prácticas a utilizar durante el proceso.

El RUP es un modelo en fases, este identifica cuatro fases diferentes en el proceso del software, las fases en el RUP están mucho más relacionadas con asuntos de negocio más que con los técnicos. Las fases con las que cuenta RUP son:

- *Inicio*. El objetivo de la fase de inicio es el de establecer un caso de negocio para el Sistema. Se deben identificar todas las entidades externas (personas y sistemas) que interactuarán con el Sistema y definir estas interacciones. Esta información se utiliza entonces para evaluar la aportación que el sistema hace al negocio. Si esta aportación es de poca importancia, se puede cancelar el proyecto después de esta fase
- *Elaboración*. Los objetivos de la fase de elaboración son desarrollar una comprensión del dominio del problema, establecer un marco de trabajo arquitectónico del sistema, desarrollar el plan del proyecto e identificar los riesgos clave del proyecto. Al terminar esta fase, se debe tener un modelo de los requerimientos del sistema, una descripción arquitectónica y un plan de desarrollo del software.
- *Construcción*. La fase de construcción fundamentalmente comprende el diseño del sistema, la programación y las pruebas. Durante esta fase se desarrollan e integran las partes del sistema. Al terminar esta fase, debe tener un sistema operativo y la documentación correspondiente lista para entregarla a los usuarios.
- *Transición*. La fase final del RUP se encarga de entregar al usuario final, haciéndolo trabajar en un entorno real. Esto se deja de lado en la mayor parte de los modelos de procesos del software pero es, en realidad, una actividad de alto costo y a veces problemática. Al terminar esta fase, se debe tener un sistema software documentado que funciona correctamente en su entorno operativo.

3.3.2 Metodologías Ágiles

Como indica Pressman (2010), en el año 2001, Kent Beck (2001) y otros 16 notables desarrolladores de software, escritores y consultores (grupo conocido como la "Alianza Ágil") firmaron el "Manifiesto por el desarrollo ágil de software". En él se establecía lo siguiente:

Estamos descubriendo formas mejores de desarrollar software, por medio de hacerlo y de dar ayuda a otros para que lo hagan. Ese trabajo nos ha hecho valorar:

1. *Los individuos y sus interacciones*, sobre los procesos y las herramientas
2. *El software que funciona*, más que la documentación exhaustiva
3. *La colaboración con el cliente*, y no tanto la negociación del contrato
4. *Responder al cambio*, mejor que apegarse a un plan
5. Es decir, si bien son valiosos los conceptos que aparecen en segundo lugar, valoramos más los que aparecen en primer sitio.

Este manifiesto supuso un antes y un después en el Desarrollo del Software, desde ese momento se dividió entre la vieja guardia que defendía el Desarrollo Tradicional y las nuevas generaciones que defienden estas nuevas metodologías.

Pero, ¿qué significa agilidad? Según Ivar Jacobson (2002) la agilidad se refiere a:

> *La agilidad se ha convertido en la palabra mágica de hoy para describir un proceso del software moderno. Todos son ágiles. Un equipo ágil es diestro y capaz de responder de manera apropiada a los cambios. El cambio es de lo que trata el software en gran medida. Hay cambios en el software que se construye, en los miembros del equipo, debidos a las nuevas tecnologías, de todas clases y que tienen un efecto en el producto que se elabora o en el proyecto que lo crea. Deben introducirse apoyos para el cambio en todo lo que se haga en el software; en ocasiones se hace porque es el alma y corazón de éste. Un equipo ágil reconoce que el software es desarrollado por individuos que trabajan en equipo, y que su capacidad, su habilidad para colaborar, es el fundamento para el éxito del proyecto.*

Pero además de todo esto la agilidad debe incluir partes que sí que están en el manifiesto e Jacobson (2002) no lo menciona, como puede ser la mayor colaboración entre el equipo y el cliente o entre los compañeros del equipo.

En las metodologías agiles el cliente debe pasar a ser uno más del equipo, intentando a toda costa eliminar los términos "nosotros y ellos", puesto que el software no se desarrolla para "nosotros" sino para "ellos".

José H. Canós, Patricio Letelier y Mª Carmen Penadés (2003) establecen la siguiente tabla comparativa entre las Metodologías Ágiles y las Metodologías Tradicionales:

Metodologías Tradicionales	Metodologías Ágiles
Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo	Basadas en heurísticas provenientes de prácticas de producción de código
Cierta resistencia a los cambios	Especialmente preparados para cambios durante el proyecto
Normas impuestas externamente	Normas impuestas internamente (por el equipo)
Proceso mucho más controlado, con numerosas políticas/normas	Proceso menos controlado, con pocos principios
Existe un contrato prefijado	No existe contrato tradicional o al menos es bastante flexible
El cliente interactúa con el equipo de desarrollo mediante reuniones	El cliente es parte del equipo de desarrollo
Grupos grandes y posiblemente distribuidos	Grupos pequeños (<10 integrantes) y trabajando en el mismo lugar
Más artefactos	Pocos artefactos
Más roles	Pocos roles
La arquitectura del software es esencial y se expresa mediante modelos	Menos énfasis en la arquitectura del software

Tabla 3. 1 *Tabla comparativa Metodologías Tradicionales contra Metodologías Agiles*

Fuente: (Canós, Letelier, & Penadés, 2003)

Una vez ya hemos definido cuales son los principios de las Metodologías Ágiles y se ha definido que es la agilidad en los procesos del software, es el momento de resaltar algunas de las principales metodologías:

3.3.2.1 Programación Extrema (XP)

José H. Canós, Patricio Letelier y Mª Carmen Penadés (2003) definen la XP como una metodología centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico

El ciclo de desarrollo se divide (a grandes rasgos) en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelta al paso 1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración.

El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega, Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

3.3.2.2 Otras Metodologías Ágiles

Finalmente, José H. Canós, Patricio Letelier y Mª Carmen Penadés (2003) establecen el siguiente listado de metodologías ágiles, las cuales difieren en características y en algunos aspectos muy específicos:

- **SCRUM**. Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas *Sprints*, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.

- **Crystal Methodologies**. Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn. En esta el desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo *Crystal Clear* (3 a 8 miembros) y *Crystal Orange* (25 a 50 miembros).
- **Dynamic Systems Development Method** (DSDM). Define el marco para desarrollar un proceso de producción de software. Nace en 1994 con el objetivo de crear una metodología RAD (Rapid Application Development) unificada. Sus principales características son: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación. Las tres últimas son iterativas, además de existir realimentación a todas las fases.
- **Adaptive Software Development** (ASD). Su impulsor es Jim Highsmith. Sus principales características son: iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios. El ciclo de vida que propone tiene tres fases esenciales: especulación, colaboración y aprendizaje. En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda desarrollan las características y finalmente en la tercera se revisa su calidad, y se entrega al cliente. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo.
- **Feature Driven Development** (FDD). Define un proceso iterativo que consta de 5 pasos. Las iteraciones son cortas (hasta 2 semanas). Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software. Sus impulsores son Jeff De Luca y Peter Coad.
- **Lean Development** (LD). Definida por Bob Charette's a partir de su experiencia en proyectos con la industria japonesa del automóvil en los años 80 y utilizada en numerosos proyectos de telecomunicaciones en Europa. En LD, los cambios se consideran riesgos, pero si se manejan adecuadamente se pueden convertir en oportunidades que mejoren la productividad del cliente. Su principal característica es introducir un mecanismo para implementar dichos cambios

Una vez todos los diferentes métodos están definidos, llega el momento de explicar cómo la Ingeniería de Requerimientos ayuda a los equipos de desarrollo a identificar cuáles son las necesidades de los usuarios para los que se está desarrollando el sistema. Sin una buena toma de requerimientos posiblemente las necesidades del cliente no se cubran y se sienta decepcionado con el software.

3.4 Ingeniería de Requerimientos

Antes de realizar cualquier desarrollo es necesario comprender los requerimientos que tendrá ese software. A esa tarea se le conoce como Ingeniería de Requerimientos o Requisitos. Pressman (2010) define esta disciplina así y divide sus fases en:

El espectro amplio de tareas y técnicas que llevan a entender los requerimientos se denomina *ingeniería de requerimientos*. Desde la perspectiva del proceso del software, la ingeniería de requerimientos es una de las acciones más importantes de la ingeniería de software que comienza durante la actividad de comunicación y continúa en la de modelado. Debe adaptarse a las necesidades del proceso, del proyecto, del producto y de las personas que hacen el trabajo.

La ingeniería de requerimientos tiende un puente para el diseño y la construcción. Este puente principia en los pies de los participantes en el proyecto, donde se definen las necesidades del negocio, se describen los escenarios de uso, se delinean las funciones y características y se identifican las restricciones del proyecto. Pero sin importar el punto de arranque, el recorrido por el puente lo lleva a uno muy alto sobre el proyecto, lo que le permite examinar el contexto del trabajo de software que debe realizarse; las necesidades específicas que deben abordar el diseño y la construcción; las prioridades que guían el orden en el que se efectúa el trabajo, y la información, las funciones y los comportamientos que tendrán un profundo efecto en el diseño resultante.

La ingeniería de requerimientos proporciona el mecanismo apropiado para entender lo que desea el usuario, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable, especificar la solución sin ambigüedades, validar la especificación y administrar los requerimientos a medida de que se transforman en un sistema funcional Thayer (1997) Incluye siete tareas diferentes: concepción, indagación, elaboración, negociación, especificación, validación y administración. Es importante notar que algunas de estas tareas ocurren en paralelo y que todas se adaptan a las necesidades del proyecto.

- **Concepción**. ¿Cómo inicia un proyecto de software? En ciertos casos, una conversación casual es todo lo que se necesita para desencadenar un trabajo grande de ingeniería de software. Pero en general, la mayor parte de proyectos comienzan cuando se identifica una necesidad del negocio o se descubre un nuevo mercado o servicio potencial.
- **Indagación**. Consiste habar a los usuarios y a otras personas sobre cuáles son los objetivos para el sistema o producto, que´ es lo que va a lograrse, como se ajusta el sistema o producto a las necesidades del negocio y, finalmente, como va a usarse el sistema o producto en las operaciones cotidianas.
- **Elaboración**. La información obtenida del cliente durante la concepción e indagación se expande y refina durante la elaboración. Esta tarea se centra en desarrollar un modelo refinado de los requerimientos identifique distintos aspectos de la función del software, su comportamiento e información.
  La elaboración esta´ motivada por la creación y mejora de escenarios de usuario que describan como interactuara´ el usuario final (y otros actores) con el sistema. Cada escenario de usuario se enuncia con sintaxis apropiada para extraer clases de análisis, que son entidades del dominio del negocio visibles para el usuario

final. Se definen los atributos de cada clase de análisis y se identifican los servicios que requiere cada una de ellas.

- **Negociación**. No es raro que los usuarios pidan más de lo que puede lograrse dado lo limitado de los recursos del negocio. También es relativamente común que distintos usuarios propongan requerimientos conflictivos con el argumento de que su versión es "esencial para nuestras necesidades especiales".
- **Especificación**. El termino especificación tiene diferentes significados para distintas personas. Una especificación puede ser un documento escrito, un conjunto de modelos gráficos, un modelo matemático formal, un conjunto de escenarios de uso, un prototipo o cualquier combinación de estos. Para sistemas grandes normalmente, el mejor enfoque puede ser un documento escrito que combine descripciones en un lenguaje natural con modelos gráficos
- **Validación**. La calidad de los productos del trabajo que se generan como consecuencia de la ingeniería de los requerimientos se evalúa durante el paso de validación. La validación de los requerimientos analiza la especificación a fin de garantizar que todos ellos han sido enunciados sin ambigüedades; que se detectaron y corrigieron las inconsistencias, las omisiones y los errores, y que los productos del trabajo se presentan conforme a los estándares establecidos para el proceso, el proyecto y el producto.

3.4.1 Requerimientos funcionales

Ian Sommerville (2005) sintetiza sobre los requerimientos funcionales:

Los requerimientos funcionales de un sistema describen lo que el sistema puede hacer. Estos requerimientos dependen del tipo de software que se desarrolle, de los posibles usuarios del software y del enfoque general tomado por la organización al redactar los requerimientos. Cuando se expresan como requerimientos del usuario, habitualmente se describen de una forma bastante abstracta. Sin embargo, los requerimientos funcionales del sistema describen con detalle la función de éste, sus entradas y salidas, excepciones, etcétera.

Los requerimientos funcionales para un sistema software se pueden expresar de diferentes formas. Estos requerimientos funcionales del usuario definen los recursos específicos que el sistema debe proporcionar.

En principio, la especificación de requerimientos funcionales de un sistema debe estar completa y ser consistente. La completitud significa que todos los servicios solicitados por el usuario deben estar definidos. La consistencia significa que los requerimientos no deben tener definiciones contradictorias. En la práctica, para sistemas grandes y complejos, es prácticamente imposible alcanzar requerimientos de consistencia y completitud

Una razón de esto es que es fácil cometer errores y omisiones cuando se redactan especificaciones para sistemas grandes y complejos. Otra razón es que los Stakeholders del sistema tienen necesidades diferentes, y a menudo contradictorias. Estas contradicciones pueden no ser obvias cuando los requerimientos se especifican por primera vez, por lo que se incluyen requerimientos contradictorios en la especificación.

Es posible que los problemas surjan solamente después de un análisis más profundo o, a veces, después de que se termine el desarrollo y el sistema se entregue al cliente.

3.4.2 Requerimientos no funcionales

Ian Sommerville (2005) recapitula sobre los requerimientos no funcionales:

Los requerimientos no funcionales, como su nombre sugiere, son aquellos requerimientos que no se refieren directamente a las funciones que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema.

Según Sommerville (2005), los requerimientos no funcionales rara vez se asocian con características particulares del sistema. Más bien, estos requerimientos especifican o restringen las propiedades emergentes del sistema como puede ser especificar el rendimiento del sistema, la protección, la disponibilidad, al igual que otras propiedades. Esto significa que a menudo son más críticos que los requerimientos funcionales particulares. Los usuarios del sistema normalmente pueden encontrar formas de trabajar alrededor de una función del sistema que realmente no cumple sus necesidades. Sin embargo, el incumplimiento de un requerimiento no funcional puede significar que el sistema entero sea inutilizable.

Los requerimientos no funcionales no solo se refieren al sistema software a desarrollar. Algunos de estos requerimientos pueden restringir el proceso que se debe utilizar para desarrollar el sistema.

Los requerimientos no funcionales surgen de las necesidades del usuario, debido a las restricciones en el presupuesto, a políticas de la organización, a la necesidad de interoperabilidad con otros sistemas software o hardware, o a factores externos como regulaciones de seguridad o legislación sobre la privacidad.

Cada día las necesidades de los usuarios y de las compañías cambian, algunas metodologías anteriormente explicadas se centran en crear documentación muy bien estructurada, otras en dar entregas rápidas y constantes al cliente. Otras como Design Thinking, se centran en el usuario, sus necesidades, objetivos y quebraderos de cabeza. Tratando de empatizar con el usuario, a fin de poder ver el mundo a través de sus ojos.

3.5 Design Thinking

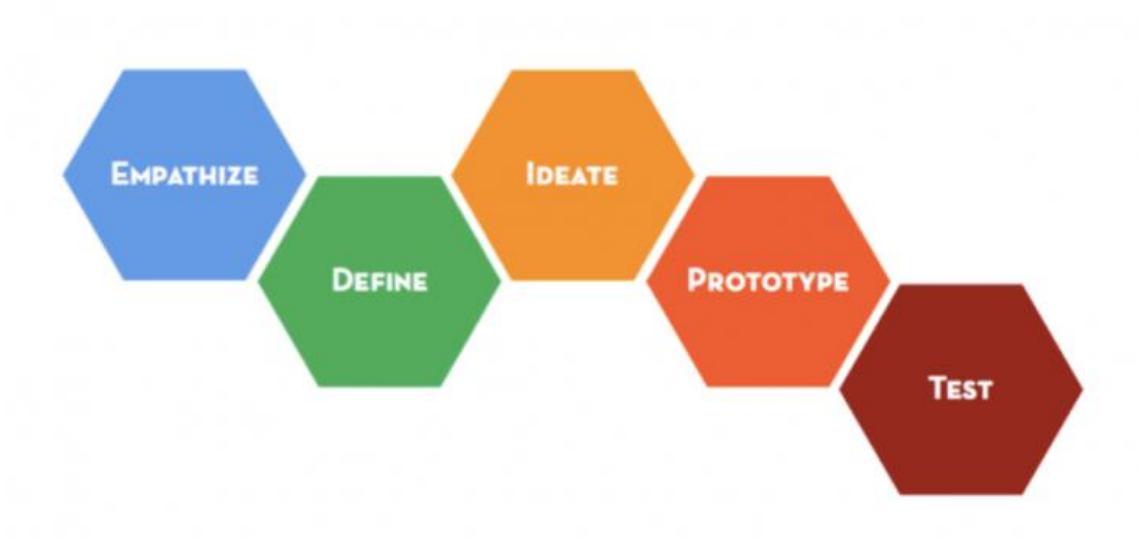


Figura 3. 3 *Fases del Design Thinking*

Fuente: (Hasso-Plattner-Institut, 2016)

Design Thinking es una metodología que ha ido cogiendo protagonismo en los últimos años gracias principalmente a **IDEO** (2016) y al **Instituto de Diseño de Stanford** (2010).

Design Thinking es un enfoque sistemático centrado en las personas que trata de solventar complejos problemas de todos los aspectos de la vida. Esta estrategia va mucho las lejos que las metodologías tradicionales, las cuales abordan las problemas desde el punto de vista de la solvencia técnica, las necesidades de los usuarios y sus requerimientos

Esta estrategia necesita de una retroalimentación continua entre los desarrolladores de la solución y usuarios finales. Design Thinking no sólo se pone en la piel de los usuarios finales, intenta entender sus comportamientos a través de la observación directa y a través de entrevistas. Las diferentes ideas y soluciones que surgen de este proceso se crean prototipos a fin de que los diferentes usuarios finales pueden probarlos retroalimentando de esta manera al equipo de desarrollo. Este proceso sigue durante todo el proceso de desarrollo del producto.

Design Thinking permite innovar y solventar problemas de una manera efectiva gracias al balanceo entre tres fuerzas, Viabilidad técnica, viabilidad económica y los deseos de los usuarios. La innovación se encuentra al encontrar el balance perfecto entre las necesidades de los usuarios, que es viable técnicamente y que sea económicamente, (Hasso-Plattner-Institut, 2016).

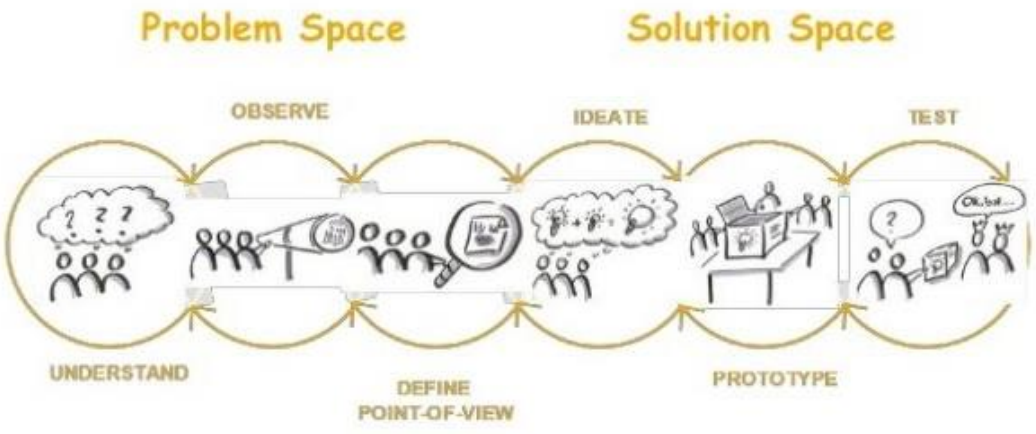


Figura 3. 4 *Agrupación de fases del Design Thinking en Espacios*

Fuente: (SAP SE, 2016)

Algunas compañías como tienes su propia aproximación sobre la metodología Design Thinking, dado que ellos agrupan a las fases de Empatizar y Definir bajo lo que llaman como **Espacio del Problema** y la fase de Idear, Prototipar y Test bajo lo que llaman como **Espacio de la Solución,** (SAP SE, 2016).

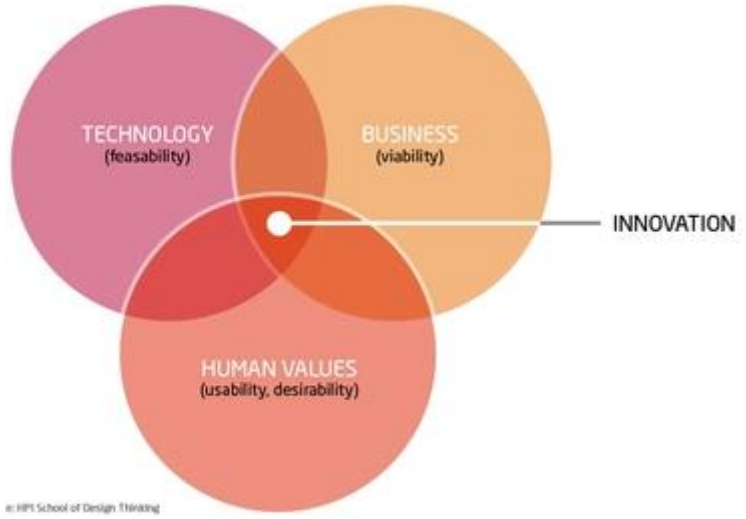


Figura 3. 5 *Combinación de las intenciones del Design Thinking*

Fuente: (Hasso-Plattner-Institut, 2016)

3.5.1 Fases

De acuerdo con Instituto de Diseño de Stanford (2010) Design Thinking cuenta con cinco fases:

1. **Empatizar:**

¿En qué consiste la fase de empatizar?

La empatía es la base del proceso de diseño que esta´ centrado en las personas y los usuarios. Lo básico para ser empático es:

1. Observar: Mirar a los usuarios y sus comportamientos en el contexto en el que viven, donde trabajan. Se debe siempre tratar de observar desde el exterior sin entrometerse, las mejores ideas vienen de estas situaciones.
2. Involucrarse: Es necesario que la conversación fluya, esta puede ser desde una conversación de pasillo, breve o una conversación mucho más estructurada. Es necesario preparar preguntas para siempre mantener la atención de los usuarios
3. Inmersión: experimentar lo que el usuario siente.

Esta fase forma parte del *Espacio del Problema*.

¿Por qué empatizar?

Al ser un diseño centrado en el usuario necesitas de entender a las personas para las que estas diseñando.

Observando lo que las personas hacen y como interactúa con el entorno te das pistas de que piensa y siente. Esto ayuda además a entender que de verdad necesitan. A través de la observación puedes capturar las manifestaciones físicas de sus experiencias, que dice. Esto te dará una perspectiva diferente de la experiencia del usuario. Las mejores soluciones surgen del mejor entendimiento del comportamiento humano.

Relacionarse con los usuarios directamente te revela una tremenda cantidad de información acerca de lo piensan y cuáles son sus principales valores. Algunas veces estos pensamientos no son ni conocidos por el propio usuario, hasta que salen a luz a través de esta relación directa. Relacionarse con el usuario final permite:

- Descubrir las necesidades reales de las personas.
- Identificar a los usuarios correctos para los que desarrollar la solución
- Destapar emociones y pensamientos que guían los comportamientos.

