AGILIZACIÓN DEL PROCESO DE DESARROLLO WEB MEDIANTE SISTEMA DE GESTIÓN DE CONTENIDO PARA EL BACKEND

Eduardo Federico Santillán

Ingeniería en Informática

Facultad de Ingeniería

2016

Índice

INTRODUCCIÓN

Abstract

El presente proyecto pretende desarrollar e implementar herramientas y técnicas necesarias para agilizar el proceso de desarrollo web.

Para esto se analizaron las problemáticas relacionadas al problema, los flujos de trabajos actuales y las herramientas utilizadas para cumplir con los objetivos. A partir de esto se planteó la mejor solución posible que consiste en el desarrollo de un sistema de administración de sitios web y proyectos de desarrollo web en general (normalmente llamado panel de administración), más la definición, documentación e implementación de los procesos para lograr el producto final.

Breve descripción del problema y su importancia

El problema se presenta en una empresa de diseño y desarrollo web. Algunos de los problemas clásicos que se presentan en la empresa son:

- Tiempos que exceden lo planificado.
- Costos que superan lo presupuestado.
- Dificultad para dar mantenimiento a un sistema web ya entregado al cliente. (Principalmente debido al uso de tecnologías inadecuadas, obsoletas y/o sin mantenimiento).
- Cuellos de botella en el proceso de entrega de un sistema.
- Desarrollo de software de manera artesanal y sin procedimientos definidos.

La empresa en cuestión realiza en su mayoría sitios web empresariales, relativamente sencillos. Pero también realiza trabajos de desarrollo más complejos.

Dividiremos estos desarrollos de la siguiente manera a fin de entenderlos mejor:

Figura 1: Tipos de sitios web. Fuente: Elaboración propia.

Sitios Web Empresariales: Entenderemos estos como aquellos sitios web con poca modificación de su contenido en el tiempo y que cuentan muchas veces con secciones clásicas y claramente distinguibles. Ejemplo de secciones: Home, ¿Quienes Somos?, ¿Qué Hacemos?, Contacto.

Sitios Web Complejos: Estos podrán tener una gran variedad de características y muy diversos objetivos, pero en general diremos que son aquellos sitios de mucha actualización de información, mayor cantidad de secciones que los Sitios Web Empresariales y gran cantidad de contenido. También podemos decir que suelen incluir:

- *Login*/Registro.
- Catálogo de productos.

- Carro de productos.
- Posibilidad de pago online.
- Gran cantidad de formularios.
- Menús de navegación complejos.
- Subscripción por distintos medios.
- Funciones para optimizar el SEO *on-site.*

Desarrollos Web a Medida: Esto serán aquellos desarrollos que realizan una o más funciones de negocio dentro de la empresa y que necesitan requerimientos especiales y poco frecuentes. Generalmente estos desarrollos suelen ser sitios para uso en una intranet o de acceso restringido en internet y sin contenido disponible para el público en general. Ejemplo de Desarrollos Web a Medida: gestión de stock, solución online para la gestión de alquileres para una inmobiliaria, gestión de proveedores y clientes.

En el 100% de los casos, sean proyectos sencillos o complejos, siempre se cuenta con un panel de administración para la carga, edición y borrado de datos. Aquí es donde se presenta un cuello de botella, a medida que los requerimientos de un proyecto se complejizan las herramientas actuales con las que se desarrolla se convierten en un impedimento más que una ayuda. Así mismo, se presenta otro cuello de botella cuando las herramientas, técnicas y procedimientos usadas por cada desarrollador en particular impiden el trabajo en equipo y la intervención de distintos desarrolladores en distintos proyectos.

Es necesario:

- Un software de desarrollo que se adapte, de ser posible, al 100% de los casos que la empresa necesita, de interfaz amigable, fácilmente modificable y que suponga una reducción de costos y tiempos de desarrollo para la empresa.
- La definición de procedimientos para las distintas instancias de desarrollo web.

Motivación para abordarlo

La principal motivación para abordar el problema es reducir los tiempos y costos de desarrollo. Pero existen otras cuestiones que impulsan la decisión de solucionar estos problemas.

- Reducir el estrés de los desarrolladores.
- Usar tecnologías “modernas”.
- Cambiar el flujo de trabajo.
- Crear un producto que permita empezar a consolidar una marca para la empresa.
- Mejorar la satisfacción de los clientes.
- Dejar de realizar software de manera artesanal.

Pasos a realizar

Los pasos generales a realizar son:

1. Analizar el problema en cuestión.
2. Investigar soluciones existentes.
3. Definir requisitos específicos.
4. Planificar la solución.
5. Desarrollar.