Una vez se consigue empatizar con el usuario, consigues comprender lo que piensa, que siente y que ve. Esta etapa es fundamental para el éxito del proyecto, se le conoce como inmersión, dado que entrar en un mar de aprendizaje del usuario.

¿Cómo empatizar?

Para empatizar se necesita:

- Observar: ver a los usuarios y su comportamiento en su contexto, en sus vidas. Los hallazgos más interesantes se suelen encontrar al observar lo que dicen y lo que de verdad hacen.
- Relacionarse: muchas veces se le llama a esta parte entrevistas, pero debería ser más como una conversación
- Ser informal. Esto sirve para sonsacar información muy relevante, pero para poder sacar información con un significado más profundo es necesario preguntar, el por qué.
- Mirar y observar: es necesario combinar el observar y el relacionarse. Preguntar a los usuarios a que realizar una tarea, decirles que sigan los pasos y mientras tanto preguntar porque hacen eso de esa manera que piensan acerca de este paso. Hacer que los usuarios digan lo que están pensando mientras realizan el paso o mientras interactúan con el objeto.

Transición: Empatiza >> Definir

Desempaquetar: cuando terminan las conversaciones con el cliente es necesario plasmar las conclusiones del trabajo, se necesita procesar todas las cosas que se han observado, visto, se han oído, a fin de poder crear una gran imagen del todo. Este proceso de desempaquetar la información es una oportunidad para compartir lo encontrado con otros desarrolladores. Una buena práctica es plasmar todo lo encontrado en una pared con Pos*its*, esto ayuda a crear mapas mentales. Esto ayuda a crear una síntesis del proceso que te lleva a la siguiente fase.

2. Definir

¿En qué consiste la fase de Definir?

La fase de Definir consiste en aclarar y centrarse en el Espacio del Problema pero siempre teniendo en cuenta con esta fase se termina ese espacio y se salta al Espacio de la Solución.

La dificultad de esta fase consiste en definir un resumen con el significado y las acciones que forman el problema. Hay dos objetivos principales en la fase de diseño, la primera es desarrollar un conocimiento profundo de los usuarios y crear una idea de cómo podría ser la solución, esto permite crear un resumen de cuál es el problema, esto se le suele conocer como "*tu punto de vista, Point of View (POV)*". Tu punto de vista debería ser una declaración que se centra en los usuarios, que permite interiorizar y comprender las necesidades descubiertas durante la fase de empatía.

Esta fase es mucho más que definir el problema sobre el que estás trabajando, el POV es la única visión de lo encontrado en la fase de empatía. Entender el significado del problema, abordar y comprender la verdad permite crear una solución que será acertada.

¿Por qué Definir?

Esta fase es fundamental para la fase de diseño ya que es la fase en la que se expresa cual el problema que se está abordando, cuál es su complejidad… Un POV debería permitir:

- Enfocar y centrar el problema
- Inspirar al equipo

- Permite un marco de referencia para evaluar diferentes ideas
- Da al equipo un respaldo para tomar decisiones en paralelo
- Captura el cuerpo y alma de la gente con la se reunió
- Deberías ser capaz de editarlo mientras sigues aprendiendo del cliente
- Guiar el esfuerzo de la innovación

Transición: Definir >> Idear

En la fase de Definir se especifica el verdadero significado del problema y como es de grande el desafío al que se enfrenta. Un buen POV deberá guiar hasta idear un diseño de cómo solucionar el problema.

Una buena práctica para esta fase es empezar a crear una lista de "¿Cómo deberíamos de….?" Esto ayudara a que la "tormenta de ideas, brainstorming" fluya con naturalidad, si se crea una buena lista puede hacer que las ideas/soluciones se creen casi en masa.

3. Idear

¿En qué consiste la fase de Idear?

Idear es la fase durante del Design Thinking en la que hay que centrarse en la creación de idas. Mentalmente representa el proceso de ir a lo largo de los diferentes términos y conceptos. En esta fase es mucho más útil desvariar y dar ideas que se podría decir que son alocadas, más que el hecho de centrarse en el proceso obvio, de los extremos surgen las mejores soluciones, es más útil tener variedad y diversidad de ideas. Luego se podrá probar la validez de esas alocadas ideas.

¿Por qué Idear?

Se idea para moverse de la fase de identificar el problema (Espacio del Problema) a fin de explorar soluciones para el usuario (Espacio de la Solución). Algunas buenas prácticas para idear son:

- Mantener alejado de las soluciones obvias o aquellas y acercase a las que tienen el potencial de innovar.
- Emplear las perspectivas y potenciales del equipo.
- Descubrir las áreas inexploradas
- Crear cantidad y variedad de opciones innovadoras
- Quitar las soluciones obvias de tu cabeza, esto ayuda al equipo a acercase a la verdadera innovación

Transición: Idear >> Prototipado

Para evitar perder todo el potencial innovador se deben crear diferentes prototipos y llevarlos al mundo real para probarlos. A fin de encontrar esas ideas que se van a prototipar lo mejor que se puede realizar es una votación de cuáles son las mejores ideas para el equipo, basado en ella prototipar las tres-cuatro más votadas.

Lo fundamental de esta fase es ser consciente de cuando se están generando ideas y de cuando se están evaluando las ideas.

4. Prototipado

¿En qué consiste la fase de Prototipado?

Esta fase consiste en llevar las ideas y pensamientos al mundo real, al mundo tangible. Un prototipo puede ser cualquier cosa que tome forma de algo tangible, un muro lleno de *Posits*, una actividad de roles, un objeto normal, una interfaz o incluso una Storyboard. La resolución del prototipo debería ser proporcional con el progreso del proyecto. En las primeras iteraciones necesitas crear prototipos de una manera muy rápida para poder explorar e investigar las diferentes posibilidades/ideas.

Los prototipos más exitosos son en los que los diseñadores y los usuarios pueden interactuar con él. Con lo que se vaya aprendiendo en las diferentes iteraciones ayudan a identificar soluciones exitosas.

¿Por qué Prototipar?

Tradicionalmente el prototipado se ha utilizado para probar funcionalidades. Pero el prototipado en el Design Thinking se utiliza para muchas más razones:

- Ganar empatía: prototipar permite profundizar en los hallazgos encontrados en el Espacio del Problema.
- Explorar: construir para poner a prueba. Desarrollar múltiples ideas y opciones
- Testar: creando prototipos se pueden probar y perfeccionar soluciones con los propios usuario finales.
- Inspirar: inspirar tanto a ti mismo como a otros miembros del equipo, Stakeholders, clientes, creando una visión de la solución.

Transición: Prototipar >> Test

Prototipar y testear son fases que prácticamente van cogidos de la mano. Lo que estas intentando poner a prueba y la manera de cómo lo vas a poner a prueba es muy importante de considerar antes de ponerse a prototipar.

No se puede pensar en poner directamente el prototipo en manos del usuario sin haber pensado de cómo se va a hacer para que el usuario lo pruebe. La falta de este plan puede desechar ideas que realmente son innovadoras, pero no se han sabido llegar a explicar.

5. Test

¿En qué consiste la fase de Test?

Esta fase comienza cuando se pide Feedback de los prototipos que se han creado, se da la oportunidad del usuario de empatizar con el diseñador que está intentando de diseñar una solución.

Testar da la oportunidad de refinar soluciones y hacerlas mejor. Esta fase es la más interactiva de todas, en la cual se ponen nuestras soluciones de baja fidelidad a prueba, el contexto en el que vive el usuario final. Aquí es cuando se descubre si el prototipo puede llegar a funcionar o debe ser desechado

¿Por qué Testar?

- Permite refinar los prototipos y las posibles soluciones: testar te posibilita saber si habrá segunda iteración con un prototipo o si debe ser desechado. Algunas veces esto significa volver a la pizarra.
- Permite aprender más del usuario. Poner a prueba las ideas da la oportunidad de observar la empatía de los usuarios y continuar construyendo una relación, de aquí pueden salir soluciones inesperadas.
- Poner a prueba to POV. Algunas veces sólo al testar una idea revela si se había dado en el clavo o si se había errado sobre el blanco.

¡Es hora de Iterar!

Iterar es fundamental para un buen diseño. Hay que realizar este proceso completo varias veces. En cada iteración se aprenderán cosas nuevas que permitirán acercarse cada vez más a la solución acertada. Para iterar se debe volver a hacer brainstormings con grupos diferentes a fin de entender puntos de vista diferentes.

3.5.2 Métodos

Existen multitud de métodos/técnicas diferentes para conseguir el mayor potencial para conseguir el mayor potencial en cada una de las fases. Ahora se resumirán algunos de los métodos más utilizados en las metodologías de Design Thinking según el Instituto de Diseño de Stanford:

1. **Asumir siempre la actitud de un principiante.**

¿Por qué asumir la actitud de un principiante?

Cuando se realizan talleres de Design Thinking es muy imponte dejar a un lado mucha de la experiencia que se pueda tener de proyectos anteriores o de la propia experiencia. Este conocimiento es tremendamente útil sólo si se utiliza en el momento adecuado y con intencionalidad. Las suposiciones pueden ser una idea equivocada o estereotipos y pueden realmente limitar la empatía que se es capaz de construir con el usuario y los propios miembros del equipo.

¿Cómo asumir la actitud de un principiante?

- **No juzgar**. Hay que limitarse a observar y a crear una relación con el usuario sin influir y poner en valor sus acciones, circunstancias, decisiones o problemas.
- **Preguntar sobre todo.** Pregunta de todo lo que se dice durante y mientras se esté con el usuario, incluso las cosas que se tienen claras. Hay que llegar al punto de empatía de ser capaz de ver el mundo de la manera que lo percibe el usuario.
- **Se curioso.** Hay que esforzarse por querer saberlo todo, especialmente de las circunstancias que resultan familiares o incomodas.
- **Encontrar patrones.** Sigue los hilos de la conversación más interesantes que surgen mientras se interactúa con el usuario.
- **Escuchar no oír.** Hay que absorber todo lo que el usuario dice o menciona, y de la manera que lo dice, sin llegar a pensar en que en la próxima cosa que se va a decir.

2. ¿Qué? - ¿Cómo? - ¿Por qué?

En la fase de Empatía y mientras se está observando, ¿Qué? - ¿Cómo? - ¿Por qué? es una herramientas que permite llegar a niveles de detalle mayores. Esta simple estructura permite que el usuario se mueva de simples observaciones concretas de lo que sucede a una observación mucho más abstracta con un potencial emocional mucho mayor.

¿Cómo usar "*¿Qué? - ¿Cómo? - ¿Por qué?*"?

- Dividir una hoja en tres secciones qué, cómo y por qué.
- Empezar con observaciones concretas. ¿Qué está haciendo la persona a la que se está observando? Hay que utilizar frases descriptivas con adjetivos descriptivos.
- Comprender lo que hace. ¿Cómo realiza lo que está haciendo el usuario? ¿Le requiere un esfuerzo?, ¿Parece frustrado?, ¿Esta actividad está teniendo un impacto en el estado de ánimo del usuario ya se positivamente o negativamente? Otra vez, hay que escribir las respuestas de la manera más descriptiva posible.
- Dar un paso atrás hacia la interpretación. ¿Por qué el usuario que estas abarcando está haciendo eso de esa manera en particular? Este paso requiere normalmente que se sepa el estado emocional del usuario. Este paso revela conjeturas que deberías poner a prueba más adelante con los usuarios.

3. Preparación para la entrevista

El tiempo con los usuario es muy preciado y es necesario sacar el mayor partido a de él. Mientras que por un lado es necesario dejar hueco a la espontaneidad y a los descubrimientos inesperados que surgen en la conversación, sin embargo nunca se debe perder la responsabilidad de preparar las entrevistas. Se deben preparar las preguntas que se le van a hacer al usuario.

¿Cómo preparar la entrevista?

Haz un brainstorming de preguntas

Escribir las potenciales preguntas que el equipo es capaz de generar. Hay que intentar construir las preguntas pensando en las posibles respuestas del usuario para poder cubrir diferentes temas.

Identificar y ordenar los temas

Esto es muy similar a agrupar las ideas, es necesario agrupar las preguntas por temas, una vez se ha identificado los diferentes temas, hay que determinar el orden en que se van a preguntar para que la conversación fluya de la manera más natural.

Pulir las preguntas

Una vez todas las preguntas esta agrupadas por temas y ordenadas, resulta de mucha utilidad volver a revisarlas a ver si hay preguntas redundantes en alguna parte de la conversación.

4. Entrevistar para generar empatía

En la metodología de Design Thinking se busca entender los pensamientos, emociones, motivaciones de los usuarios a fin de poder determinar cómo podemos innovar para ellos.

¿Cómo hacer la entrevista?

- **Preguntar por qué,** incluso aunque el entrevistador crea que sepa la respuesta, preguntando a las personas evocamos a que el usuario diga lo que de verdad está pensando. Algunas veces las respuestas de los usuarios pueden sorprender.
- **Nunca decir "Normalmente" mientras se hace una pregunta.** En lugar de preguntar sobre el número de veces u ocurrencias, es mucho más útil formularla "*Cuéntame la última vez que ____*".
- **Fomenta las historias.** Ya sea verdad o no la historia que está contando el usuario, las historias revelan el cómo ven el mundo. Es útil hacer las preguntas de modo que evoquen a contarla a modo de historia.
- **Busca inconsistencias.** Muchas veces hay un abismo entre lo que se dice y lo que de verdad se hace. Estas inconsistencias que están muchas veces ocultas tienen un gran potencial.
- **Hay que poner atención en el lenguaje no verbal.** Hay que estar pendiente del lenguaje corporal.
- **No tenga miedo a los silencios.** Incluso aunque el usuario se pare antes de responder, no le ayude sugiriendo la respuesta. Si no se siguen esta práctica los usuarios suelen responder afirmativamente a lo que se sugirió.
- **No hacer preguntas binarias.** Las preguntas binarias se pueden responder con una sola palabra. Hay que evitar a toda costa esto dado que lo que se busca son historias.
- **Máximo 10 palabras por pregunta.** Si no el usuario seguramente se pierda.
- **Hay que hacer las preguntas de una en una.** Hay que resistir la necesidad de emboscar al usuario.
- **Hay que estar seguro de que se está preparado para capturar la información.** Las entrevistas es necesario realizarlas en parejas. Uno guiará la conversación lanzando las preguntas y el compañero se encargará de ir cogiendo notas.

5. Compartir y capturar

Se comparte con los miembros del equipo por tres motivos. El primero es para ver que han oído y visto sobre los diferentes usuarios. Aunque todos los miembros del equipo hayan estado presentes en la entrevista. En segundo lugar al compartir cada uno el punto de vista se pueden sacar información extra que en un principio no se había extraído. En tercer lugar al capturar hasta el último detalle del trabajo de campo se puede empezar el espacio por saturación

Desempaquetar todas las observaciones y todas las historias que se escucharon, se sintieron durante el trabajo de campo. Cada uno de los miembros del equipo debe compartir sus notas con el resto del equipo, para este proceso se recomienda escribir como máximo una frase por Posits. Todo estos Posits pegarlos en una pared y con ello posteriormente puede empezar el proceso de saturar y agrupar.

6. Saturar y Agrupar

El paso anterior ayuda a poner todos los pensamientos y experiencias en un medio físico y visual. El quipo debe explorar todos esos pensamientos, experiencias e historias tratando de encontrar patrones, temas, hilos conductores que permitan darle un significado a esos pensamientos para crear una solución a ellos.

¿Cómo saturar y agrupar?

Hay que poner los Posits que se han hecho en el paso anterior en una pared. Una vez este toda la pared saturada de Posits hay que tratar de agrupar esa información, poniendo un título a los diferentes grupos. La meta final es sintetizar los datos en información interesante a fin de poder diseñar buenas soluciones.

7. Crear Perfiles de Usuario

Al crear estos perfiles, también llamados **Personas**, se puede tener a un golpe de vista todas las observaciones reconocibles de cada uno de los usuarios.

A fin de crear las diferentes Personas, el equipo necesita juntar todas las observaciones que se encontraron con los usuarios. Después de esto se deben agrupar buscando puntos de unión entre usuarios, estos puntos de unión pueden ser desde hábitos, motivaciones, el puesto, etc.

En estos perfiles de usuario se recomienda poner las características principales de un usuario tales como el rol, las responsabilidades, las necesidades y los puntos débiles.

8. Punto de Vista

Un Punto de Vista, Point of View (**POV**), ayuda al diseñador a ver el mundo con los ojos del usuario, entender cuáles son sus necesidades. Un buen POV se crea respondiendo a preguntas del tipo ¿Cómo podríamos nosotros…?.

A fin de capturar la mayor parte de la información se recomienda escribir las frases de la siguiente manera.

Como [USUARIO] quiero [NECESIDAD] porque [RAZÓN DE LA NECESIDAD]

Las necesidades del usuario deberían salir de una manera muy sencilla.

¿Cómo podríamos nosotros….?

Las preguntas del tipo ¿Cómo podríamos nosotros?, "**How might we…?**" (**HMW**), son preguntas cortas que ayudan a crear un brainstorming.

9. Brainstorming

El Brainstorming, Tormenta de Ideas en castellano, es una buena manera de conseguir un montón de ideas que no habrían surgido si se sienta uno solo con un lápiz y un papel. La intención de un brainstorming es hacer uso del pensamiento colectivo del grupo, escuchando y relacionándose con los otros creando así ideas en común. Los brainstormings se pueden llevar a cabo durante todo el proceso de Design Thinking pero además, en cualquier punto del proyecto donde se necesiten ideas.

¿Cómo hacer una Tormenta de Ideas?

Las buenas practicas recomiendan establecer un periodo de tiempo en el que se están dando ideas, en ese periodo se tiene que intentar cubrir el objetivo que se quiere solucionar, después hay que juzgar el valor o no de esas ideas.

Una buena manera de empezar un brainstorming es tratando de responder a las preguntas del tipo **HMW**.

Hay dos maneras de capturar las ideas:

1. Copista: el copita escribe y pone de una manera visual, en una pared o una pizarra todas las ideas que van surgiendo. Estos es muy importante para capturar todas las ideas, siempre sin juzgar el valor o no de la idea.
2. Todos a una y cada uno de los miembros va escribiendo sus ideas, es muy importante de compartirlas verbalmente con el resto del esquipo. Es muy buena idea el utilizar Posits, para que se vayan pegando en la pared.

10. Prototipar para poner a prueba

Las pruebas de prototipos ayudan a poner aprueba diferentes aspectos del Espacio del Problema y del Espacio de la Solución. La manera fundamental de poner a prueba los prototipos es llevándolos a los usuario a su propio ambiente.

La idea de los prototipos no reside en crear una maqueta o un modelo a escala del concepto de la solución, la idea es llevar experiencias a los que los usuarios puedan reaccionar.

Poner a prueba con el usuario es una parte fundamental del Diseño Centrado en el Usuario. Hay que poner a prueba la idea con usuarios finales a fin de poder refinar la idea y también para mejorar el entendimiento que tienes sobre las personas para las que se está diseñando.

El procedimiento para poner a prueba el prototipo es el siguiente:

1. Dejar al usuario experimentar con el prototipo. Hay que poner el prototipo en manos del usuario y darle un contexto mínimo para que pueda entender que tiene que hacer.
2. Tienen que hablar de su experiencia. Mientras está utilizando el prototipo hay que preguntar qué piensa, que siente al utilizarlo.
3. Observar activamente. Mirar como utiliza el prototipo, es muy importante no corregir la manera en la que lo está utilizando.
4. Investigación posterior. Llegado el momento el prototipo se debe dejar a un lado y empieza la parte en la se interroga al usuario y se le pregunta que piensa, que mejoraría, etc.

Design Thinking tiene multitud de métodos diferentes que permiten empatizar y entender las necesidades de los usuarios de una manera mucho más profunda, además de tener muchos métodos muy útiles para conseguir ideas rompedoras e innovadoras. Una vez se tiene claro que se desea crear hay que sacarlo del mundo abstracto de las ideas y llevarlo al mundo tangible en el que vivimos y para ello es necesario diseñarlo de tal manera que pueda ser utilizable.

3.6 Principios Universales del Diseño

Dieter Rams (2012) es un diseñador industrial alemán. Rams es considerado por muchos como una figura clave del renacimiento del *Diseño Funcionalista* de finales de la década 1950 y 1960 y unos de los diseñadores más influyentes en el siglo 20. El diseñador explica su visión del diseño a través de la frase "*Weniger, aber besser*" que podría traducirse al castellano como “Menos, pero mejor”.

Rams (2012) estableció unos principios para el buen diseño:

1. *Un buen diseño debe ser discreto*. No debería llamar la atención o tener una decoración inútil. En contraposición al arte, los productos deben cumplir una finalidad.
2. *El buen diseño es estético*. Debería ser bonito cuando se mira y dar una buena impresión a primera vista.
3. *Un buen diseño es diseño lo mínimo posible*. Hay que centrarse en lo esencial, el problema que la aplicación debería solventar. Hacer de esas características muy buenas.
4. *El buen diseño hace un producto entendible*. Debería explicarse por sí solo, sin necesitar un manual.
5. *Un buen diseño hace un producto útil*. Un producto se compra para ser utilizado.
6. *El buen diseño es honesto*. No debería pretender ser algo que simplemente no es.
7. *El buen diseño es innovador*. Utilizando las posibilidades tecnológicas para crear algo mejor.
8. *Un buen diseño es atemporal*. No debe seguir toda tendencia, debe ser algo que se vea bien y sea aceptado el futuro.
9. *El buen diseño está pensando hasta el último detalle*. Se debe sentir que todo se ha cuidado. No debería haber partes que puedas quitar y tirarlas.

La razón de la mención de este diseñador es que estos principios que a primera vista son sólo útiles cuando se diseña un producto físico, a día de hoy son utilizados por los Ingenieros de Software cuando crean sus soluciones software.

Algunas empresas tecnológicas como Apple utilizan estos principios cuando diseñan sus productos ya sea sus ordenadores o sus Sistemas Operativos. Algunos de los productos más exitosos de Apple están basados en antiguos diseños de radios, altavoces o micrófonos del propio Rams. (Solórzano, 2015). Como puede verse en la figura 3.6, los parecidos son más que razonables.

Figura 3.6 *Comparativa de diseño entre Radio de Bolsillo Braun (1959) y el iPod Classic 1st Generation (2001)*

Fuente: (Solórzano, 2015)

3.7 Interacción Persona Computador

La Asociación de los Sistemas Informáticos, **Association for Computing Machinery** ACM (2009), define la Interacción Persona Computador, Human Computer Interaction (**HCI**), como una disciplina que aúna el diseño, la evaluación y la implementación de la interacción que sucede entre los sistemas informáticos y los humanos, junto con el estudio de los fenómenos que le rodean.

Según Gerard Jounghyun Kim (2015) el HCI es el estudio de cómo la computadora influye en el trabajo humano y en su actividad. El termino computadora a día de hoy incluye a los portátiles, Smartphone, sistemas embebidos, Smartwatch, etc. Además, el HCI está asociado a la disciplina del diseño, centrado en como diseñar la computadora de manera que sea fácil y agradable de utilizar. Un componente clave del diseño es la noción de usabilidad, que podemos definirla en términos de eficiencia, eficacia y satisfacción.

El HCI es un área interdisciplinar (ingeniería, psicología, ergonomía y diseño) que se ocupa de teorizar, diseñar, implementar y evaluar las diferentes maneras en las que las personas interactúan con los dispositivos informáticos. Esta disciplina ha ido cobrando importancia a lo largo de los últimos años y está presente en prácticamente todas las facetas de nuestras vidas. Ahora mismo se está centrando en cómo hacer que las interfaces sean altamente usables. Este término se refiere a que resulten fácil de usar, sean eficientes en la tarea para la que se ha diseñado, sean seguras y guíen al usuario a terminar la tarea. Si la interfaz es usable y eficiente esto se traduce en una mayor productividad y en una mayor satisfacción del usuario. A día de hoy, a parte de todas las características anteriores, nos encontramos con que además debe tener un atractivo estético

Cuando se diseña una interfaz de usuario hay que crear un medio eficaz de comunicación entre los seres humanos y el ordenador. El cómo hacer eso resulta tremendamente complejo sin embargo Theo Mandel (1997) en su libro sobre el diseño de la interfaz, acuñó tres reglas doradas:

1. Dejar el control al usuario.
2. Reducir la carga de memoria del usuario.
3. Hacer que la interfaz sea consistente.

Por otro lado Pressman (2010) explica y desarrolla estas tres reglas de diseño como siguen:

Dejar el control al usuario

Los diseñadores pueden sentirse tentados de introducir limitaciones y restricciones, simplificando así la interfaz. El resultado suele ser una interfaz fácil, pero que sea frustrante utilizar. Mandel (1997) define cierto número de principios de diseño que permiten que el usuario tenga el control:

- Definir modos de interacción de manera que no se obligue al usuario a realizar acciones innecesarias o no deseadas.

- Dar una interacción flexible. Debido a que diferentes usuarios tienen distintas preferencias para la interacción, debe darse la posibilidad de elegir.
- Permitir que la interacción del usuario sea interrumpible y también reversible.
- Facilitar la interacción a medida que aumenta la habilidad y permitir que aquella se personalice.
- Diseñar la interacción directa con objetos que aparezcan en la pantalla. El usuario tiene la sensación de control cuando puede manipular los objetos que se necesitan a fin de ejecutar un trabajo en la misma forma en la que lo haría si el objeto fuera algo físico.
- Ocultar los tecnicismos internos al usuario ocasional.

Reducir la necesidad de que el usuario memorice

Cuanto mayor sea la carga de memoria exigida al usuario mayor número de errores cometerá. Es por ello que una interfaz bien diseñada no debe sobrecargar la memoria del usuario. Mandel (1997) define los siguientes principios de diseño que permiten que una interfaz reduzca la necesidad de que el usuario memorice:

- Reducir la demanda de memoria de corto plazo. Cuando los usuarios se involucran en tareas complejas, la demanda de memoria de corto plazo es significativa. La interfaz debe diseñarse para disminuir la necesidad de recordar acciones, entradas y resultados del pasado. Esto se logra dando claves visuales que permitan al usuario reconocer acciones anteriores, en lugar de que tenga que recordarlas.
- Definir los atajos de teclado de manera que sean intuitivos, por ejemplo utilizando la primera letra de la acción.
- La distribución visual de la interfaz debe basarse en una metáfora del mundo real.
- Revelar información de manera progresiva.

Hacer consistente la información

La interfaz debe presentar la información de manera consistente, es decir, que toda la información esté organizada de acuerdo con las reglas que se respeten en el texto de pantallas desplegadas. Que los mecanismos de entrada se limiten a un par de tipos a lo largo de la aplicación y que los mecanismos para pasar de una tarea a otra se definen e implementen de manera consistente.

Diseño de una interfaz para Aplicaciones Web

Cuando se diseña un producto como una aplicación Web, los requerimientos del mismo crecen respecto al de una aplicación tradicional. Sin embargo, Dix (1999) afirmó que para diseñar una interfaz de usuario para una aplicación web se debería responder a tres preguntas fundamentales:

1. **¿Dónde estoy?** La interfaz debe informar al usuario de su localización en la jerarquía del contenido.
2. **¿Qué puedo hacer ahora?** La interfaz debe siempre ayudar al usuario a ver cuáles con sus opciones, cuáles de ellas están disponibles, que contenido es relevante, etc.

3. **¿Dónde he estado y hacia dónde voy?** La interfaz debe facilitar la navegación para ello, debe disponer de alguna solución que permite al usuario saber de dónde viene y que trayectoria puede tomar para moverse a cualquier punto de la Web App.

Cuando un usuario abre una aplicación web, la interfaz es la primera impresión que recibe de la página. En ese momento, el contenido, las capacidades que tiene la página, lo sofisticada que sea y los servicios que tenga, pasan a un segundo plano. Ya que una interfaz mal diseñada hará que el usuario se sienta decepcionado y posiblemente prefiera utilizar otra página diferente. Por tanto, la interfaz debe captar al usuario. Bruce Tognozzi (2001) define una seria de principios fundamentales de lo que debe ser una interfaz:

- Las interfaces eficaces son atractivas visualmente y perdonan los errores, lo que da a sus usuarios la sensación de tener el control. Los usuarios perciben rápidamente la totalidad de sus opciones, captan cómo lograr sus metas y cómo hacer su trabajo.
- Las interfaces eficaces no preocupan al usuario con el funcionamiento interno del sistema. El trabajo se guarda de manera cuidadosa y continua, con opción total para que el usuario deshaga cualquier actividad en cualquier momento.
- Las aplicaciones y servicios eficaces realizan un máximo de trabajo, al tiempo que requieren un mínimo de información de parte de los usuarios.

Por otra parte Pressman (2010) cuenta de boca de Tognozzi (2001) cuáles son los principios generales del diseño:

- **Previsión**. Hay que prever cuales pueden ser la acciones que va a hacer el usuario, hay que pensar cual va a ser el siguiente movimiento del usuario.
- **Comunicación**. La interfaz debe mostrar cual el estado de la tarea que ha iniciado el usuario.
- **Consistencia**. El uso de tipografía, colores, tipo de menús y estética debe ser consistente a lo largo de la aplicación
- **Autonomía controlada**. La interfaz debe facilitar el movimiento del usuario a través de la aplicación, pero lo debe hacer de manera que obligue a respetar las convenciones que se hayan establecido para la aplicación
- **Eficiencia**. El diseño de la aplicación y su interfaz deben optimizar la eficiencia del trabajo del usuario, no la del desarrollador que la diseña y construye ni del ambiente cliente-servidor que la ejecuta
- **Flexibilidad**. La interfaz debe tener flexibilidad suficiente para permitir que algunos usuarios realicen tareas directamente, y que otros exploren la aplicación en forma aleatoria
- **Centrarse**. La interfaz de la aplicación (y el contenido que presente) debe mantenerse centrada en las tareas en curso del usuario.
- **Objetos de la interfaz humana**.
- **Reducción de la latencia**. En vez de hacer que el usuario espere a que termine alguna operación interna (como descargar una imagen gráfica compleja), la aplicación debe usar tareas múltiples, de manera que permita que el usuario continúe con su trabajo mientras finaliza la operación.
- **Metáforas**. Una interfaz que use una metáfora de interacción es más fácil de aprender y de usar, en la medida en la que la metáfora sea apropiada para la

aplicación y el usuario.

- **Mantener la integridad de los productos del trabajo**. Un producto del trabajo debe guardarse en forma automática, de modo que no se pierda si ocurriera un error.
- **Legibilidad**. Toda la información presentada en la interfaz debe ser legible, es decir, debe ser accesible para grupos de personas con problemas visuales.
- **Dar seguimiento al estado**. Cuando resulte apropiado, debe darse seguimiento al estado de la interacción del usuario y guardarlo, de modo que éste pueda salir y volver más tarde para recuperarlo de donde lo haya dejado
- **Navegación visible**. Una interfaz web bien diseñada da "la ilusión de que los usuarios están en el mismo lugar, con el trabajo llevado a ellos". Cuando se emplea este enfoque, la navegación no es asunto del usuario. En vez de ello, éste recupera objetos del contenido y selecciona funciones que se despliegan y ejecutan a través de la interfaz
- **Ley de Fitt**. "El tiempo para llegar a un objetivo depende de la distancia que hay hasta él y del tamaño que tenga". Con base en un estudio realizado en la década de 1950 la Ley de Fitt "es un método eficaz para modelar movimientos rápidos e intencionados, donde un apéndice (como una mano) comienza en reposo en una posición específica de arranque y vuelve al reposo dentro de un área objetivo" Si una secuencia de selecciones o entradas estandarizadas (con muchas opciones diferentes dentro de la secuencia) es definida por una tarea de usuario, la primera selección (con el ratón, por ejemplo) debe estar físicamente cerca de la siguiente selección.

Aunque estos principios de diseño fueron propuestos en el año 2001, cuando la web como la conocemos hoy en día no existía, estos principios siguen funcionando y aunque alguno de ellos necesite de una pequeña actualización para adaptarla a los tiempos actuales, aún son perfectamente válidos. Muchos de estos principios siguen apareciendo en las últimas guías de diseño de las principales compañías de software, como puede ser Microsoft, Google o Apple.

3.8 Human Interface Guidelines

Las Human Interface Guidelines (**HIG**) son guías que se ofrecen a los desarrolladores del software y diseñadores a fin de crear una experiencia de usuario consistente, intuitiva y entendible. La mayoría de las guías se limitan a delimitar cómo será el *"look and feel"*, aspecto y comportamiento en castellano, en un entorno de desarrollo (Android, Mac OS, Windows, iOS, Linux...). Las guías están basadas en principios ya probados en la industria, a través de diferentes estudios de usabilidad, (elementary.io, 2016).

El objetivo principal de las guías es crear consistencia a lo largo de un entorno de desarrollo, incluyendo aplicaciones desarrolladas por terceros.

Algunos ejemplos muy significativos en lo que HIG se refiere son las guías de las siguientes empresas, ya que sus productos son utilizados por millones de usuarios diariamente, deben ser claras y establecer unas buenas prácticas para los desarrolladores:

3.8.1 Google Material Design Guideline

En esta guía Google (2016), intenta crear un lenguaje visual que sintetiza los principios clásicos del buen diseño junto con las últimas innovaciones y posibilidades de la tecnología. Esta guía es utilizada con desarrolladores en millones de aplicaciones ya sea en navegación web, Smartphone, Smartwatch, vehículos con sistema Android o en una Android TV.

La principal característica de Material Design es que imita a la perfección cómo sería la interfaz en un documento real como si la propia interfaz estuviera formada por capas que se solapan una sobre otras. En la figura 3.7 se puede observar esa estratificación:

Figura 3.7 *Distribución en capas de Material Design*

Fuente: (Google - Alphabet Inc. , 2016)

Material Design (Android Developer - Google Inc., 2013) es una metáfora. Esta metáfora es la unificación del espacio racional y un sistema de movimiento. Material es una realidad táctil, inspirada en la escritura clásica en papel y tinta. Las superficies, esquinas y sombras aportan la sensación de que esta las capas están estratificadas. Además, junto al uso de metáforas basadas en movimientos reales hacen que sea

rápidamente entendido por el usuario.

Los fundamentos de luz, superficie y movimiento son la llave que permite expresas como se mueven los objetos, interactúan, como existen en el espacio y como se relacionan entre ellos. Las sutiles sombras, dividen espacios e indican el movimiento de objetos por la superficie.

El movimiento da significado y refuerza al usuario a ser el que manda. Las acciones del usuario se reflejan en el diseño transformándolo. Toda la acción sucede en un solo ambiente. Los objetos son representados al usuario sin romper la continuidad de la experiencia mientras se transforma y se reorganiza. Un movimiento con significado y apropiado sirve para que se centre la atención del usuario.

Material Design es un entorno tridimensional que contiene luz, materia y sombra. En este entorno la materia tiene ancho, altura y fondo, como si de los tres ejes de coordenadas se trataran. Todo elemento tiene un pixel de grosor en el eje Z. Los elementos deben colocarse en un orden jerárquico a lo largo del eje Z como muestra la siguiente figura:

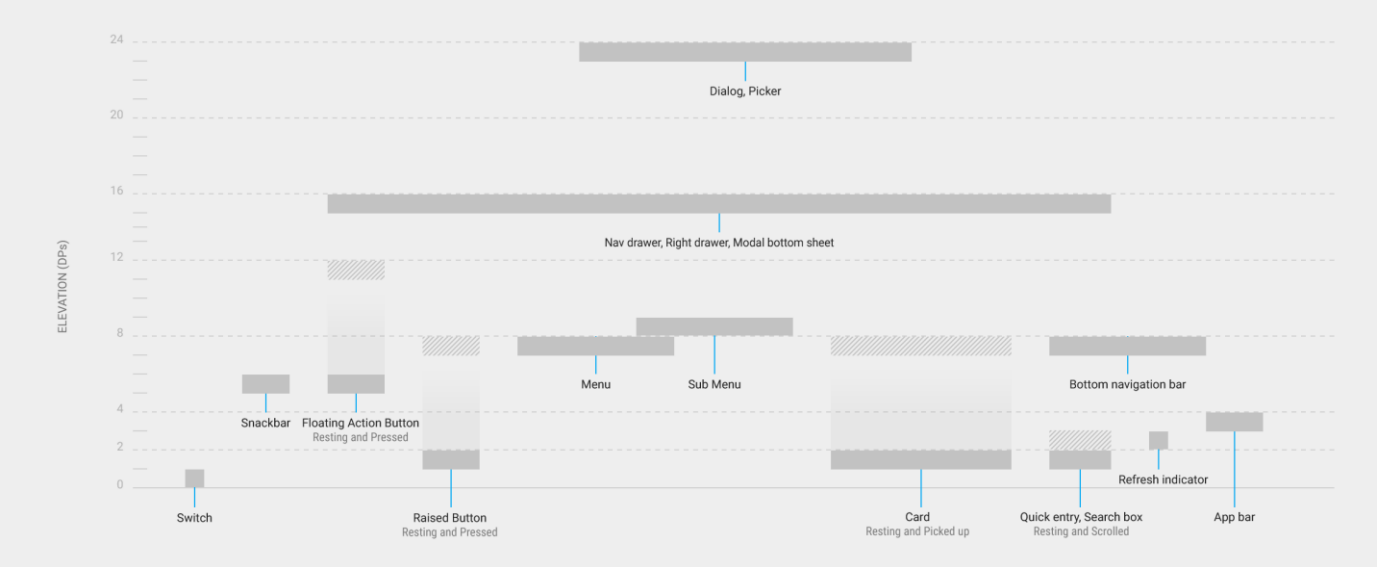


Figura 3.8 *Distribución y elevación entre los diferentes componentes de Material Design*

Fuente: (Google - Alphabet Inc. , 2016)

3.8.2 Apple Human Interface Guideline

Apple (2016) a diferencia de Google no tiene unificadas las guías de diseño entorno a una sola, cada producto macOS, iOS, watchOS y tvOS tienen sus respectiva guía de diseño centrándose cada una en las necesidades del producto.

Por su parte Apple considera que las aplicaciones de sus sistemas de deben tener esta serie de características, las cuales, según ellos, les permiten diferenciarse de la competencia:

- **Claridad**. A lo largo del sistema, el texto debe poder ser leído a cualquier tamaño, los iconos deben ser preciosos y coherentes, centrándose en afinar la funcionalidad motiva mejorar el diseño. El color, fuentes, gráficos y los elementos de interfaz permite resaltar el contenido relevante y verbalizar lo interactivo.

- **Deferente**. Un movimiento fluido y fresco, una interfaz bonita ayudan a la gente a entender e interactuar con el contenido al mismo tiempo que nunca pelean contra él. El contenido típicamente llena la totalidad de la ventana, mientras los elemento translucidos desdibujan y permite entender mucho mejor. Minimizando el uso de embellecedores, gradientes, y sombras que caen manteniendo la interfaz con luz y espaciosa, asegurándose siempre que el contenido es primordial.

- **Profundidad**. Las diferentes capas visuales y un movimiento real expresan una jerarquía, cuentan vitalidad y facilitan el entendimiento. Las transiciones dan un sentido profundo mientras navegas por el contenido.

Apple recomienda mantener estos tres principios en mente a lo largo del desarrollo de la aplicación a fin de obtener la mejor UX y para alcanzar el éxito. A parte de estos tres principios Apple habla de las siguientes características que deben tener las aplicaciones desarrolladas para sus productos:

- **Integridad Estética**, esta representa como de bien esta la apariencia y el comportamiento de una aplicación. Por ejemplo una aplicación que ayuda a las personas a realizar una tarea critica, debe hacerles centrarse utilizando lo sutil, gráficos discretos, controles estándar y comportamientos predecibles. Por otro lado, una aplicación inversiva como puede ser un juego debe capturar al usuario a través de la apariencia.
- **Consistencia**. Una aplicación consistente implementa estándares y paradigmas familiares utilizando elementos propios del sistema, iconos conocidos, fuentes comunes y una terminología uniforme y por todos conocidos. La aplicación debe incorporar características y comportamientos de la manera que espera la gente.
- **Manipulación directa**. La manipulación directa del contenido permite y facilita el entendimiento de las personas. Cuando un usuario rota su dispositivo o utiliza gestos que afectan al contenido de la pantalla, hacen que el usuario puede ver de manera inmediata el resultado de sus acciones.
- **Feedback**. La retroalimentación de las acciones muestran el resultado y mantienen a la gente informada. Los elementos interactivos están resaltados brevemente cuando son pulsados, las barras de progreso comunican al usuario el estado de las operaciones en ejecución y las animaciones y el sonido ayudan a clarificar el resultado de las acciones.
- **Metáforas.** La gente aprende más rápido cuando la aplicación virtualiza los objetos y acciones de metáforas que le resultan familiar de experiencias. Las metáforas funcionan bien porque permiten al usuario interactuar físicamente con el elemento.
- **Dejar el control al usuario.** Una aplicación debe sugerir cual es a siguiente acción o de cuáles son las consecuencias potenciales en caso de realizarla, normalmente es un error por parte del desarrollador de tomar las decisiones del usuario. Las mejores aplicaciones deben de poder encontrar un balance entre las decisiones de los usuarios y las decisiones tomadas automáticamente. Una buena aplicación debe dar la sensación de que el usuario está tomando las decisiones y todos los elementos que maneja son predecibles y familiares.

3.9 Diseño de la interfaz del usuario

Durante el proceso de diseño se pasan por diferentes fases, desde el diseño en papel hasta el diseño a través de herramientas como Photoshop. Sin embargo herramientas como **Balsamiq**, ayudan a crear rápidamente y de manera muy sencilla prototipos de las diferentes ventanas que componen un sistema.

3.9.1 Balsamiq

Este tipo de herramientas permiten crear prototipos de una manera muy sencilla y visual. Balsamiq además tiene la posibilidad de crear simulaciones de navegación entre las diferentes ventanas mediante la creación de conexiones entre las ventanas, que al pulsar sobre un botón este nos lleva a la siguiente ventana, como si ese botón tuviera algún tipo de funcionalidad, (Balsamiq, 2016).

Otra funcionalidad interesante de Balsamiq es que permite crear estudios de usabilidad, bien es cierto que con algunas limitaciones. También se puede exportar los prototipos creados en formato *.pdf* o en *.png* para poder utilizarlo en presentaciones a los diferentes Stakeholders u otros colegas de trabajo. Como se puede observar en la siguiente figura:

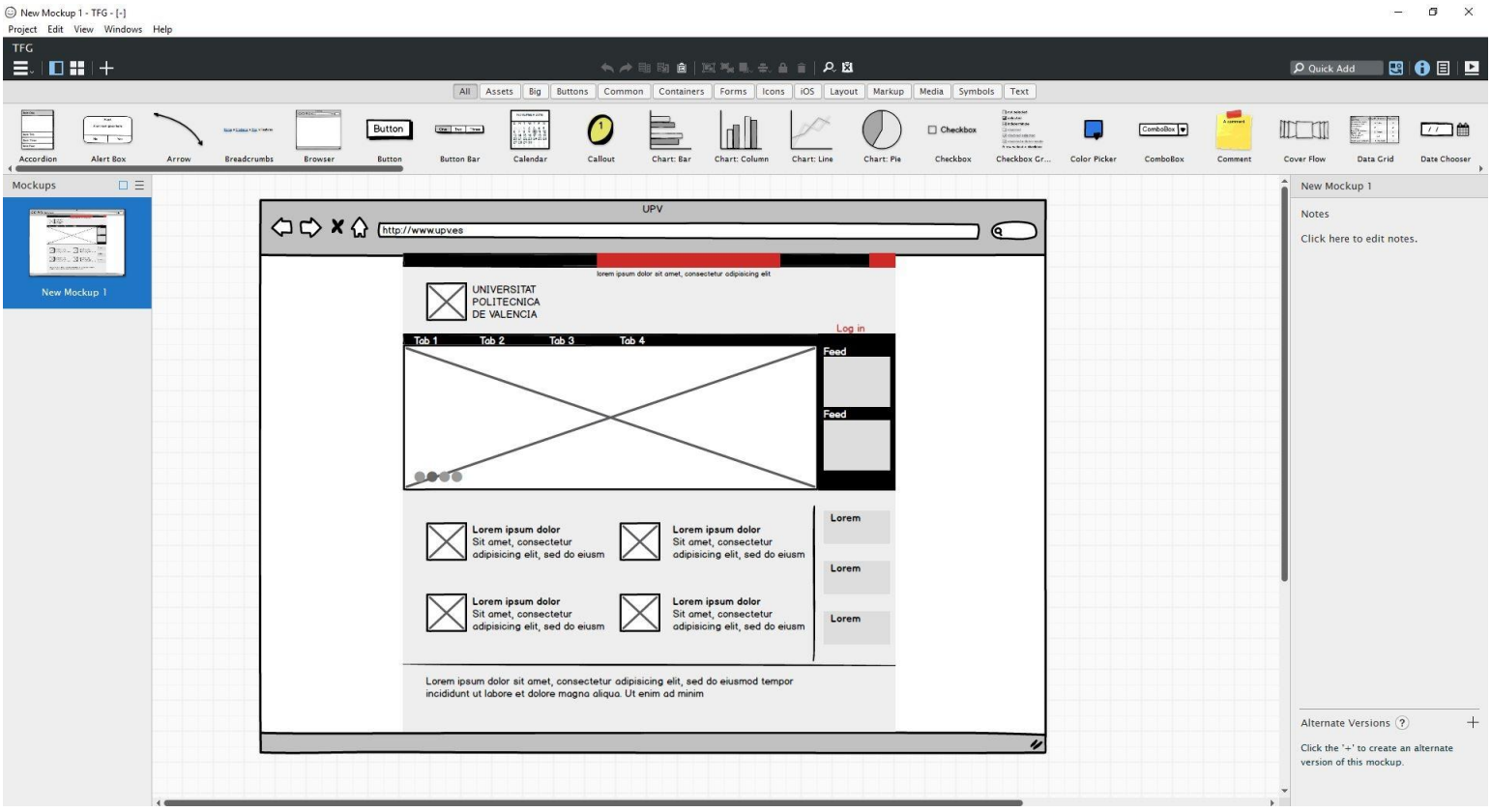


Figura 3. 9 *Interfaz de Balsamiq junto con Mockup de la pantalla de inicio de la UPV*

Fuente: Elaboración propia

4. Propuesta de mejora

4.1 Contexto del Problema

Nos situamos en Febrero del 2016, departamento de soporte de **XYZ**. En este contexto surge una idea. Viendo el pobre rendimiento y la empinada curva de aprendizaje para los nuevos trabajadores (sobre todo para nativos digitales), surge la idea de cambiar ese software por otro adaptado a las necesidades y expectativas de los usuarios actuales.