El proceso de desarrollo se realizará usando algunos conceptos y herramientas de la metodología KANBAN.

Criterios de éxito

El principal criterio de éxito será que una vez terminado todo el proceso y puesta en marcha la solución, esta cumpla con sus objetivos y signifique una mejora en el proceso de desarrollo de la empresa, a la vez que soluciona los problemas iniciales.

ESTADO DE LA CUESTIÓN

Antecedentes relacionados con el problema

En sus inicios la web se conformaba de sitios estáticos, desarrollados usando tecnologías como HTML y CSS[1]. Estos sitios básicamente no modificaban su contenido nunca y se dependía de un desarrollador para lo modificará en caso de que, por algún motivo, se necesitará cambiar un texto o una imagen.

Con el surgimiento de la web 2.0, la popularización de los gestores de contenido, las reglas de posicionamiento en buscadores y las nuevas estrategias de marketing, se empezaron a necesitar nuevas soluciones para brindar a las empresas y creadores de contenido las herramientas necesarias para afrontar nuevos desafíos.

Las soluciones llegaron en forma de:

- **Gestores de contenido avanzados.** Que permiten de manera relativamente sencilla crear una gran diversidad de sitios web con distintas funcionalidades en poco tiempo. Ej: Wordpress, Joomla, Drupal.
- **Paneles que solo se adaptan a una y solo una estructura de sitio web.** Estos permiten funcionalidades específicas y verticales, una tienda online, una wiki, un sistema de gestión de conocimiento, etc. Ej: Magento.
- **Paneles de administración a medida para cada solución.** Estos son desarrollados a medida por empresas de desarrollo web y permiten obtener funcionalidades o cumplir requerimientos que con las otras soluciones no se puede o se vuelve muy complicado.

En cuanto al problema relacionado al proceso nos topamos con la situación de estar realizando desarrollo web de manera totalmente desestructurada, sin guías ni criterios unificados. Esto significa básicamente que no existe un consenso sobre cómo llevar a cabo cada etapa del proceso de desarrollo, las herramientas a utilizar ni las vías de entrega y comunicación que existen sobre cada etapa.

Actualmente entonces nos encontramos en la situación de que al recibir la solicitud de un nuevo trabajo de desarrollo, el cómo y cuándo estará listo este trabajo dependerá exclusivamente de la/s personas/s encargada/s de ese trabajo en particular.

[1]"Historia del Diseño Web - Dweb3d." 2013. <http://www.dweb3d.com/blog/diseno-web/historia-del-diseno-web.html> Página vigente al 6 Feb. 2015.

Soluciones alternativas

Dado que este trabajo pretende crear 2 soluciones, una en forma de software y otra en forma de definición de procedimientos, entonces se presentarán las soluciones para ambos casos por separado.

Soluciones para el software de desarrollo

Aquí nos encontramos con varias opciones. Las que resumimos en estas 4.

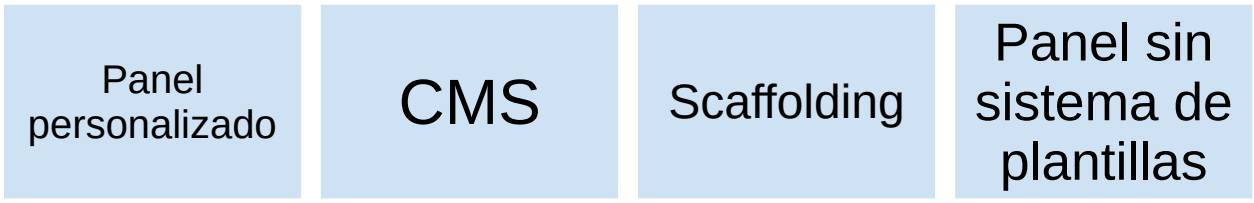

Figura 2: Software de desarrollo. Fuente: Elaboración propia.

Panel personalizado

Esta opción plantea que cada proyecto tenga su panel desarrollado desde cero. Con esto logramos satisfacer las necesidades del cliente de la forma en la que él lo solicita ya que cada requerimiento puede ser cumplido pues no existen otras barreras más allá de las que puede llegar a presentar el lenguaje de programación elegido.

Las principales desventajas de crear un panel para cada proyecto:

- El trabajo requerido es mucho. Los tiempos y costos se extienden más allá de lo que normalmente un cliente está dispuesto a afrontar.
- Los paneles de administración se realizan con diseños distintos y dependiendo de los gustos del cliente o en su defecto del programador que los realice. Esto no ayuda a fortalecer la marca de la empresa de desarrollo pues no hay una interfaz reconocible a simple vista en cada uno de sus trabajos.