La idea inicialmente giraba en torno a un software fácil de utilizar que cumpliera las expectativas de los usuarios actuales cuyos marcos de referencia se crean en el uso diario de las aplicaciones que llevan en sus Smartphone, con una interfaz clara y versátil. Además, se buscaba un rendimiento que permitiera una rápida realización de las tareas, con muchas de las tareas más tediosas y manuales sencillamente automatizadas, evitando así tener que realizar doble trabajo. Por último, el producto software debía cumplir con los estándares de seguridad que las empresas exigen.

En el Departamento de Soporte de XYZ se da soporte a más de 86000 clientes a lo largo de 190 países. Más que dar un soporte una vez el problema haya ocurrido, se centra en dar una asistencia mucho más proactiva para lo que necesita de la creación de una relación con el cliente al objeto de alcanzar el éxito mutuo. En el Departamento de Soporte los trabajadores actúan como consultores que ofrecen y recomiendan los servicios, sugieren información y eventos que podrían hacer que el cliente alcanzara el mayor valor añadido posible de todo lo que ofrece el soporte.

Como se indicó anteriormente, el Departamento de Soporte no da un soporte estándar, sino que se trata de un soporte Premium, con un alto costo anual para el cliente por lo que el cliente debe ver el valor de este pago de dinero extra. Se podría explicar con la analogía de la primera clase en un avión, esperas un trato especial y un sitio más cómodo, además de tener servicios extra que en turista no están disponibles.

Dentro del departamento encontramos dos roles principales:

1. **Asistente de Soporte**. Este se encarga de dar soporte a los clientes con productos in situ, es decir, los clientes se encargan de mantener o externalizan los Centro de Procesamiento de Datos (**CPD**). Estos clientes suelen ser grandes multinacionales con cientos de miles de usuarios, por lo que un buen soporte es primordial.

2. **Manager del Éxito**. Este segundo rol se encarga de dar soporte a los clientes con soluciones Cloud, es decir, la propia XYZ se encarga de mantener la disponibilidad, escalabilidad y flexibilidad del CPD. Estos clientes suelen ser multinacionales mucho más pequeñas, pero al ser un producto estratégico para XYZ, está centrándose mucho en ofrecer un soporte muy proactivo y Premium.

El software con el que se da soporte actualmente, consiste en un conjunto de cuatro aplicaciones diferentes:

1. ***Front-end* del Customer Support System** (CSS). Este sistema se encarga de las relaciones con el cliente, la colaboración. El cliente abre un incidente y llega hasta el departamento a través de este sistema. El sistema sirve como la plataforma sobre la cual los Asistentes de Soporte, contestan y resuelven el incidente abierto por el cliente.
   A este sistema se entra a través del navegador como si de una página web normal se tratara. El sistema detecta que te estás conectando desde la red corporativa y permite acceder al sistema.
2. ***Back-end* del Customer Support System** (CSS). En este sistema el acceso es diferente, ya que se entra a través de otra aplicación y su uso está basado en la ejecución de transacciones. El acceso a través de transacciones es realmente potente pero requiere por parte del usuario un alto esfuerzo de memoria. En el *back-end* se tiene acceso a una vista en la que se puede consultar todos los sistemas que tiene el cliente con la empresa XYZ.
   En este sistema existe información muy relevante sobre el cliente y permite hacerse una idea muy clara de cuál es el estado del cliente, versión, nivel de paquetes de mejoras, tipo de base de datos, sistema operativo que utiliza, etc.
3. ***Front-end* del Customer Relationship Management** (CRM). Este sistema al igual que el *front-end* del CSS se entra a través del navegador. En este sistema principalmente se pueden consultar información sobre incidentes previos, podríamos decir que tiene la historia del cliente con XYZ. En el CRM se registra todo lo acontecido con el cliente, que servicios se les han reservado, sobre qué sistema, la razón de la reserva (actualización, problema de rendimiento, investigación de configuración de los parámetros del sistema…).
   Además de servir para comprobar la conexión de los sistemas del cliente a los sistemas internos. Sin esas conexiones habilitadas no se puede resolver ningún problema al cliente.
4. ***Back-end* del Customer Relationship Management** (CRM-ICP). En este sistema se entra a través de transacciones. Principalmente es utilizado para generar reportes sobre el rendimiento de los sistemas. Estos reportes generados automáticamente son extensos documentos de unas 60 páginas, que contiene información sobre el rendimiento de un sistema:

   > *Tiempo medio de respuesta, tiempo de carga de una vista, crecimiento de la base de datos, tamaño de la base de datos, tiempo medio de ejecución de las transacciones, carga de trabajo del sistema, número de usuarios activos, entre otros muchos parámetros igualmente importantes.*

   La información que incluye estos reportes es altamente importante, pero al ser reportes de alrededor de 60 páginas no se pueden leer rápidamente. Surgen preguntas, ¿Se puede mostrar la información de otra manera?
   Estos reportes además incluyen información sobre posibles soluciones a los problemas de rendimiento del sistema, al incluir referencia notas técnicas.

Uno de los principales problemas y foco de muchas de las quejas de los usuarios del departamento de soporte es que estos cuatro sistemas están aislados, sin comunicación entre ellos, por lo que muchas veces hay que realizar el mismo trabajo en varios de los sistemas a fin de documentar todo correctamente.

Otro problema del sistema está en que es un **software heredado** al cual se le han ido añadiendo funcionalidades a fin de poder cumplir con los requerimientos del departamento. Sin embargo, estos sistemas jamás fueron pensados para realizar muchas de estas funcionalidades, por lo que ni cumplen las expectativas ni el rendimiento es el esperado.

Uno de los principales retos al que nos enfrentamos desde el comienzo del proyecto es que el software es utilizado diariamente por más de 8000 empleados a lo largo de 54 países. Por lo que la transición de un software a otro debía crear el menor número de problemas posibles.

La **reticencia al cambio**, siempre que la gente escucha que el sistema con el que lleva trabajando años va a cambiar reacciona con cierta oposición. Aquí entran en juego uno de los problemas que muchas empresas empiezan a notar y es que para 2025 el 70% (PwC, 2016) de la fuerza de trabajo estará formada por los llamados nativos digitales, usuarios que esperan tener el mismo rendimiento y sencillez de uso que tienen al utilizar su Smartphone, en un software empresarial. Estos nativos digitales chocan de frente con la posible reticencia que tienen las generaciones anteriores a un cambio brusco en un sistema.

Los cuatro sistemas tienen en común una interfaz altamente compleja, que requiere de bastante entrenamiento para poder utilizarla, además de necesitar de mucha memoria a corto plazo, toda la información disponible es mostrada en una sola pantalla. Por lo que muchas veces el usuario se siente sobresaturado de información y no tiene muy claro que puede y que no debe hacer. Además de no ser una aplicación que esté basada en roles, por lo que todo los usuarios pueden editar la información aunque no debiera poder cambiarla.

Otro de los retos al que nos enfrentábamos es que el nuevo sistema debía tener en cuenta a un producto recientemente adquirido por la empresa, especializado en gestión de Recurso Humanos, llamado **Factores del Triunfo** a partir de ahora. Este nuevo producto había que empezar a integrarlo con el resto de la compañía, ya que hasta la fecha estaba utilizando su propio sistema de soporte. Por lo que durante todo el proyecto hemos tenido en cuenta que no solo debíamos tener en mente a un solo departamento sino dos: dos roles distintos, necesidades muy parecidas, pero no iguales.

Finalmente, uno de los retos más difícil al que siempre se enfrenta cualquier proyecto es el aspecto económico. El proyecto necesita de expertos, desarrolladores, diseñadores y el soporte de gente que este dentro del consejo de dirección de la empresa. ¡Hay que buscar un sponsors!

4.2 Primera toma de contacto

Tras las primeras reuniones del equipo se decide que la mejor manera de empezar a entender cuáles son los principales problemas que tienen los Asistentes de Soporte con los sistemas es realizar una encuesta preliminar. Esta encuesta cuenta con más de 30 preguntas abiertas, algunas de la cuales son:

- ¿Cómo prepara la estrategia y como empieza la relación con el cliente?
- ¿Qué información necesita buscar?
- ¿Con que otros roles dentro de la organización necesita contactar?
- ¿Qué información le proporcionan?
- ¿Qué información necesita cuando un compañero está fuera de la oficina y debe cubrirle con un cliente?
- ¿Cómo hace para distribuir el esfuerzo entre todos sus clientes?
- ¿Cómo crea una relación exitosa con el cliente?
- ¿Cómo sabe el cliente lo que tiene planeado para ellos?
- ¿Cómo hace un seguimiento del plan?
- ¿Crea metas/objetivos para el cliente?
- ¿Cómo comparte la información con el cliente?
- ¿Qué parte de tu trabajo debería ver siempre el cliente?
- ¿Considera aceptable el rendimiento del sistema?

Estas y otras muchas más preguntas fueron enviadas a más de 60 personas a lo largo de las cuatro regiones geográficas, *EMEA, APJ, LA* y *NA*. La encuesta trataba de ser lo más abierta posible e intentaba cubrir todos los roles de la organización.

Tras analizar la encuesta se encontraron información muy valiosa que confirmaba muchas de las sospechas que tenía el equipo:

- Pérdida de información. Al no estar ni centralizados ni interconectados muchas veces los trabajadores solo incluían la información en uno de los sistemas.
- Pérdida de información a través del correo electrónico. Muchos trabajadores confirmaban que utilizaban el correo electrónico como medio de comunicación con el cliente. Al no utilizar el sistema CSS, se pierde mucha información al no poder tener acceso al correo personal del trabajador.
- Utilización de herramientas externas, como Notepad u OneNote para guardar información del cliente.
- Gran dificultad para encontrar a la persona adecuada, ajena al departamento que pudiera dar información relevante del cliente.
- Muy difícil encontrar información sobre un cliente cuando se tiene que sustituir a un compañero, al estar muchas veces sólo en su ordenador.
- Rendimiento muy pobre, con un alto tiempo de carga durante las horas puntas de trabajo y perdida del progreso del trabajo al colgarse el sistema.
- Utilización de documentos Word o Excel para compartir información con el cliente. Algunos encuestados afirmaban que daba mala imagen al ser un simple documento que todo el mundo puede hacer.
- Dificultad de mostrar al cliente los éxitos conseguidos.

- Dificultad al tratar de encontrar información relevante sobre el cliente al tener que buscar en muchos sistemas diferentes.
- Interfaz con demasiadas funcionalidades, muchas veces frustrante y compleja.
- La gran mayoría de los encuestados respondieron positivamente a que establecieran objetivos/metas. Sin embargo, en donde diferían era en cómo se establecían esas metas, diferentes formatos, diferentes herramientas, etc.
- También se encontraron diferencias en la manera de trabajar entre las diferentes regiones.
- Falta de plantillas para comunicarse con el cliente.
- Falta de un una infraestructura donde establecer los objetivos y metas del cliente, esto implica mucho trabajo manual.

Además de toda la información anteriormente mencionada, también se identificaron diferentes necesidades, metas y objetivos que había que conseguir en el proyecto.

Una vez la encuesta estaba perfectamente estudiada se tenían suficientes argumentos como para obtener más recursos materiales y humanos. Con lo que la alta dirección del Departamento de Soporte dotó de suficientes fondos para continuar con el proyecto.

4.2.2 Herramientas Existentes

Una vez que la encuesta estaba terminada y estudiada se dispuso a estudiar que herramientas ya existentes dentro de la compañía podrían funcionar para el Departamento de Soporte.

Para ello se creó un Excel que contenía cinco secciones:

- Usabilidad
- Visibilidad
- Reportes
- Colaboración
- Personalización

La intención con este Excel se basaba en puntuar numéricamente cada una de las posibles candidatas, a lo largo de estas cinco categorías. Cada una de las cinco categorías tenía, dependiendo de su importancia, diferente peso sobre el conteo total de puntos. Sobre cada categoría se introdujo una nota de corte que para superarla había que obtener el 50% de los puntos.

Se estudiaron a conciencia 12 herramientas, puntuándolas siempre con objetividad. Tras analizar las herramientas candidatas, prácticamente todas las herramientas suspendieron en el conteo total, y como dato curioso las herramientas actuales que se estaban utilizando, sacaron una nota menor que la mayoría de las herramientas.

Dado que tras estudiar que las herramientas ya existentes no cumplían con las necesidades había que desarrollar una que si lo hiciera. Este punto se cubrirá posteriormente.

4.3 Design Thinking

Dado que ya se tenía constancia de que el problema con la herramienta existía, tras un tiempo de deliberación interna del equipo, se decidió que había que proseguir entendiendo las necesidades del Asistente de Soporte y del Manager del Éxito. Una vez extraídas las necesidades, de ahí surgirían los requerimientos de la nueva herramienta.

La metodología escogida para este proceso fue el Design Thinking. La razón de porqué se eligió esta metodología y no otra reside en que uno de los fundadores de la compañía es uno de los mayores defensores de esta metodología. Teniendo incluso una escuela en EEUU que se centra en formar y aplicar esta metodología en escenarios reales.

La metodología de Design Thinking como se explicó en el capítulo 3.6, consta de tres fases:

1. Espacio del Problema
2. Espacio del Diseño
3. Espacio de la Solución

4.3.1 Espacio del Problema

En esta fase se centra en entender cuál es problema que se intenta resolver. Para ello se definieron unos perfiles, llamados Personas, que definían cuales son las necesidades del cada uno de los roles que utilizan de manera directa o indirecta las herramientas.

Una vez los perfiles estaban claros y definidos, se creó lo que llamamos *la historia* de esa Persona con el sistema. Una vez los pasos estaban establecidos, se realizó un ejercicio que se conoce como *Punto de Vista*. Una vez los puntos de vista están creados, los requerimientos de la aplicación son establecidos de una manera mucho más sencilla.

Para comprobar la integridad de las Personas de las Historias se hicieron muchos puntos de control, entrevistando a los usuarios finales, tratando de validar la información.

4.3.1.1 Personas

En las personas se intenta crear una carta de presentación del rol, con sus responsabilidades, necesidades, objetivos, motivaciones, actitudes y problemas.

Esta carta de presentación se realiza a fin de entender mejor y empatizar con esa persona. Cuenta cómo trabaja, qué necesita para obtener el éxito y cuáles son sus quebraderos de cabeza. Para realizar la primera versión de este documento se recurrió a la descripción del rol que aparece en el contrato y para los problemas se extrajeron de la encuesta preliminar.

Se identificaron cinco Personas diferentes que utilizaban de manera directa o indirecta el sistema, los llamamos:

1. Amy, la Asistente de Soporte y Manager del Éxito
2. Marc, el Manager

3. Tara, la Líder de Equipo
4. Frank, el Líder Funcional
5. Selina, la Senior Líder

Cada uno de estas personas tiene diferentes necesidades e interactúan con el sistema de maneras diferentes. Por ejemplo, los Managers buscan más datos de productividad mientras que Amy o Tara son las que de verdad interactúan con el cliente a través del sistema.

Amy - La Asistente de Soporte

Responsabilidades:

- Planear y llevar a cabo modelos hechos a medida de los objetivos del cliente
- Crear y mantener una excelente relación de entendimiento mutuo con el cliente
- Comunicarse de manera clara, guiando al cliente a lo largo de todos los recursos que el Departamento de Soporte ofrece
- Contribuir al éxito del cliente a través de los servicios y proporcionando los recursos adecuados para ello.
- Mantener actualizado y ser transparente con los clientes

Objetivos/Metas:

- Tener una relación excelente con el cliente, basada en una relación de respeto y confianza mutua
- Crear valor para el cliente, estableciendo junto con el los objetivo
- Demostrar el valor creado a mis clientes y a mi manager
- Colaborar de manera efectiva con todos los grupos de interés
- Crear una transparencia total de la relación con el cliente

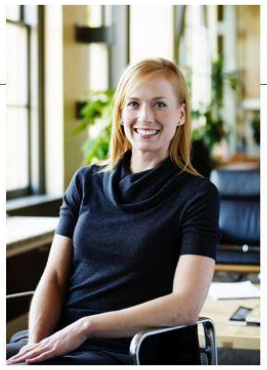
Necesidades:

- Necesita desarrollar empatía a través del entendimiento de la situación del cliente, su estrategia y objetivos
- Necesita información relevante sobre el cliente idealmente centralizado en un sistema
- Necesita estar actualizada de las últimas novedades en los productos de XYZ
- Necesita tener Feedback por parte del cliente a fin de poder demostrar el éxito con los clientes

Problemas, “Dolores de Cabeza”:

- Falta de un proceso estandarizado, con plantillas y mejores prácticas
- Gastando mucho esfuerzo en tareas sin valor como puede ser documentar
- Mala utilización de mi rol por parte del cliente
- Dependencia en factores externos para satisfacer las necesidades del cliente

- Influencia limitada sobre el cliente, actualmente las recomendaciones las debe implementar el cliente



Amy
The Advisor

"I want to make my customer successful, clearly show the value of XYZ support and be an advocate for my customer"

About

38, married, 5 years of experience in XYZ Solution Support
I'm "the face" of XYZ customer support, striving for excellent relationships with my customers being recognized as their trusted advisory link for all support related topics.
Working mostly in the office or at home with regular remote and occasional onsite meetings with customer representatives.

Works With

CONFIDENTIAL

Job Responsibilities

- Plan and execute tailored goal-based engagement plans for my customers and translate customers' requirements to XYZ
- Create and maintain an excellent relationship and reciprocal understanding with my customers
- Communicate clearly by empowering and guiding the customer to the correct resources available to them in XYZ
- Drive value realization and help safeguard retention and revenue
- Contribute to customer success with functional expertise and facilitate adequate resources
- Provide up-to-date overview and transparency about my customer engagements

Needs

- I need to develop empathy and a thorough understanding of my customers' situation, incl. strategy, goals and milestones
- I need relevant, comprehensive information about my customers, ideally from one central data source
- I need up to date information about XYZ solutions & products, ideally from one central data source
- I need effective and efficient processes and tools in place to ensure my operational excellence and mitigate vital fundamental pillars break down to avoid damage for my customer and XYZ
- I need timely feedback from my customers and senior management to demonstrate my success and the health of the customer

Goals

- Have an excellent relationship with my customers, based on mutual trust and personal respect
- Create value for my customers by driving commonly agreed goals, and providing them with timely communication
- Demonstrate created value to my customers and my management
- Collaborate effectively with all stakeholders and share adequate resources with them
- Create full transparency about my engagements

Pain Points

- No current structured processes, standards and templates for best practices
- Spending too much effort on non-value adding tasks, e.g. documenting efforts
- Being misused by customers for incident chasing due to XYZ s internal inability to adhere to SLAs
- Reliance on other internal factors to satisfy the customer needs that are out of the Advisors control
- Limited influence that my customers actually implement recommendations after giving them instructions and advice

Figura 4. 1 *Carta de presentación de la Persona Amy, la Asistente de Soporte y Manager del Éxito*

Fuente: Elaboración Propia

En el **Anexo 2 - Cartas de Presentación** se pueden encontrar el resto de las cartas de presentación del resto de usuarios que se analizaron.

Si bien al resto de Personas se le creó su carta de presentación. Llegado el momento de realizar la *Historia del Usuario* y dado que el resto de perfiles no utilizan la aplicación como tal, sino que utilizan los **KPI** (Indicadores Clave de Rendimiento) extraídos de las herramientas para tomar decisiones de dirección, no se les creo ninguna Historia del Usuario.

Una vez las cartas de presentación estaban creadas y debido a límites en los recursos humanos el equipo, se decidió que la mejor opción era realizar un primer prototipo con una funcionalidad muy concreta, que posteriormente se explicará, solamente dirigida al perfil del usuario (Persona) de Amy. La idea es que una vez creado el primer prototipo se tendrían suficientes argumentos como para conseguir más recursos humanos y materiales para proseguir con el resto de perfiles.

4.3.1.2 Historia del Usuario

Estas Historias del Usuario definen los pasos que realiza ese usuario para completar una tarea.

La historia se creó basándose en los pasos que ya se realizaban y poco a poco cambiada y validada en un proceso iterativo que duro casi un 4 semanas.

Validación

En cuanto a la validación, fue un proceso largo que se centraba en entender cuáles son las necesidades de ese usuario y por qué realiza esa tarea.

Lo más importante en este paso es entender por qué se realiza esta tarea de esa manera. Muchas veces los usuarios realizan una tarea sin saber por qué razón la está realizando, sencillamente siempre ha sido así.

Storyboard de Amy

Esta Storyboard cuenta una historia, la Historia de Amy, divida en pasos desde que recibe un cliente hasta que le muestra el valor creado al cliente. Estos pasos representan todas tareas que debe realizar Amy con su nuevo cliente.

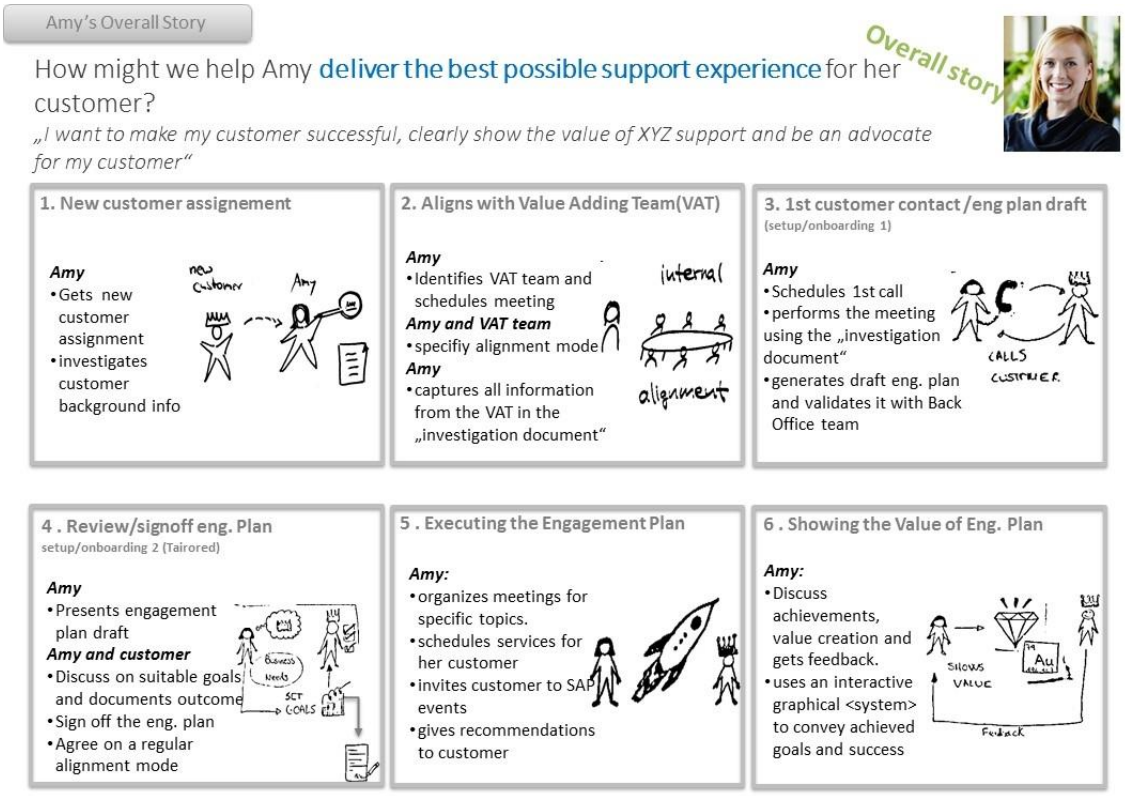


Figura 4.2 *Storyboard la Persona Amy, la Asistente de Soporte y Manager del Éxito*

Fuente: Elaboración Propia

A continuación se explicará cada uno de los pasos que se validaron con los usuarios:

1. Un nuevo cliente es asignado. Amy recibe un nuevo cliente e investiga los sistemas del cliente.
2. Amy se alinea con los diferentes Stakeholders, e identifica a lo más relevantes.
3. Amy realiza la primera llamada con el cliente y genera un borrador del plan de ejecución
4. El Plan de Ejecución es revisado y firmado por el cliente.
5. Amy lleva a cabo el Plan de Ejecución
6. Amy muestra el valor creado sobre el cliente, analiza junto al cliente los éxitos que han alcanzado y se cierra la relación.

En el **Anexo 1 - Storyboard Amy** se pueden encontrar todos los pasos con los diferentes sub-apartados al detalle.

Solamente para aclarar el concepto, un Plan de Ejecución es:

- Documento creado por Amy que incluye todos los servicios que se van a reservar para el cliente y qué cursos y formación debe realizar para estar lo mejor preparado. Se puede decir que es un mapa que establece qué se va a hacer y cuándo se va a hacer a fin de alcanzar el éxito.
- Contiene qué servicios deben ser reservados en función de los proyectos que tiene el cliente. Realmente es un elemento crítico para tener una relación exitosa con el cliente

Todos estos pasos fueron validados con los Asistentes de Soporte. Al estar en el mismo Departamento que los Asistentes de Soporte la validación resultaba familiar, informal y relativamente fácil.

Una vez todos los pasos estaban definidos había llegado el momento de ponerse "las gafas de Amy" para ver el mundo a través de sus ojos y tratar de definir sus **POV** (Puntos de Vista en castellano).

4.3.1.3 POV (Point of View)

Para realizar esta tarea y ayudar a empatizar con Amy hay que tratar de responder a preguntas del tipo **HMW** (How might we…?, ¿Cómo podríamos nosotros…?), es decir, se coge cada uno de los pasos que forman parte de la historia el usuario y se formulan preguntas de esta manera:

- ¿Cómo podríamos ayudar a Amy a recolectar información relevante del cliente?
- ¿Cómo podríamos ayudar a Amy a recolectar la última información sobre los sistemas del cliente?
- ¿Cómo podríamos ayudar a Amy a preparar eficientemente las llamadas con el cliente?
- ¿Cómo podríamos ayudar a Amy a entender la estrategia y objetivos del cliente?
- ¿Cómo podríamos ayudar a Amy a posicionar servicios estratégicos basados en el *plan de ejecución*?
- ¿Cómo podríamos ayudar a Amy a documentar las llamadas con el cliente?

- ¿Cómo podríamos ayudar a Amy a reducir el esfuerzo de documentación con el cliente?
- ¿Cómo podríamos ayudar a Amy a trabajar de manera efectiva junto con el cliente?
- ¿Cómo podríamos ayudar a Amy a guiar al cliente sobre la basta oferta de soporte?
- ¿Cómo podríamos ayudar a Amy a convencer al cliente para implementar las recomendaciones?
- ¿Cómo podríamos ayudar a Amy a demostrar el valor creado?
- ¿Cómo podríamos ayudar a Amy a que sea reconocida como un valor por parte del cliente?
- ¿Cómo podríamos ayudar a Amy para que el cliente crea en el valor del soporte de XYZ?

Con todas esas preguntas y teniendo siempre en la cabeza el Storyboard de Amy, se debería ser capaz de establecer los requerimientos de la herramienta para esa Persona.

Esta es una tarea que requiere de mucha empatía con esa Persona, puesto que hay que tratar de responder como ella respondería a esa pregunta. Para ello las buenas prácticas del Design Thinking recomiendan responder a esas preguntas de la siguiente manera:

Como Amy, quiero….

Hay que responder a todas esas preguntas cubriendo cada uno de los pasos que hay en la historia, posteriormente como en todo proceso de Design Thinking hay que validar lo generado con los usuario finales.

Un punto muy importante a tener en cuenta en esta parte del proceso es que hay que separar las posibles soluciones de la propia descripción del problema, es decir, lo que el sistema debe proporcionar al usuario (Amy) basado en sus necesidades. Se identificaron 65 requerimientos a partir del Storyboard de Amy. Tras estudiarlos se pudo observar que estos pertenecen a cinco grandes grupos de o funciones (Notificación, Documento de Investigación, Colaboración, Plan de ejecución y Feedback).

Dado que la lista de requerimientos es demasiado extensa como para mostrarla entera. Solamente se va a resaltar los requerimientos pertenecientes a la función Documento de Investigación:

- Como Amy, quiero revisar y entender cuáles son las acciones requeridas de esta nueva tarea, para así poder organizar de manera eficiente mi trabajo
- Como Amy, quiero revisar y editar el Documento de Investigación inicial, para así poder preparar la reunión con lo Stakeholders
- Como Amy, quiero editar fácilmente el Documento de Investigación, y así poder incorporar toda la información que he capturado en la reunión con el VAT
- Como Amy, quiero poder determinar cuando los cambios que he realizado en el Documento de Investigación son visibles, y así poder controlar que la información es veraz

- Como Amy, quiero saber quién cambia o actualiza el Documento de Investigación, y así poder evitar fallas en la información
- Como Amy, quiero poder revisar el Documento de Investigación antes de la primera llamada con el cliente , para estar lo mejor preparada
- Como Amy, quiero poder resaltar cualquier información que no esté disponible en el Documento de Investigación, y así asegurarme que esa información es preguntada en la llamada
- Como Amy, quiero poder usar el Documento de Investigación para liderar la llamada, y así poder compartir y validar toda la información relevante con mi cliente y capturar la nueva información
- Como Amy, quiero poder mantener notas y actualizaciones en el propio Documento de Investigación, y así poder documentar toda la información de la llamada de una manera más sencilla
- Como Amy, quiero poder cambiar entre una vista interna y externa de Documento de Investigación, y así poder compartir la pantalla con el cliente
- Como Amy, quiero poder terminar de completar el Documento de Investigación junto con el cliente, y así tener suficiente información para crear el Plan de Ejecución

Todas estas frases representan las acciones que desearía realizar Amy en lo referente al Documento de Investigación. Todos estos requerimientos pertenecen a diferentes pasos del Storyboard de Amy.

En el **Anexo 3 - Requerimientos** se pueden encontrar al detalle cada uno de los 65 requerimientos identificados en el Espacio del Problema.

Una vez que los requerimientos están completamente establecidos, el Espacio del Problema se puede considerar por terminado. Puesto que el problema está completamente planteado y delimitado.

Los 65 requerimientos que se identificaron responden a las necesidades de la Persona Amy, desde que recibe un cliente hasta que cierra la relación con ese cliente. Excluyendo de esta lista el punto 5, *Llevar a cabo el Plan de Ejecución*, por razones de confidencialidad, la empresa XYZ ha decidido que no debía compartir qué servicios y qué posibles escenarios cubren el Plan de Ejecución.

4.3.2 Espacio de la Solución

Una vez que todos los requerimientos del sistema estaban claros, es decir, el Espacio de Problema se había completado y el problema al que nos enfrentamos está perfectamente delimitado y estructurado, llega el momento de saltar al espacio de la Solución o cómo se podría dar solución a ese problema.

Las buenas prácticas del Design Thinking establecen que ha llegado el momento de buscar Diseños, es decir, cómo se puede solucionar un problema concreto. Esta solución puede ser un botón, o puede ser una tabla que muestre información. El límite es la propia imaginación en esta parte del proyecto. En esta fase ayuda mucho si se crean muchos diseños y se falla muchas veces quedándose naturalmente con las partes buenas, iterando con los usuarios y tratando de encontrar la solución correcta.

Debido a los recursos limitados se tomó la decisión de que se debía elegir solamente una de las cinco funciones que se habían identificado de los requerimientos, a fin de poder mostrar el primer prototipo a finales del 2016, principios del 2017, pudiendo conseguir un mayor importe económico para continuar el proyecto al estar recién empezado el año fiscal. Las cinco funcionalidades eran:

- Notificación
- Documento de Investigación
- Colaboración
- Plan de ejecución
- Feedback

Para tomar la decisión de cual fuera la solución, primero se preguntó a quién iba a utilizar la aplicación, los usuarios finales. Se cogió una pequeña muestra de usuarios y todos estuvieron de acuerdo en que la funcionalidad que mejor les vendría en un primer momento seria tener el "***Documento de Investigación***". Durante la entrevista se trató de sonsacar esa funcionalidad de manera indirecta. Todos confirmaron que encontrar toda la información relevante del cliente les llevaba muchísimas horas de trabajo siendo ésta además una información crítica para el éxito de la relación.

Este documento debía contener toda la información relevante del cliente, las Partes Interesadas o Stakeholders, los sistemas del cliente, los proyectos, los incidentes y la historia del cliente con XYZ. Además, debía mostrar la información en un formato sencillo. Este documento debería servir para que Amy empezara a conocer al cliente que le ha sido asignado y estuviera lo mejor preparada para la primera llamada con el cliente.

Llegado a este punto del proyecto había que buscar una herramienta que pudiera cumplir con los requerimientos de esta funcionalidad. Tras hablar con expertos en la materia de Alemania, nos mencionaron que tenían un producto bastante novedoso, llamado **Knowledge Workspace** (Innovation Center Network, 2016), basado en Polymer que tenía mucho potencial a día de hoy de convertirse en un producto puntero. Por temas de confidencialidad no puedo entrar en cómo está estructurado a nivel interno pero lo que sí puedo mencionar es que estaría desplegado en la nube como un Software as a Service (**SAAS**), utilizando la infraestructura interna que tiene la propia compañía en lo que a

servicios de la nube se refiere, así como utilizar una novedosa Base de Datos en memoria para tener un acceso muchísimo más rápido.

Knowledge Workspace, está diseñado para funcionar con cualquier tipo de aplicación, utilizando interfaces de datos, utilizando un espacio de trabajo, las herramientas colaborativas y sistemas de ticketing. Knowledge Workspace combina todas esas características de la manera más sencilla posible. Permite traer la información correcta, junto con la gente correcta, junto con una herramienta de análisis interactiva, colaborativa y con toma de decisiones.

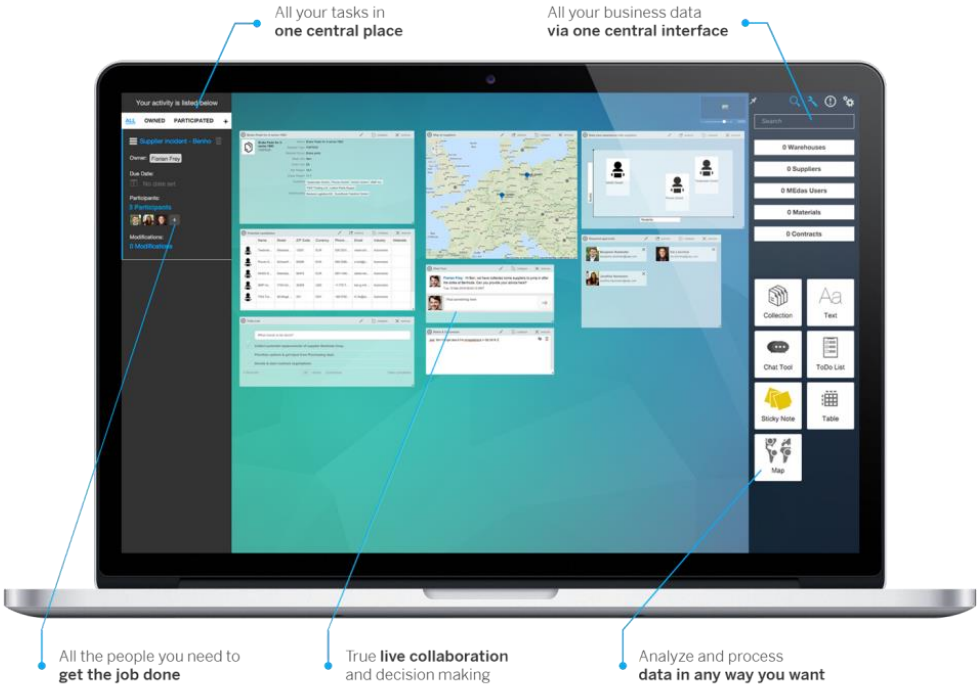


Figura 4.2 *Knowledge Workspace ejemplo*

Fuente: (Innovation Center Network, 2016)

Por otro lado **Polymer** (2016) es un framework de desarrollo creado por Google para el desarrollo aplicaciones Web. Sin embargo dado que no formo parte del equipo de desarrollo, desconozco los detalles técnicos del desarrollo.

Polymer está basado en el estándar de diseño que tiene ahora mismo Google, **Material Design** (2013), por lo que las propuestas de solución debían estar basadas en ese estilo de diseño. Sin embargo, pasaron varias semanas hasta que el equipo de Alemania confirmó que Knowledge Workspace estaba basada en Material Design, por lo que se realizaron varias propuestas de diseño basadas en las guías de diseño ya existentes en la propia compañía.

Además se utilizaron algunos consejos que tiene Apple (2016) en su propia página de desarrolladores. Los consejos principalmente están relacionados con tamaño de texto, como formatearlo, como contrastar alguna información, la importancia del espacio en blanco, la importancia de una buena organización de los controles, etc.

4.3.2.1 Estructura del Documento de Investigación

Lo primero que había que identificar era qué datos debe mostrarse en un Documento de Investigación. Una vez estuviera estructurada, ya podíamos diseñar las interfaces que solucionarían ese problema. Tras entrevistar a varios usuarios, identificamos siete grandes grupos:

1. Información Básica del Cliente
2. Indicadores del Cliente
3. Stakeholders
4. Proyectos
5. Historia
6. Sistemas
7. Incidentes

A estos siete grupos añadimos una extra, Notas, la cual se estableció en la lista de requerimientos como necesaria para contener información variada sobre el cliente.

Para organizar los datos se creó una estructura en árbol de la información que iría dentro de cada uno de esos ocho grupos.

El documento de Investigación debe contener toda la información relevante sobre el cliente a un golpe de vista.

Se puede observar en el Anexo 4 – Wireframes, la estructura del propio Documento de Investigación, así como los datos que hay dentro de cada pestaña. Sin embargo por razones de confidencialidad, las pestañas de Proyectos, Historia e Incidentes no han podido ser mostradas.

4.3.2.2 Wireframes

Los Wireframes sirven de esquemas del diseño y de la ordenación de los diferentes elementos de la interfaz.

La principal idea de los Wireframes es realizar muchos diferentes tratando de encontrar que diseño se ajusta mejor a las necesidades del usuario. Como indican las buenas prácticas del Design Thinking es mejor fallar a una etapa temprana del proyecto.

Para llevar a cabo los Wireframes en un primer momento se realizaron los primeros modelos en papel, se buscaba diseños rápidos y que fueran fácilmente desechables. Los primeros diseños es mejor realizarlos en papel puesto que si los usuarios ven un modelo con un diseño realmente trabajado, los usuarios piensan que ese diseño está grabado en la piedra y así va a ser la versión final

Un ejemplo de los primeros Wireframes que se realizaron:

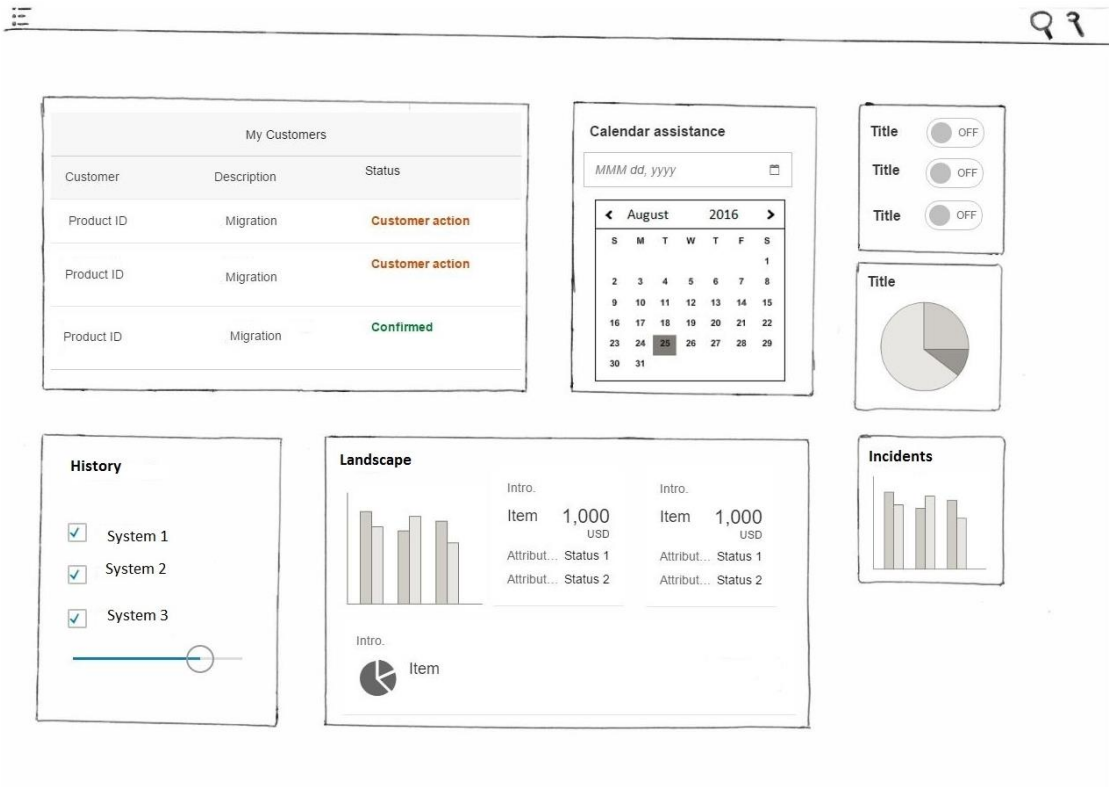


Figura 4.3 *Ejemplo primer Wireframe basado en Knowledge Workspace*

Fuente: Elaboración propia

Como se pueden observar, estos primeros diseños están muy alejados, de ser un modelo funcional, al igual que prácticamente no contienen información. Estos se crearon con la intención de empezar a validar la información que debía aparecer, como puede ser los Sistemas del Cliente (Landscape), o los diferentes Incidentes del cliente.

Conforme se va validando la información, los diseños pueden ir mejorando en lo que a aspecto visual. Para ello se utilizó la herramienta **Balsamiq**, la cual permite a través de una interfaz muy sencilla la creación de potentes Wireframes.

En la segunda iteración de diseño se trataba de encontrar como se puede organizar la información, es decir, una fila de pestañas, o una columna de pestañas, u organizarla en baldosas al puro estilo Windows 10. En este momento las posibilidades de diseño son prácticamente infinitas. Solamente hay que tratar la que más se adecua al formato que está acostumbrado los usuarios. Algunos ejemplos de estos diseños son:

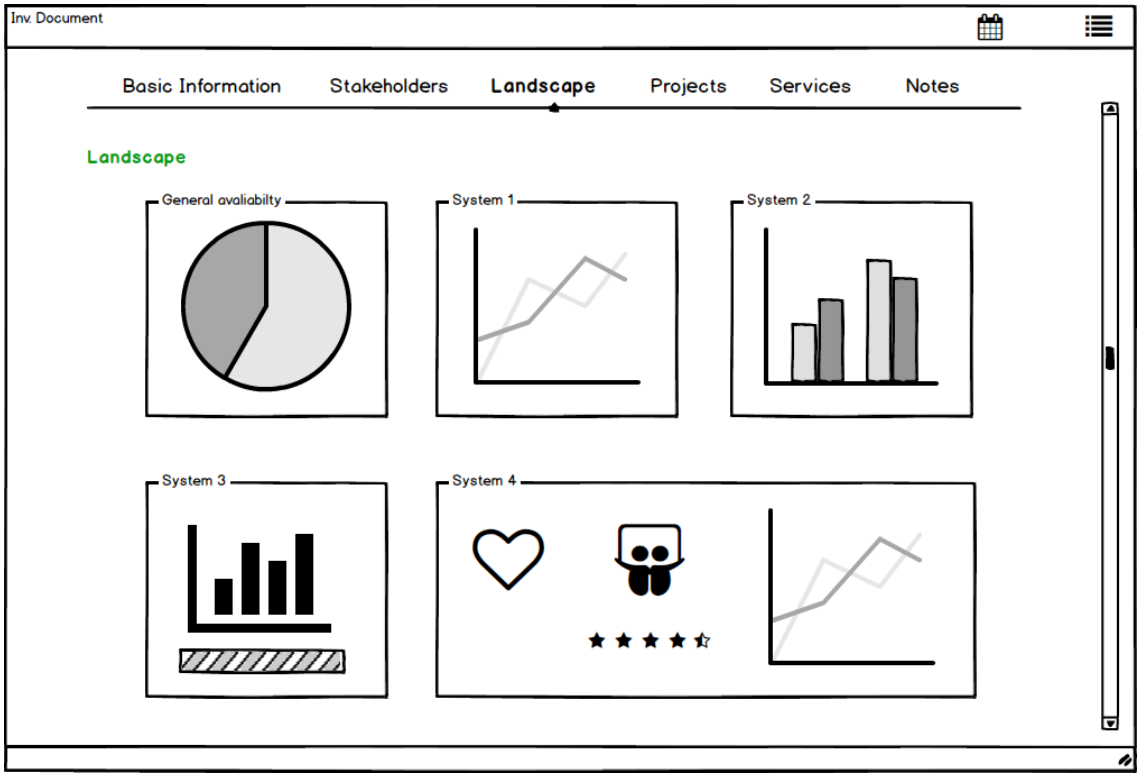


Figura 4.4 *Wireframe basado en Fiori Guideline*

Fuente: Elaboración propia en Balsamiq

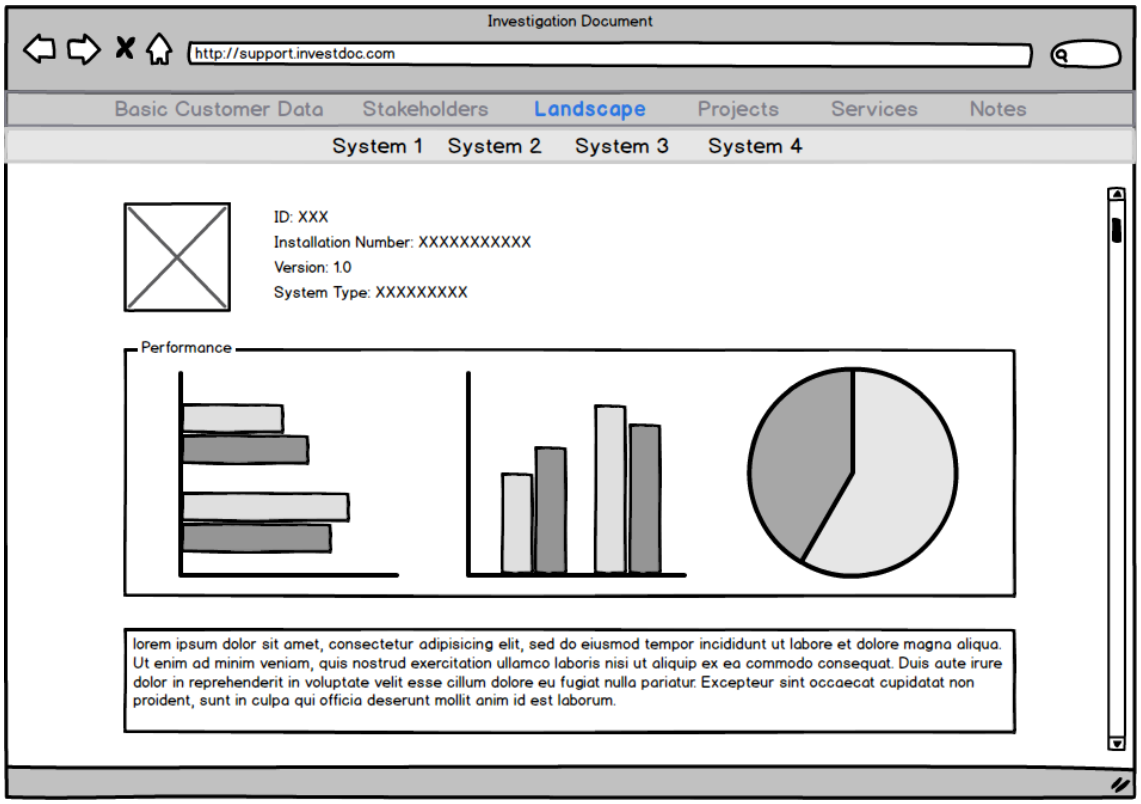


Figura 4.5 *Wireframe basado en estilo web tradicional con barra superior*

Fuente: Elaboración propia en Balsamiq

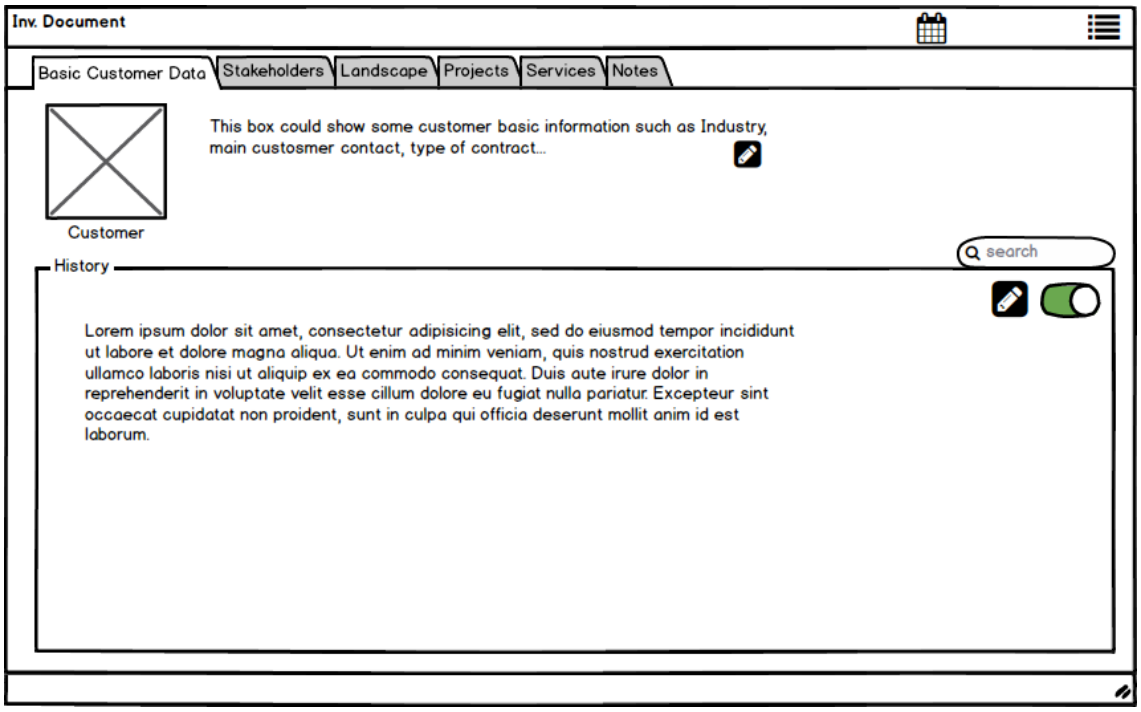


Figura 4.6 *Wireframe basado en pestañas con barra superior*

Fuente: Elaboración propia en Balsamiq

Como se puede observar los Wireframes son cada vez más complejos y con más soluciones gráficas como botones, iconos, etc. Pero siguen pareciendo que son modelos

sencillos y que se pueden cambiar y ser desechados. A fin de validar los diferentes diseños se procedió a entrevistar a un pequeño grupo de usuario y que contara sus pensamientos al respecto.

Una vez el usuario está delante del Wireframe, se le debe dar una pequeña información para contextualizar que representa esa pantalla que está viendo, al igual que información se está buscando validar. Tras las respuestas de los usuarios, los diseños que más gustaron fueron los que tenían una barra en la parte superior con las diferentes secciones.

Llegado a este punto se decidió crear un escenario de prueba. Para ello se crearon Wireframes de todas y cada una de las pestañas que forman un Documento de Investigación. Con la siguiente intención:

1. Verificar el diseño básico y grafico del Documento de Investigación
2. Comprobar si el usuario entiende como accedes a los datos y el contenido de las diferentes áreas
3. Comprobar la sencillez de cada una de las pantallas

Para ello se utilizó una herramienta interna de XYZ. La herramienta está especialmente desarrollada para realizar estudios y escenarios de prueba de prototipos. Una vez todos los Wireframes se han subido a la plataforma se pueden crear conexiones entre las diferentes pantallas, como si de un mapa se tratase.

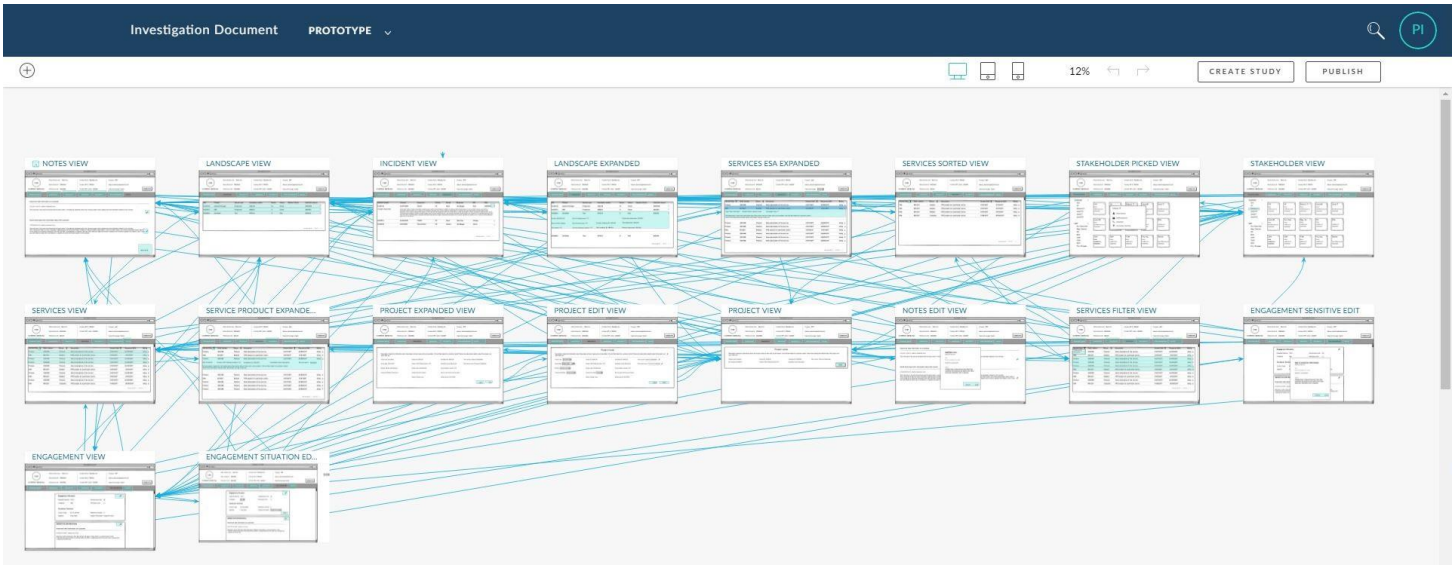


Figura 4.7 *Conexión entre los diferentes Wireframes*

Fuente: Elaboración propia

Una vez todas las conexiones están establecidas, hay que crear el estudio. Para lo cual se establecieron una serie de preguntas que los usuarios debían de responder.

Para ello se propuso a los usuarios respondieran a tres preguntas:

1. ¿Le gusta la disposición de la información del cliente en la pantalla?
   La respuesta se dejó como texto libre, para que pudieran expresarse libremente.

2. Navega a lo largo del prototipo, clicando en cada una de las pantallas tratando de ver información sobre el cliente. Después por favor completa la sección de

Feedback acerca la sencillez con la que navega y encuentra la información en este prototipo.
La respuesta se dejó como texto libre, para que pudieran expresarse libremente.

3. ¿Podría listar cual es, según su criterio la importancia de cada una de las pestañas?
   La respuesta se configuro como lista de prioridad para que el estudio fuera más sencillo.

La encuesta del prototipo fue mandada a los usuarios y mientras se escribe esta sección está teniendo lugar el estudio.

Posteriormente se procederá al estudio de los resultados generados por la aplicación de prueba. Se pretende estudiar cuál ha sido la reacción de los usuarios a este primer prototipo y aplicar esas recomendaciones a posteriores iteraciones del diseño.

Una vez el prototipo este totalmente estudiado se procederá a su desarrollo por el equipo de desarrollo de software situado en Alemania.

En el **Anexo 4 - Wireframes** se pueden encontrar muchos de los diseños que se crearon y enviados a los usuario finales para que fueran revisados.

4.3.3 Espacio de la Implementación

La fase de implementación todavía no ha empezado dado que, la fase de diseño sigue estando sin estar totalmente completada. Se necesitan un par de iteraciones más para pulir el diseño a alto nivel de la herramienta. Mientras se termina el equipo de desarrollo está formándose en la metodología **SCRUM** y en el framework Polymer.

Una vez el Espacio del Diseño esté terminado, y toda solución al problema identificado en el Espacio del Problema este delimitado y estudiado. Se tendrán suficientes argumentos como para poder saltar al **Espacio de la Implementación**, o desarrollo del software.

Los tiempos están delimitados en el proyecto global, se pretende tener un primer prototipo funcional durante el primer trimestre del año 2017. Una vez el prototipo este desplegado en la nube, se entregara a un grupo reducido de usuarios para que lo prueben, la prueba estará monitoreada, a fin de extraer el mayor beneficio de la prueba.

En caso de que la prueba sea un éxito, y se tenga buen Feedback de los usuarios. Se tendrán suficientes argumentos para conseguir más recursos tanto humanos como económicos, a fin para proseguir con el proyecto, el diseño e implementación de las demás funcionalidades identificadas en el Espacio del Problema.

Una vez el proyecto sea conocido y esté bien posicionado económicamente, la idea es tener un primer prototipo funcional completo para las Personas Amy y Tara para el primer trimestre del año 2018, y un sistema totalmente productivo para el último trimestre del 2018.

Por lo que el proyecto completo habrá durado para esas fechas cerca de tres años de duro trabajo.

5. Conclusión

La realización de este Trabajo de Fin de Grado se planteó con la intención de estudiar las necesidades, objetivos y responsabilidades de los trabajadores de un Departamento de Soporte junto con los sistemas de información actuales que le dan soporte, para realizar una propuesta de mejora. La problemática era totalmente real y ciertamente preocupante pudiendo afectar a cómo percibe el cliente al Departamento de Soporte.

Al principio del trabajo se plantearon una serie de objetivos, de los cuáles se explicará a continuación el nivel de comprensión y de alcance adquirido:

Analizar y comprender la problemática existente en los sistemas actuales del Departamento de Soporte.

Este objetivo lo considero cumplido en tanto en cuanto se ha podido entender la problemática. Sin embargo, para alcanzar esta problemática, se tuvo que realizar un esfuerzo muy grande con los usuarios finales, tratando de sonsacar que es lo que falla en el sistema. Para ello se utilizaron diferentes técnicas como son las entrevistas personales a usuarios, la observación directa, así como una primera encuesta lanzada globalmente tratando de ver si ese problema que se pensaba que existía era algo endémico o era global del departamento.

Mediante la encuesta se confirmó que los problemas eran reales, y estaban presentes globalmente en el departamento, se identificaron muchos problemas que fueron contemplados en los requerimientos.

La observación directa de los usuarios ha formado parte del estilo del proyecto, al tener un acceso muy fácil a los usuarios.

Analizar las necesidades, objetivos, responsabilidades y problemas de los diferentes perfiles (Personas) que interactúan con los sistemas.

Este objetivo lo considero cumplido, en la medida en que se identificaron a los diferentes roles en el departamento, así como se establecieron cuáles eran todas las características de estos usuarios.

Para poder alcanzar este objetivo se realizaron principalmente entrevistas tanto presenciales como telemáticas, a representantes de cada uno de los roles. Tratando de validar la información extraída de la iteración anterior, siendo cada vez más completa y verídica su carta de presentación.

Analizar los requerimientos del nuevo sistema, basándose en las necesidades del usuario final.

En cuanto a este objetivo no se puede juzgar como totalmente cumplido teniendo en cuenta que solamente se ha limitado a los requerimientos de un grupo de usuarios. Sin embargo, ese usuario es quien realmente utiliza la herramienta dado que el resto de grupos de usuarios se nutren de datos que se extraerían del sistema. Además de que la limitación en el presupuesto y en el personal ha hecho que se limitara el alcance.

En este objetivo se podrían incluir líneas futuras de este proyecto, como puede ser realizar los requerimientos para el resto de Personas a fin de crear un sistema que pueda ser utilizado globalmente por el Departamento de Soporte.

Analizar, comprender y aplicar la metodología Design Thinking a un caso real.

Debido a que esta metodología de resolución de problemas es una competencia que cada vez más buscan los departamentos de recursos humanos, es una parte del proyecto que se desarrolló de una manera más extensa.

Las posibilidades de aplicación del Design Thinking no se limitan al mundo del Software sino que puede ser utilizado en cualquier situación que se busque resolver un problema, la creación de un producto innovador, reinventar productos o servicios ya existentes. La limitación está en lo que es factible técnicamente, viable económicamente y si eso tiene usabilidad pero no en la propia metodología.

Diseñar una nueva la Experiencia del Usuario, basada en los últimos estilos de diseño junto con los principios más comprobados en la industria.

Finalmente este objetivo ha sido cumplido en tanto en cuanto se han creado muchos diseños, diferentes y siempre pidiendo Feedback a los usuario finales para ir puliendo y mejorando los diseños. Sin embargo, por otro lado, el diseño no ha podido ser totalmente confirmado y establecido. A día de hoy los diseños siguen cambiando y se sigue recibiendo Feedback de los usuarios.

Al ser un sistema tan grande y tan complejo, incluso la ventana más sencilla tiene muchos elementos y nada se puede dejar al azar.

A modo de conclusión personal, formar parte de este equipo desde prácticamente el nacimiento del proyecto me ha permitido tener una visión casi de la totalidad de lo que sucede en él; mejorar las competencias transversales como el trabajo en equipo y en la comunicación; la necesidad de aprender rápido; trabajar bajo presión y, así como, una mejora en la resolución de problemas. Por último indicar que la realización de las prácticas en una multinacional tan importante y el Trabajo de Fin de Grado me va a permitir acceder al mercado laboral al completar el ciclo Universitario dado que continuaré en la compañía trabajando una vez cerrado el ciclo.

6. Bibliografía

Bibliografía utilizada:

ACM - Association for Computing Machinery. (29 de 07 de 2009). *ACM SIGCHI Curricula for Human-Computer Interaction*. Obtenido de Sigchi: http://old.sigchi.org/

Android Developer - Google Inc. (3 de 12 de 2013). *Canal YouTube - Android Developers*. Obtenido de YouTube: https://goo.gl/XqKQMS

Apple Inc. (11 de 10 de 2016). *Apple Developer*. Obtenido de UI Design Do's and Don'ts: https://developer.apple.com/design/tips/

Apple Inc. (2016). *Design Guideline*. Obtenido de https://developer.apple.com/design/

Balsamiq. (2016). *Balsamiq*. Obtenido de https://balsamiq.com/

Beck, K. (2001). Manifesto for Agile Software Development.

Boehm, B. (1988). A Spiral Model for Software Development and Enhancement. .

Canós, J. H., Letelier, P., & Penadés, M. C. (2003). Metodologías Ágiles en el Desarrollo del Software. *Taller realizado en el marco de las VIII Jornadas de Ingeniría del Software de Datos, JISBD* (págs. 9-16). Alicante: Grupo ISSI.

Dayani-Fard. (1999). "Legacy Software Systems: Issues, Progress, and Challenges". En IBM, *IBM Technical Report: TR-74.165-k*.

Dix, A. (1999). *Design of User Interfaces for the Web*. Lancaster: Proc. User Interfaces to Data Systems Conference.

elementary.io. (2016). *Directrices de Interfaz Humana*. Obtenido de elementary.io: https://elementary.io/es/docs/human-interface-guidelines#human-interface-guidelines

Google - Alphabet Inc. . (2016). *Google Design*. Obtenido de Material Design : https://material.google.com/

Hanna, M. (1995). Farewell to Waterfalls. *Software Magazine*, 38-46.

Hasso-Plattner-Institut. (2016). *Hasso-Plattner Institut*. Obtenido de https://hpi-academy.de/en/design-thinking/what-is-design-thinking.html.

IDEO - Tim Brown. (10 de 11 de 2016). *IDEO - Design Thinking*. Obtenido de https://designthinking.ideo.com/.

Innovation Center Network. (2016). *Innovation Center Network SAP*. Obtenido de https://icn.sap.com/projects/knowledge-workspace.html

Jacobson, I. (2002). A Resounding 'Yes' to Agile Processes—But Also More. En *Cutter IT Journal* (págs. 18-24).

Kim, G. J. (2015). *Human-Computer Interaction - ISBN 9781482233896*. Wilmington, DE, U.S: Auerbach Publications.

Liu, K. (1998). *Report on the First SEBPC Workshop on Legacy Systems*. Durham University.

Mandel, T. (1997). The Elements of User Interface Design. En T. Mandel. John Wiley & Sons.

Polymer Project. (2016). *Polymer*. Obtenido de Polymer Project: https://www.polymer-project.org/1.0/

Pressman, R. (2010). *Ingeniería del Software - Un enfoque práctico*. México: Mc Graw Hill.

Prieto Molero, X., & Pujol Domenech, A. (2016). *Un viaje a la historia de la informatica*. Valencia: Editorial Universitat Politecnica de Valencia.

PwC. (2016). *PwC*. Obtenido de PwC - Millennials at work: https://www.pwc.com/m1/en/services/consulting/documents/millennials-at-work.pdf

Rams, D. K. (2012). *Kingston University London - Faculty of Art, Design and Architecture*. Obtenido de The Design School: https://www.thedesignschool.co.uk/group19/files/Dieter-Rams.pdf

Reiner, J. (5 de Mayo de 2005). *arstechnica.com*. Obtenido de arstechnica.com: arstechnica.com

Rumbaugh, J., & Jacobson, I. (1999). The Unified Software Development Process. En J. Rumbaugh, I. Jacobson, & G. Booch. Boston, MA, USA: Longman Publishing Co., Inc.

SAP SE. (2016). *SAP SE*. Obtenido de SAP SE: www.sap.com

Solórzano, J. M. (2015). La configuración del iPod clásico. CULCyT. 46.

Sommerville, I. (2005). *Ingeniría del Software*. Madrid: Pearson Education.

Stanford University - Hasso Plattner Institute of Design. (2010). *Hasso Plattner Institute of Design*. Obtenido de Hasso Plattner Institute of Design: https://dschool.stanford.edu/wp-content/uploads/2011/03/BootcampBootleg2010v2SLIM.pdf

Thayer, R. H., & M. Dorfman. (1997). Software Requirements Engineering. En I. C. Press.

Tognozzi, B. (2001). *First Principles*. Obtenido de asktog: http://asktog.com/atc/principles-of-interaction-design/

7. Anexos

7.1 Anexo 1 - Storyboard Amy

1. Un nuevo cliente es asignado. Amy recibe un nuevo cliente e investiga los sistemas del cliente.
    1.1. Amy reconoce el nuevo cliente en la pantalla principal. Además recibe una notificación en el Email.
        1.1.1. La notificación muestra a un alto nivel la razón de esta asignación (Categoría, nueva Implementación, actualización, migración)
        1.1.2. Las categorías deben ser flexibles y deben poder cambiarse a lo largo de la relación con el cliente
    1.2. Amy selecciona el nuevo cliente y empieza el trabajo con su nuevo cliente
        1.2.1. Las categorías deben influir en el flujo del trabajo
    1.3. Amy revisa la información del cliente (Industria, sistemas, incidentes, historia del cliente con XYZ)
        1.3.1. Amy debe tomar notas de la información que crea que debe validar con el cliente
2. Amy se alinea con los diferentes Stakeholders.
    2.1. Amy idéntica quienes son los contactos clave que debe contactar
    2.2. Amy realiza una llamada con los diferentes contactos internos
    2.3. Amy fija en el calendario futuras llamadas con los Stakeholders
    2.4. Amy captura la información obtenida con los Stakeholders
3. Amy realiza la primera llamada con el cliente y genera un borrador del plan de ejecución
    3.1. Amy fija en el calendario la primera llamada con el cliente. Se presenta y envía una invitación al cliente.
    3.2. Amy llama al cliente.
        3.2.1. Amy debe ser capaz de ver toda la información del cliente mientras realiza la llamada
    3.3. Amy valida la información con el cliente y entiende cuales son las expectativas del cliente
    3.4. Amy introduce la información en el sistema
    3.5. Amy genera un borrador de cómo se va a resolver los problemas al cliente, ayudándose de la información obtenida en la llamada
    3.6. Amy debe validar este borrador con los expertos de cada materia
    3.7. Amy establece una segunda llamada con el cliente
4. Revisar y validar el plan de ejecución
    4.1. Amy presenta al cliente el plan de ejecución
        4.1.1. Comparte la pantalla con el cliente para ver los dos la misma información
        4.1.2. Amy y el cliente discuten si los objetivos son alcanzables
    4.2. Amy edita y ajusta los objetivos basándose en los comentarios del cliente
    4.3. Amy y el cliente firman el plan de ejecución
    4.4. Amy y el cliente acuerdan la periodicidad de sus reuniones para discutir los avances de en los objetivos
5. Amy lleva a cabo el plan de ejecución

5.1. Amy realiza todas las gestiones internas planteadas en el plan de ejecución
5.2. Amy reserva los servicios que ayuden a cumplir los objetivos
5.3. Amy envía recomendaciones al cliente basadas en las mejores practicas
5.4. Amy invita al cliente a eventos de importancia para el cliente
5.5. Amy actualiza el estado de los objetivos de acuerdo al progreso
5.6. Amy se alinea con los Stakeholders para ver el progreso de los objetivos
5.7. Amy tiene llamadas con el cliente mostrando que retos han alcanzado
5.8. Amy presenta de una manera visual (gráficos) los objetivos cumplidos

6. Amy muestra el valor creado sobre el cliente, analiza junto al cliente los éxitos que han alcanzado
6.1. Amy envía al cliente una encuesta de cómo ha sido la relación con Amy
6.1.1. La encuesta debe ser enviada a más de un contacto del cliente a fin de ser lo más objetiva posible
6.2. Amy revisa junto con Marc (Manager) los logros alcanzados

7.2 Anexo 2 - Cartas de Presentación

Tara - Líder de Equipo

Responsabilidades:

- Entender los mercados locales, regionales y globales así como sus requerimientos
- Manejar a los clientes de una región
- Hacer un seguimiento a los clientes que o estén en riesgo su continuidad con el contrato de soporte o bien tengan un producto de interés para XYZ
- Asegurarse que el resto del equipo tiene recursos suficientes para cumplir las expectativas del cliente
- Generar y reportar KPI(**Key Performance Indicators**, Indicadores Clave de Rendimiento en castellano) sobre la región de la que es responsable

Objetivos:

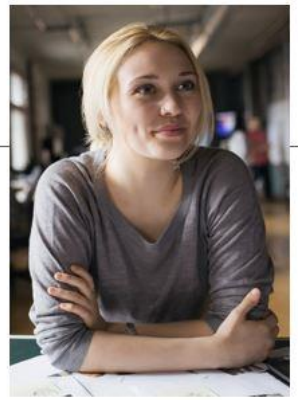
- Asegurarse de que los Asistentes de Soporte están creando valor a los clientes
- Asegurarse de que una región está cubierta de manera optima
- Demostrar al equipo el éxito alcanzado a través de reportes
- Asegurarse que los Asistentes de Soporte de su equipo estén entregando valor y ajustadas a las mejores practicas

Necesidades:

- Necesita asegurarse de tener suficientes recursos para cubrir una región
- Necesita tener una información relevante sobre el estado de una región
- Necesita tener una visión de la carga real de trabajo de su equipo para distribuir adecuadamente los recursos
- Necesita saber cuándo ha terminado la relación con un cliente
- Necesita tener una visión de la estrategia de la organización para recolocar recursos
- Necesita tener una plataforma para poder compartir información relevante sobre el estado de una región con los Stakeholders.