CMS

Un CMS es un *Content Management System* o Gestor de contenidos en español. Muchos de los sitios de la web actual están montados sobre uno de estos sistemas. Un gestor de contenidos brinda básicamente 2 partes: un panel de administración de contenido y un sistema de plantillas de diseño.

Ejemplos conocidos de CMS son Drupal, Joomla y Wordpress.

Estos cuentan con miles de extensiones que permiten ampliar sus funcionalidades. Con una extensión podemos tener toda una plataforma de pago, con otra un carro de compras, con otra podemos tener hermosas galerías, o gestión avanzada de usuarios, etc.

Además de poder agregar las funcionalidades a gusto también cuentan con miles de plantillas, también conocidas como *templates* o *themes*.

Esta combinación ha hecho que los CMS sean la opción preferida por la mayoría para publicar en línea un sitio web, pues casi no requieren trabajo y pueden usarse para crear una gran variedad de sitios web.

Las principales desventajas de un CMS:

- No todas las extensiones y *themes* son gratuitos.
- Para un sitio empresarial no siempre se desea tener un diseño ya existente, sino que se busca distinguirse del resto.
- Hay requerimientos que no siempre se logran cumplir con las extensiones a pesar de la gran cantidad que existe.
- Un CMS puede tener más opciones de las que se desea brindar a un cliente por la simple razón de que no podrán entender y manejar todas esas opciones, esto sólo provoca confusiones en aquellos que no conocen nada de tecnologías de la información.

Scaffolding

Un sistema de *scaffolding* permite crear un panel ABM (Alta-Baja-Modificación) en base a una estructura de una BD (Base de Datos) o de algunas de sus tablas. Es una técnica bastante conocida en el mundo del desarrollo de software en general.

Este sistema en principio permite crear rápidamente los formularios para la administración de una tabla, las validaciones para la carga o modificación de datos, mientras simultáneamente nos dan total libertad de modificar absolutamente todo. Pueden ser de gran ayuda para ahorrar tiempo en la creación de la estructura de un administrador de contenidos.

Y con respecto al diseño visual del sitio web también brinda total libertad pues en este caso no hay plantillas, sino que se usará el diseño que se desee.

Aunque en el mundo de desarrollo web también es conocido que algunas empresas usan un sistema de scaffolding para entregar el producto final a un cliente cobrando como si fuera un desarrollo desde cero. No se citan ejemplos para no perjudicar a nadie.

Ejemplo de sistema *scaffolding* para el desarrollo web: Laravel-4-Generators[2]

Las principales desventajas del *Scaffolding*:

- Es de poca ayuda cuando los campos de entrada requieren características fuera de lo común y que el software de *scaffolding* no tiene contempladas.
- Sólo crean los formularios para el administrador pero no un menú para acceder a los distintos listados y/o formularios.
- Generalmente, no tienen ningún diseño para la administración. Suelen ser formularios sobre una página en blanco, justamente para que cada uno personalice la interfaz de administración a gusto, pero esto también quiere decir que existe este trabajo extra.
- Normalmente no brindan ningún sistema de gestión de usuarios, roles y permisos.

Sistemas de administración sin sistema de plantillas

Podemos considerar a esta opción como la intermedia entre un CMS y el *scaffolding*. Un sistema de administración de contenidos sin sistema de plantillas brinda esa libertad necesaria la mayoría de las veces, de crear y usar un diseño a medida para la creación de una web empresarial.

Este sistema normalmente permite adaptarse a la estructura de la BD y cuenta con muchas más opciones que solo formularios con campos de texto. Además viene con un diseño para toda la parte del administrador e incluso un sistema de gestión de usuarios y *login*.

En resumen estos son sistemas cuyo panel para la carga de contenido estará completo, con un diseño agradable, sistema de usuarios, permisos y con muchas opciones para la gestión de los datos.

Principales desventajas de un Sistema de administración de contenidos sin sistema de plantillas:

- No están contempladas todas las opciones de entrada de datos, habrá más o menos dependiendo del software.

Ejemplo: http://administrator.frozennode.com/docs/introduction [3]

[2]"JeffreyWay/Laravel-4-Generators · GitHub." 2013. <https://github.com/JeffreyWay/Laravel-4-Generators> Página vigente al 10 Nov. 2014.

[3]"Laravel Administrator Documentation : Introduction." 2013. <http://administrator.frozennode.com/> Página vigente al 10 Nov. 2014.