Problemas:

- Falta de automatización y estandarización en muchos de los procesos
- Falta de transparencia de las actividades realizadas por el Asistente de Soporte
- Difícil predecir la llegada de algún nuevo cliente para asegurarse de que hay suficientes recursos
- El número de clientes asignados a un cliente no necesariamente significa la carga de trabajo a la que está sometida
- Falta de herramientas de estadísticas para la toma rápida de decisiones



Tara

The Team Champion

"I want to ensure we provide a sustainable, scalable, profitable and high quality customer engagement and ensure appropriate coverage in my region."

About

28, single, 4 years of experience in XYZ Solution Support. I'm responsible for knowing my region and market along with the specific requirements and demands coming from it. I need to ensure I have the correct assignments according to priorities defined for customer categories and that the correct skills are assigned to the different customer needs.

Working mostly in the office or at home with regular face-2-face meetings with Advisors and regular remote meetings with other stakeholders

Works With

CONFIDENTIAL

Job Responsibilities

- Understand the local, regional and global market requirements and help determine best practices for engagements
- Handle customer assignment requests for her region/market unit
- Track risk customers, and make sure risk customers as well as customer escalations are dealt with appropriately
- Ensure the team has most up to date relevant information and resources required to fulfill their role and exceed the customers' expectations
- Maintain real-time accurate assignment overview for her market / team
- Report relevant KPIs on her market / team performance to her stakeholders, incl. higher management (adhoc, if required)
- Manage and facilitate stakeholders communications, incl. continuous feedback

Needs

- I need to have clearly planned resources to ensure customer coverage in my region
- I need relevant, comprehensive information about my region (one customer based view and one employee based view), ideally from one central data source
- I need to have real-time workload view of the colleagues in order to allocate assignments
- I need to know when colleagues are finishing assignments (based on engagement plans – planned view; actual disengagement – current view)
- I need clear visibility on the strategic direction including projects related to my organization and reallocate resources accordingly
- I need to be able to evaluate the health of a customer and specifically demonstrate customers at risk and trending analysis outlining the intensity of customer engagements
- I need my team of advisors to work with measurable engagement plans with key standardized templates to avoid variances that are being delivered (minutes/agenda/forms/reporting)
- I need one platform for sharing relevant information about the market with my stakeholders

Goals

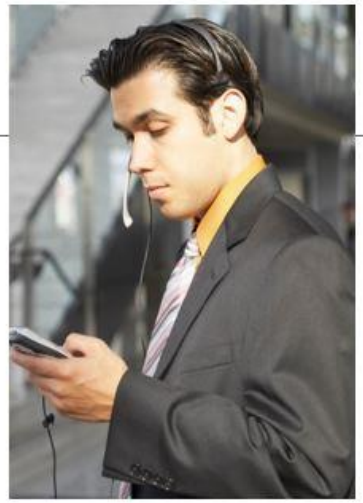
- Ensure customer success overall and create customer success Ensure optimal customer coverage in my region stories for the market
- Understand the local market and customer needs and adapt global best practices to this reality, if needed
- Align and liaise with stakeholders (inter- and intra-departmental) to achieve best business results
- Enable advisors to ensure the teams are delivering best service to customers and improve their job satisfaction
- Demonstrate team success via tracking and reporting

Pain Points

- Lack of automation and standardised templates
- No ability to accurately demonstrate the scope of issues and incidents to senior management
- Insufficient transparency of advisor activities for evaluation
- Hard to predict incoming new contracts to have sufficient buffer of resources for assignments.
- Number of Advisors' customer assignments is not necessarily related to their real workload
- Lack of sufficient reporting and reliance on other teams statistical reporting to make quick decisions

Figura 7. 1 *Carta de presentación de Tara la Líder de Equipo*

Fuente: Elaboración propia



Marc

The Manager

" I want to drive simplicity, develop amazing talent and ensure customer success"

About

40, married, 6 years of experience in XYZ Solution Support

I'm responsible for leading the change by breaking silos and enforcing real collaboration

I lead by example and I drive performance as well as promote diversity

Working mostly in the office I have regular face-2-face meetings with Advisors

In addition I have regular virtual meetings with management, functional leads, team leads and other internal stakeholders

Works With

CONFIDENTIAL

Job Responsibilities

- Ensure customer coverage in my region in close alignment market units and global ES & PC delivery strategy
- Understand the local market and adapt our global best practices to this reality, if needed
- Drive and develop the strategy for the region and ensure retention of customers
- Drive employee performance, incl. talent management and people development
- Nurture creative ideas, considering advisors' workload and capacity, to gain customer success and satisfaction
- Ensure my team knows the strategy of the company and collaborates to achieve its goals

Needs

- I need to have a clear view on available resources in my region and on the customer pipeline to be able to plan coverage accordingly; in particular I need to know when new customers come in and when a customer decides not to renew an existing contract or changes the support model
- I need relevant, comprehensive information about my region (one customer based view and one employee based view), ideally from one central data source. Reporting needs to be simplified
- I need effective and efficient processes and tools in place to ensure fair and concise evaluation of the work done by my direct reports, based on objective and comparable criteria (qualitative & quantitative)
- I need transparency about individual, team, regional and global performance in relation to shared KPIs
- I need a holistic understanding of customers "health" status, incl. size and status of the customer, licenses, go-live dates, projects)

Goals

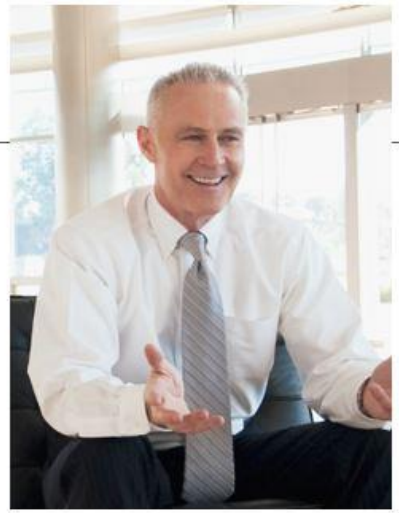
- Drive simplicity
- Develop amazing talent
- Ensure customer success and satisfaction
- Improve overall customer support experience
- Ensure retention and renewals
- Demonstrate success to higher management

Pain Points

- No centralized views/dashboards for different areas, such as operations, employee topics customer coverage
- Spending too much effort on non-value adding tasks such as manual administrative efforts and checking if messages are being followed up
- Decrease in motivation and productivity of my team due to
  - lack of clarity on KPIs
  - lack of clarity how other colleagues and teams are being evaluated
  - processes are not working smoothly across departments
  - no influence of other peoples'/teams' performance
- Overcomplicated and not always up-to-date reporting, to allow for fast decision making and to analyze trends and needs effectively

Figura 7. 2 *Carta de Presentación Marc el Manager*

Fuente: Elaboración propia



Frank

The Functional Lead

"I want to ensure we provide higher value customer engagements and a scalable operational model

About

46, married, 12 years of experience in XYZ Support.

I need to understand and oversee the regional and global market requirements and help determine best practices for engagements while ensuring processes are aligned.

Overall, I want to create and ensure higher value engagements for customers

Working mostly in the office or at home with regular face-2-face meetings market champions.

Works With

CONFIDENTIAL

Job Responsibilities

- Responsible for the global delivery strategy
- Understand the local, regional and global market requirements and help determine best practices for engagements
- Develop strategic deliverables to form part of the engagement model, incl. the way they are delivered (workflows)
- Develop a framework around defining team roles and responsibilities; create and advertise operational roles

Needs

- I need access to and understanding of regional heads' needs, pain points, and capacity due to workload/assignments
- I need to be able to react to strategic direction of senior management if there are changes
- I need transparent reporting
- I need a clear understanding and visibility of global initiatives to avoid duplication of efforts
- I need trending analysis outlining the intensity of customer engagements

Goals

- Ensure customer success
- Create successful engagement models
- Create measurable customer engagements allowing to show customer success
- Improve the work balance, effectiveness and job satisfaction of Market Champions and Advisors through enablement
- Ensure business requirements are met by supporting the regional managers
- Ensure regional business managers get the right reports to use for regional reporting for upper management

Pain Points

- No information about dormant customers or when customers will become active to ensure appropriate staffing
- Reliance on other statistical reporting to make quick decisions
- Lack of accurate/sufficient reporting
- Duplication of work efforts
- Lack of initiative reinforcement from delivery management

Figura 7. 3 *Carta de Presentación de Frank, el Líder Funcional*

Fuente: Elaboración propia

#	Persona	Usar Story	Paso relacionado en el Storyboard	Editable	Función Relacionada
1	Amy	Saber cuándo un cliente es asignado bajo mi nombre, para poder empezar a trabajar con mi nuevo cliente a tiempo	1		Notificación
2	Amy	Ser notificado a través de diferentes canales, y así no tener el riesgo de perder una notificación	1		Notificación
3	Amy	Saber la prioridad y la categoría del nuevo cliente, para así actuar de manera apropiada	1		Notificación
4	Amy	Poder elegir cuando empezar a trabajar con el cliente para así poder terminar el resto del trabajo	1		
5	Amy	Revisar y entender cuáles son las acciones requeridas de esta nueva tarea, para así poder organizar de manera eficiente mi trabajo	1		Documento de Investigación
6	Amy	Revisar y editar el Documento de Investigación inicial para así poder preparar la reunión con lo Stakeholders internos	1		Documento de Investigación
7	Amy	Reconocer todos los contactos principales de mi nuevo cliente, para así extraer el mayor rendimiento a la reunión interna	2	Si	Colaborar
8	Amy	Crear y mantener mi VAT, y así organizar mis próximas reuniones internas	2		Colaborar
9	Amy	Invitar a todos los miembros de mi VAT, y así poder conseguir toda la información requerida de mi nuevo cliente	2	Si	Colaborar
10	Amy	Tener diferentes maneras de comunicación con mi VAT, para así alinearme correctamente con mi VAT	2		Notificación
11	Amy	Saber el nivel de actualización que desean tener cada miembro de mi VAT, y así poder involucrarlos adecuadamente	2		Colaborar
12	Amy	Tener una manera sencilla de editar mi calendario, y así poder alinearme con mi VAT	2		Colaborar
13	Amy	Ver mi calendario de reuniones con mi VAT en el sistema, para así poder organizar apropiadamente cada una de las reuniones	2		Colaborar
14	Amy	Saber la disponibilidad de los miembros de mi VAT, y así mantener actualizados a todos los miembros y buscar a un sustituto en caso necesario	2		Colaborar
15	Amy	Editar fácilmente el Documento de Investigación, y así poder incorporar toda la información que he capturado en la reunión con el VAT	2	Si	Documento de Investigación
16	Amy	Poder determinar cuando los cambios que he realizado en el Documento de Investigación son visibles, y así poder controlar que la información es veraz	2		Documento de Investigación
17	Amy	Saber quien cambia o actualiza el Documento de Investigación, y así poder evitar fallas en la información	2	Si	Documento de Investigación

Figura 7. 4 *Requerimientos de la Persona Amy 1-17*

Fuente: Elaboración propia

18	VAT Team	Especificar mi nivel de prioridad de información, y así contribuir de la manera mas adecuada	2		Colaborar
19	VAT Team	Acceder y editar el Documento de Investigación, y así poder dar información muy relevante sobre el cliente a Amy	2	Si	Documento de Investigación
20	Amy	Acceder de manera sencilla al calendario, y así establecer reuniones de manera eficiente con el cliente	3		Colaborar
21	Amy	Conocer el calendario de mi cliente, y así poder invitarlo a las reuniones sin producirle conflictos	3		Colaborar
22	Amy	Saber todos los contactos relevantes de mi cliente, y así no olvidar invitar alguien a la reunión	3		Colaborar
23	Amy	Ver la disponibilidad de los miembro de mi VAT que han aceptado estar involucrados, y así poder invitarlos a la reunión	3		Colaborar
24	Amy	Tener una plantilla de correo a mano, y así poder comunicarme con el cliente de un manera adecuada y no olvidar ninguna información importante	3		Colaborar
25	Amy	Poder ver las próximas llamadas de mi calendario de una manera sencilla, y así no olvidar ninguna	3		Colaborar
26	Amy	Acceder de manera sencilla a los detalles de las próximas llamadas, y así no perder tiempo estableciendo los detalles de las llamadas	3		Colaborar
27	Amy	Revisar el Documento de Investigación antes de la primera llamada con el cliente , para estar lo mejor preparada	3	Si	Documento de Investigación
28	Amy	Resaltar cualquier información que no este disponible en el Documento de Investigación, y así asegurarme que esa información la pregunto en la llamada	3		Documento de Investigación
29	Amy	Usar el Documento de Investigación para liderar la llamada, y así poder compartir y validar toda la información relevante con mi cliente y capturar la nueva información	3		Documento de Investigación
30	Amy	Poder mantener notas y actualizaciones en el propio Documento de Investigación, y así poder documentar toda la información de la llamada de una manera mas sencilla	3	Si	Documento de Investigación
31	Amy	Poder cambiar entre una vista interna y externa de Documento de Investigación, y así poder compartir la pantalla con el cliente	3		Documento de Investigación
32	Amy	Terminar de completar el Documento de Investigación junto con el cliente, y así tener suficiente información para crear el Plan de Ejecución	3	Si	Documento de Investigación
33	Amy	Poder generar de manera automática un borrador del Plan de Ejecución, basándose en los Objetivos marcados en el Documento de Investigación, y así tener un borrador con el que poder empezar a trabajar	3		Plan de Ejecución
34	Amy	Poder compartir el borrador del Plan de Ejecución con mi VAT para obtener Feedback, y así poder realizar ajustes al plan en caso de ser necesario	3		Plan de Ejecución

Figura 7. 5 *Requerimientos de la Persona Amy 18-34*

Fuente: Elaboración propia

35	Amy	Conocer la disponibilidad del Back Office, para así poder validar la versión final del Plan de ejecución	3		Colaborar
36	Amy	Poder compartir el Plan de Ejecución, y así poder validar en futuras actualizaciones	3		Plan de Ejecución
37	Amy	Compartir el Plan de Ejecución con el cliente, y así tener una visión común del plan	4		Plan de Ejecución
38	Amy	Poder cambiar entre una vista interna y externa del Plan de Ejecución, y así poder compartir la información adecuada con el cliente	4		Plan de Ejecución
39	Amy	Poder acceder a información relevante para mi cliente, basándose en la categoría de la relación, y así poder dar solo información relevante a mi cliente	4		
40	Amy	Poder anotar las minutas de la llamada con el cliente, y así poder referirse a esa información mas adelante, en caso de necesitarse	4		Colaborar
41	Amy	Acceder y exportar las anotaciones creadas, y así poder tener esa información guardada offline	4		
42	Amy	Ver cada uno de los objetivos e información con una fecha limite del Plan del Ejecución, y así poder discutir la relevancia de esa con el cliente	4		Plan de Ejecución
43	Amy	Poder editar los objetivos, y así adecuarlos a los cambios propuestos por el cliente	4	Si	Plan de Ejecución
44	Amy	Poder compartir el Plan de Ejecución con el cliente de una manera potente, y así facilitar la colaboración	4		Colaborar
45	Amy	Dar privilegios de edición al cliente sobre el Plan de Ejecución, y así facilitar la colaboración	4	Si	Colaborar
46	Amy	Determinar el punto en el tiempo en el que los cambios sobre el Plan de Ejecución son visibles sobre todas los perfiles autorizados, y así poder controlar el flujo de información	4		Colaborar
47	Amy	Recibir una notificación cuando el cliente haga una modificación sobre el Plan de Ejecución, y así ser informada adecuadamente	4		Notificación
48	Amy	Poder concertar reuniones habituales con el cliente, y así asegurarse de que el cliente esta actualizado	4		Colaborar
49	Amy	Recibir recordatorios antes de una llamada/reunión, y así no olvidarla	4		Notificación
50	Amy	Editar el Plan de Ejecución para asegurarse de que los objetivos ya cumplidos están reflejados, y así mantener actualizado al cliente	4	Si	Plan de Ejecución
51	Cliente	Poder firmar de manera manual el Plan de Ejecución, y así poder tener tener una prueba física del acuerdo	4		Plan de Ejecución
52	Cliente	Poder realizar pequeños ajustes en las fechas y objetivos sobre el Plan de Ejecución firmado, para actualizarlo en caso de que el cliente lo solicite	4	Si	Plan de Ejecución

Figura 7. 6 *Requerimientos de la Persona Amy 35-52*

Fuente: Elaboración propia

53	Cliente	Recibir una notificación cuando el cliente firme el Plan de Ejecución, y así estar informado adecuadamente	4		Notificación
54	VAT Team	Recibir una copia del Plan de Ejecución firmado en pdf, y así poder revisarlo offline	4		Notificación
55	VAT Team	Saber quién quiere formar parte de la llamada final con el cliente, y así invitarlos en caso de necesitarlo	6	Si	Colaborar
56	Amy	Invitar a todos los Stakeholders relevantes a la llamada final, y así todo el mundo está al tanto de la reunión y puedan unirse en caso de que estén interesados	6		Colaborar
57	Amy	Tener un claro resumen del estado de los objetivos y servicios parecidos, y así poder presentar de una manera comprensiva que se hecho y que se ha alcanzado	6		Plan de Ejecución
58	Amy	Entender los éxitos, y así poder presentarlo al cliente de manera que ellos entiendan el valor del Plan de Ejecución	6		Plan de Ejecución
59	Amy	Compartir los éxitos de una manera sencilla, de una manera muy visual, y así poder compartir esos éxitos con otros Stakeholders	6		Plan de Ejecución
60	Amy	Comprender que es lo que el cliente considera el éxito en relación con su negocio, y así poder subrayar los éxitos de interés	6		
61	Amy	Documentar las lecciones aprendidas en tiempo real, y así poder realizar tareas similares en casos similares	6	Si	Plan de Ejecución
62	Amy	Revisar y firmar el Plan de Ejecución, incluyendo las lecciones aprendidas, y así poder documentar todos los puntos de acuerdo a lo largo de la relación	6		Plan de Ejecución
63	Amy	Asegurarse de que mi cliente me da Feedback acerca de mi a lo largo de la relación, y así poder saber el rendimiento y donde poder mejorar	6		Feedback
64	Amy	Asegurarse de que yo y mi manager revisamos el Feedback obtenido de los clientes, y así poder discutir los resultados y definir futuras acciones	6		Feedback
65	Amy	Cerrar la relación y archivar todos los documentos generados, y así poder ser consultados para futuras relaciones con el cliente	6	Si	Plan de Ejecución

Figura 7. 7 *Requerimientos de la Persona Amy 53-65*

Fuente: Elaboración propia

7.4 Anexo 4 - Wireframes

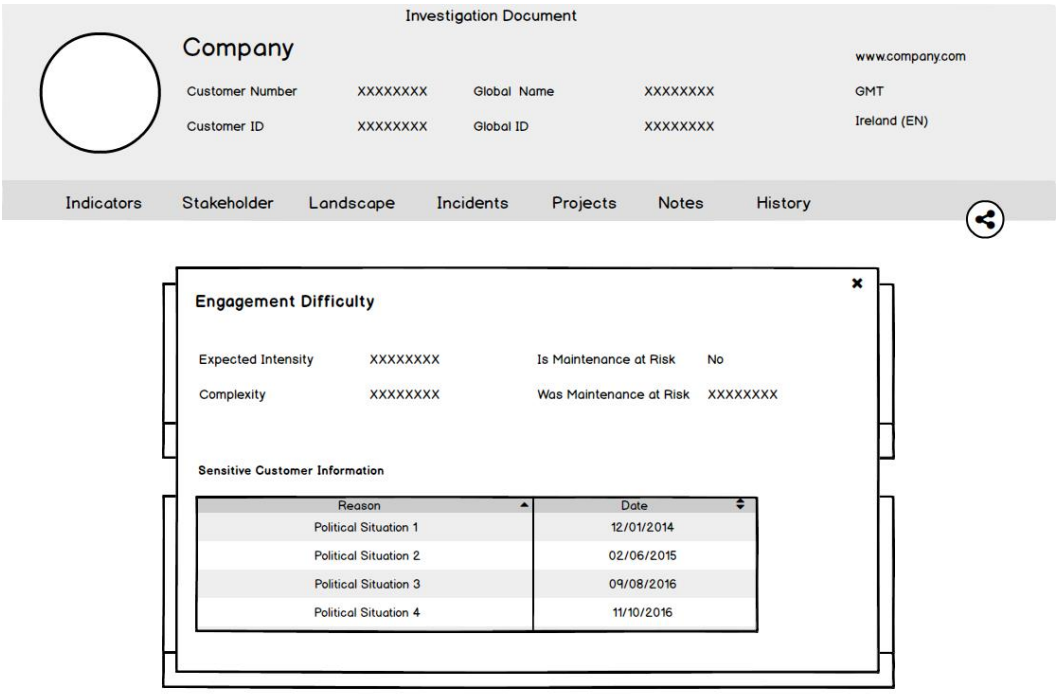


Figura 7. 8 *Wireframe Material Design, Dificultad de la Relación*

Fuente: Elaboración propia en Balsamiq

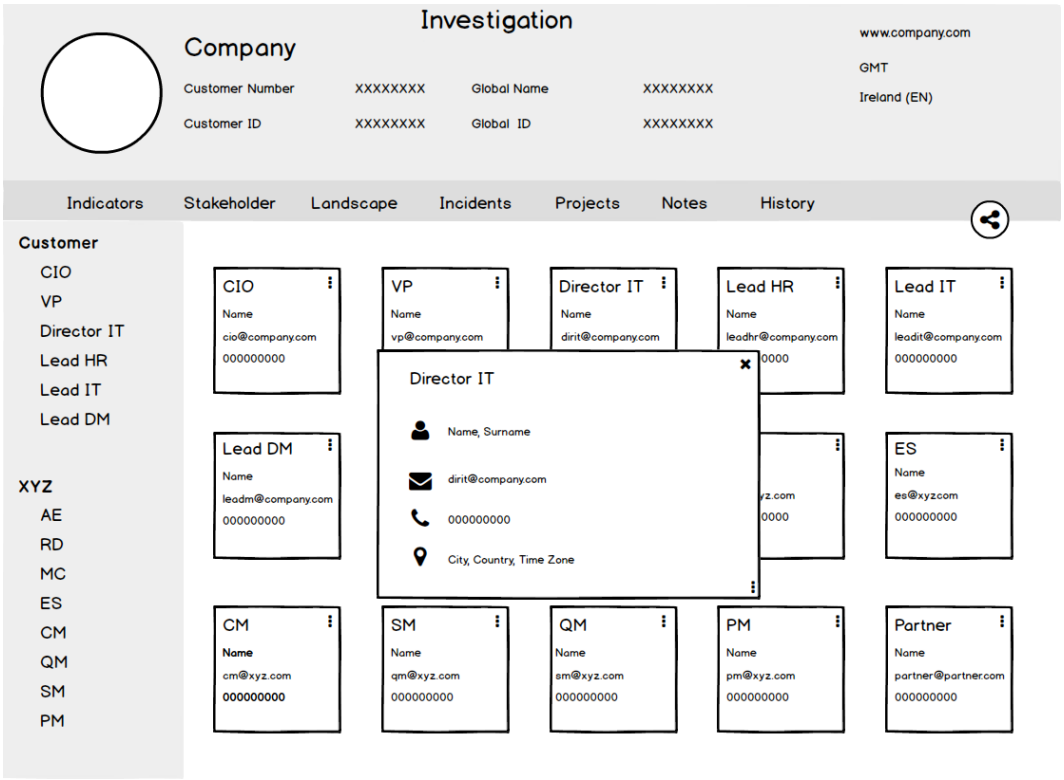


Figura 7. 9 *Wireframe Material Design, Stakeholders detalle Director de IT*

Fuente: Elaboración propia en Balsamiq

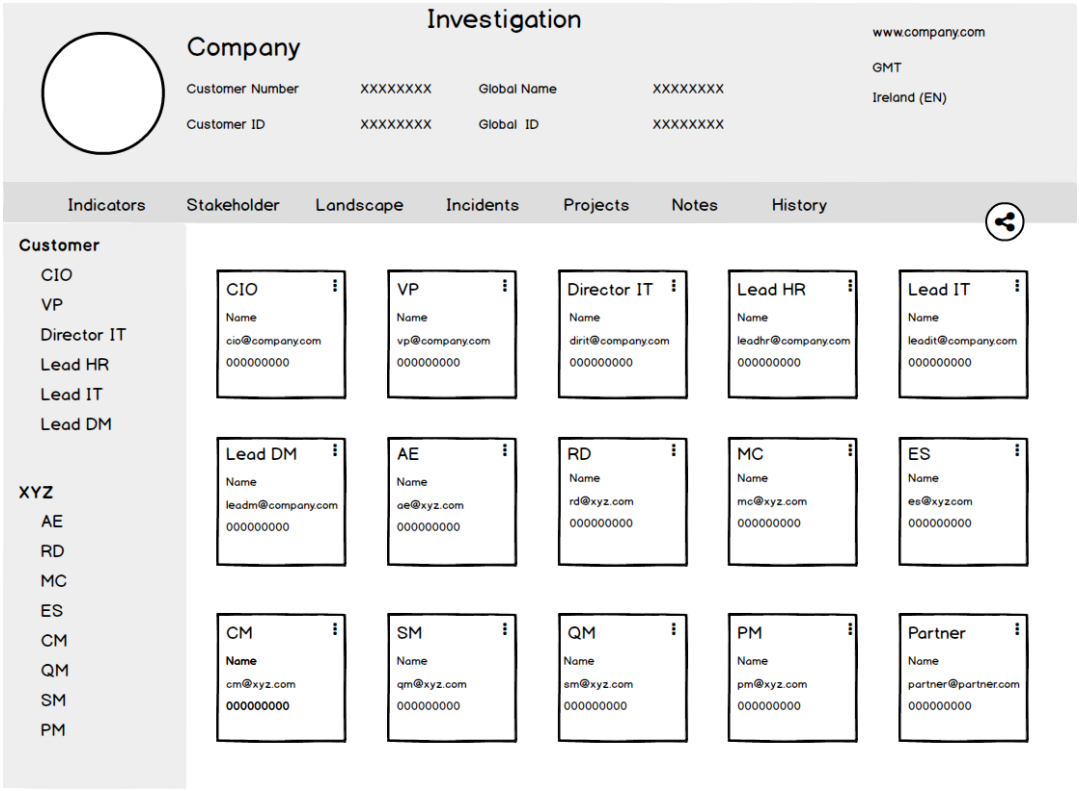


Figura 7. 10 *Wireframe Material Design, Stakeholders*

Fuente: Elaboración propia en Balsamiq

Figura 7. 11 *Wireframe Material Design, Indicadores de la relación*

Fuente: Elaboración propia en Balsamiq

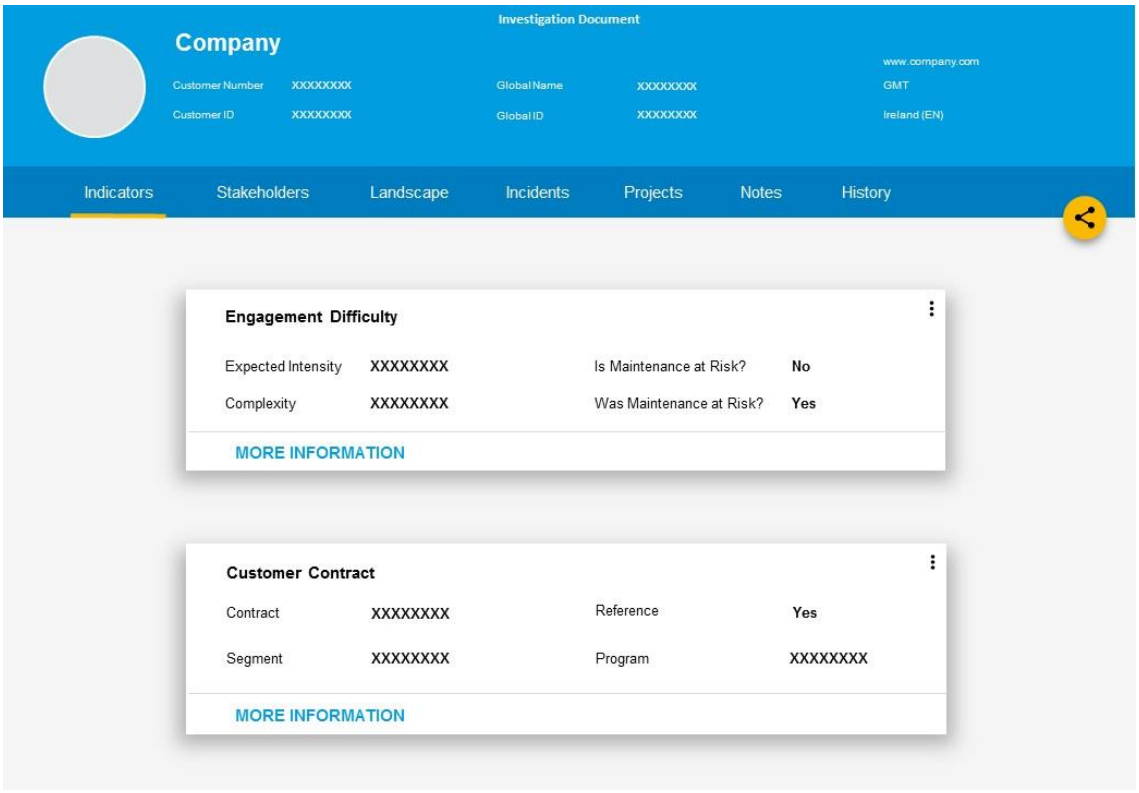


Figura 7. 12 *Wireframe Material Design, Indicadores de la Relación*

Fuente: Elaboración propia en Photoshop

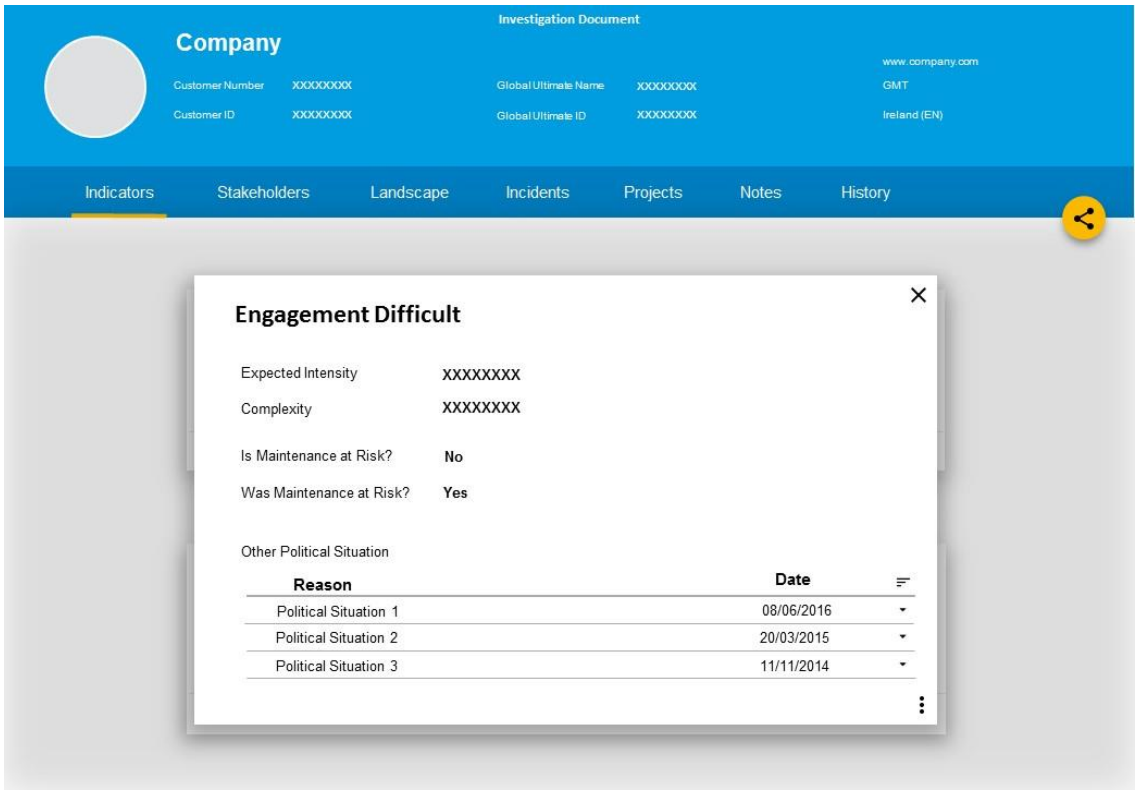


Figura 7. 13 *Wireframe Material Design, Dificultad de la Relación*

Fuente: Elaboración propia en Photoshop

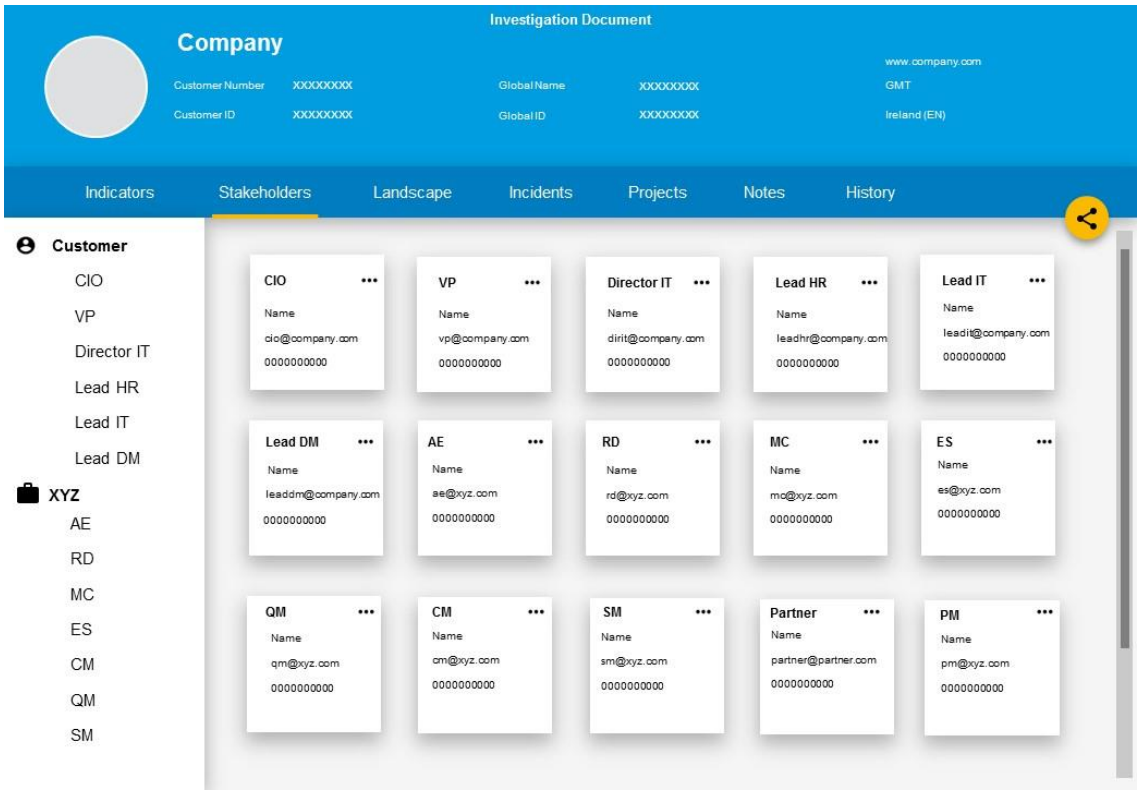


Figura 7. 14 *Wireframe Material Design, Stakeholders*

Fuente: Elaboración propia en Photoshop

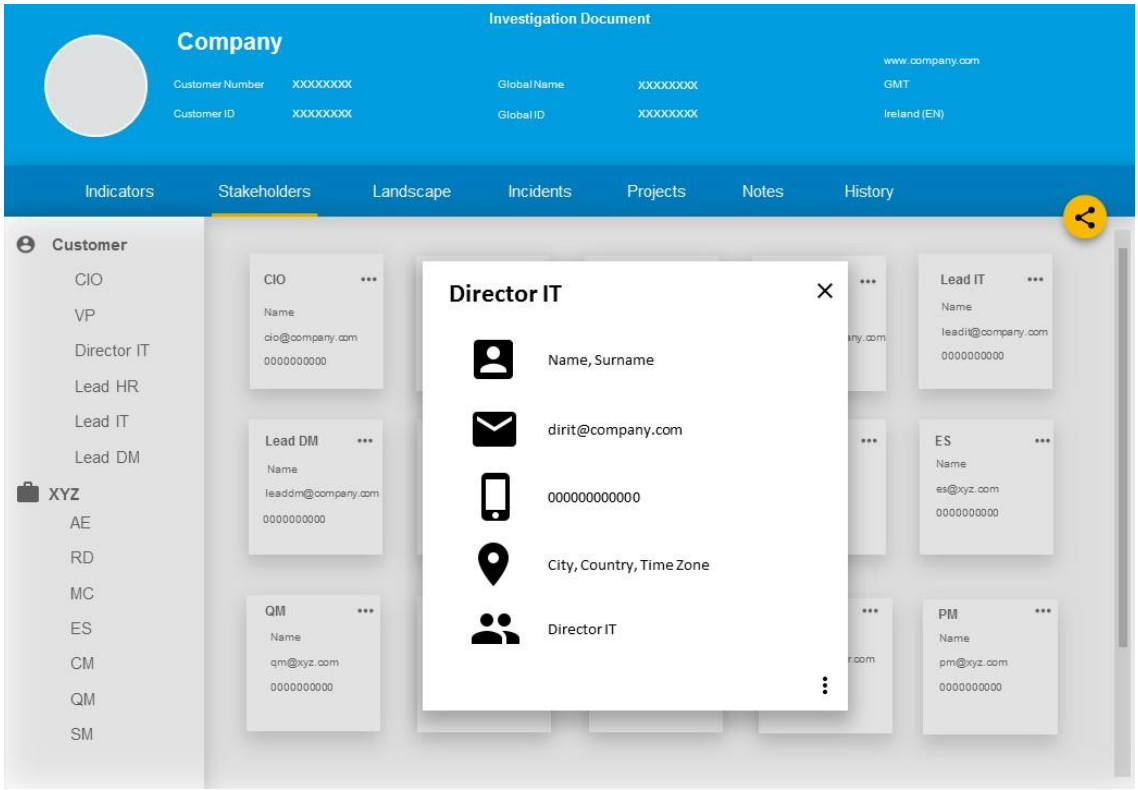


Figura 7. 15 *Wireframe Material Design, Stakeholders detalle en Director IT*

Fuente: Elaboración propia en Photoshop

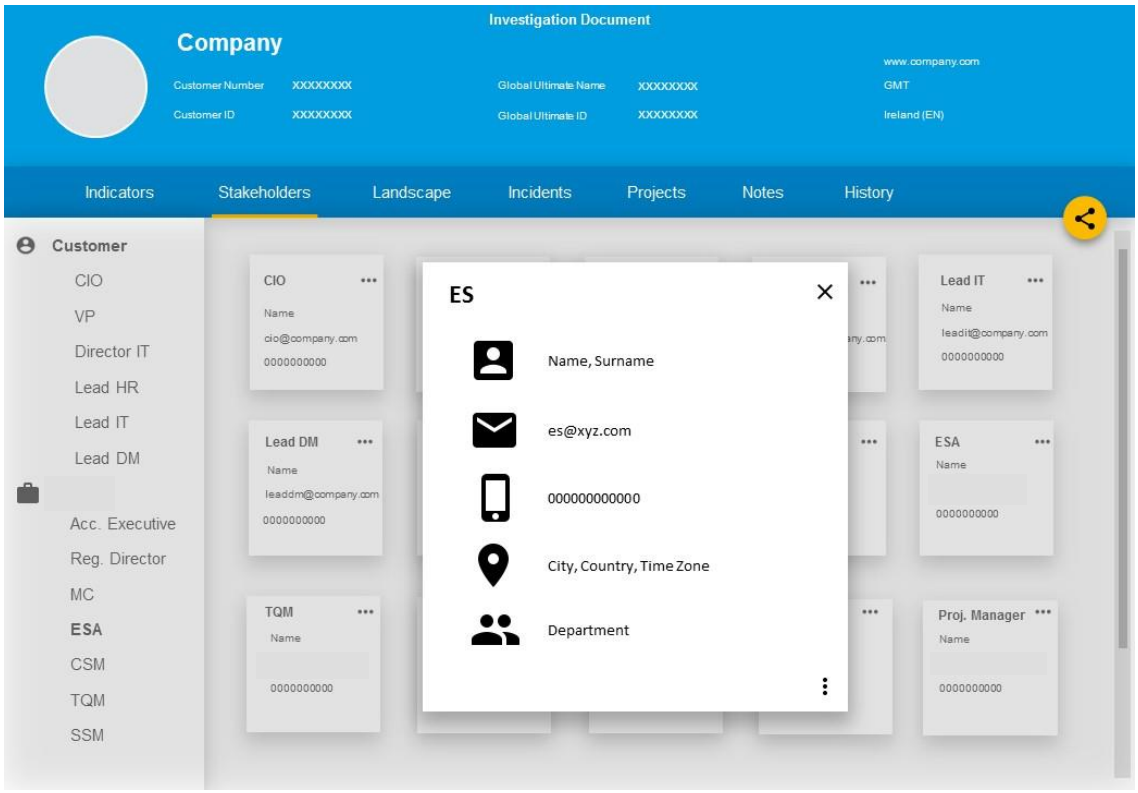


Figura 7. 16 *Wireframe Material Design Stakeholders detalle en ES*

Fuente: Elaboración propia en Photoshop

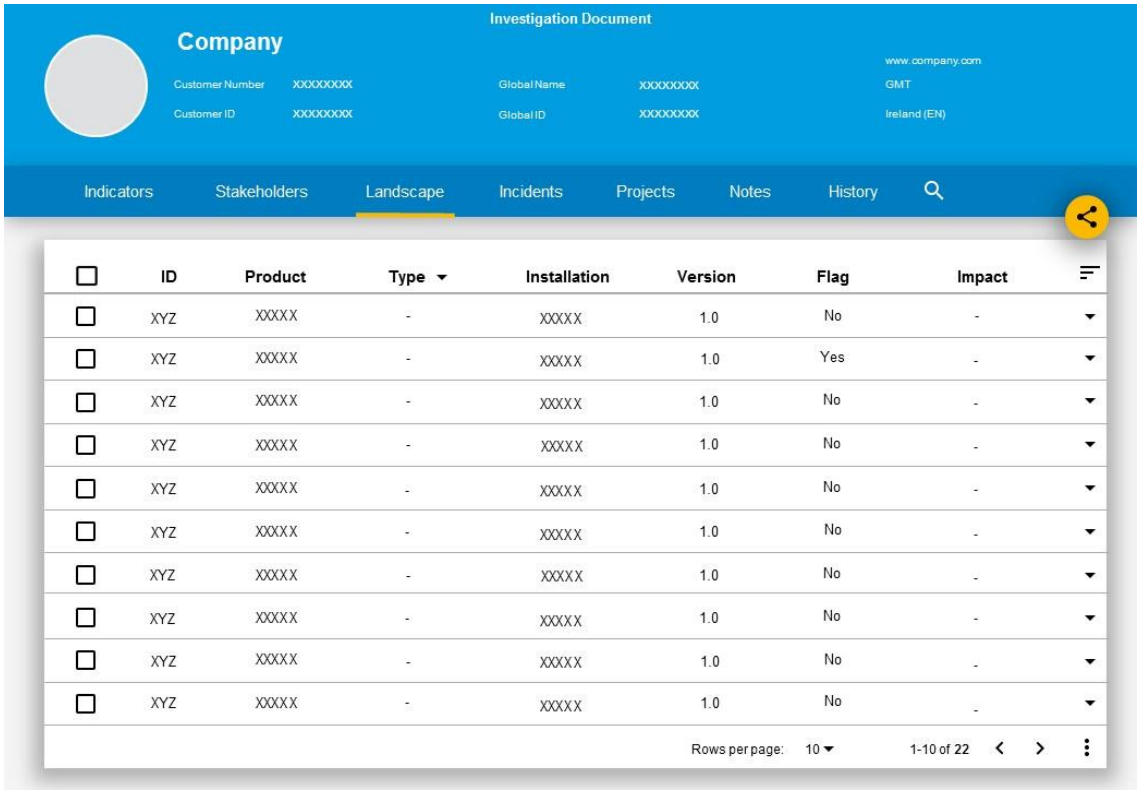


Figura 7. 17 *Wireframe Material Design, Tabla de los sistemas del cliente*

Fuente: Elaboración propia en Photoshop