Soluciones para el proceso de desarrollo

La cantidad de alternativas aquí son menos y claramente entendibles: seguir haciendo el software sin criterios ni guías o utilizar las mejores prácticas, aceptadas por la comunidad de desarrollo web.

Obviamente hacer software según los criterios independientes de cada desarrollador o grupo de desarrollo de turno tiene muchas desventajas, que se explicarán al detalle en el capítulo “Definición del problema”.

Por otra parte utilizar prácticas aceptadas, estándares y conocidas traerá grandes ventajas. Estas soluciones pueden venir en muchas formas:

- ITIL[4]
- CMMI [5] [6]
- Herramientas y técnicas estándares para las distintas tareas en el proceso de desarrollo de software.

ITIL y CMMI, aunque conceptos distintos, intentan expresar o establecer las reglas que deberían seguirse para lograr brindar un servicio de calidad. La aplicación de estos requiere una gran inversión que la empresa de destino de este trabajo no puede afrontar.

Pero la adopción de mejores prácticas no debe quedar relegada solo a la disposición de un gran presupuesto pues existen metodologías de desarrollo ampliamente conocidas que se pueden adoptar en conjunto con herramientas y técnicas estándares que permitirán dar un paso adelante y obtener ventajas en cuanto a la organización, gestión de recursos, estimaciones, comunicación y resolución de problemas en general.

[4]"ITIL: ¿qué es y para qué sirve? (parte 1) | Magazcitum. <http://www.magazcitum.com.mx/?p=50>. Página vigente al 4 Ago. 2016.

[5]"Información general de CMMI - MSDN - Microsoft." 2015. <https://msdn.microsoft.com/es-es/library/ee461556.aspx>. Página vigente al 7 Ago. 2016.

[6]"Qué significa CMMI: Qué es CMMI." 2013. <http://asprotech.blogspot.com.ar/2013/10/que-es-cmmi.html>. Página vigente al 7 Ago. 2016.

Herramientas y técnicas requeridas para la solución

La solución se decidió implementar usando PHP como lenguaje de programación y MySQL como sistema gestor de base de datos.

La razón es que PHP es ampliamente conocido no sólo por todos los desarrolladores involucrados, sino también por la comunidad de desarrollo web, recordemos que la empresa suele trabajar mucho con desarrolladores *freelance*. Mysql por su parte es igual de conocido, es sencillo de usar por desarrolladores de todos los niveles y la razón más importante es que ambas tecnologías suelen ser las predeterminadas en cualquier *hosting web* actual.

Se eligió como *framework* de trabajo a Laravel por ser uno de los más extendidos y usados en la actualidad. Cuenta con una completa documentación disponible de manera online y una gran comunidad que da soporte a los desarrolladores desde foros, redes sociales y otros sistemas de mensajería.

Además este framework de desarrollo cuenta con utilidades que brindan grandes ventajas como:

- Gestor de paquetes Composer que permite rápidamente ampliar las funcionalidades del framework. Algunos de estos paquetes pueden ser realmente interesantes para el proyecto.
- Sistemas de plantillas Blade para mejorar la integración entre PHP y HTML.
- Migraciones para la BD. Son una excelente forma de registrar cambios en la estructura de la BD y sus tablas sin hacerlo desde una herramienta como PhpMyAdmin o desde la consola de MySQL.

A su vez se usará GIT (sistema de versionado de código distribuido) para registrar todo el proceso de desarrollo de creación de un sitio web y permitir a varios desarrolladores trabajar simultáneamente en los proyectos de forma eficiente y con la menor cantidad de contratiempos posibles.

Se usará también el servicio de Bitbucket, un repositorio de código que ofrece alojamiento gratuito para proyectos privados.

Para el versionado de código se usará GIT por ser la herramienta estándar en la actualidad para esta tarea.

Y se ha probado por el equipo de desarrollo que es muy eficiente usar un gestor de tareas y evitar enviar tareas via email. En este caso se decidió usar el servicio gratuito de Asana como sistema general para la gestión de tareas. Asana es una herramienta de muy fácil adaptación a casi cualquier metodología de desarrollo.

Temas pendiente de resolución

El presente proyecto no pretende cubrir todos los aspectos del desarrollo de la solución, dejando de lado temas que están fuera del alcance académico pretendido. Algunos de estos temas son:

- Configuración de servidores para el correcto funcionamiento del software desarrollado.
- Configuración de hardware en general necesarios para la conexión a servidores.
- Migración de sistemas antiguos.

DEFINICIÓN DEL PROBLEMA

Antes de entender el problema es importante entender el flujo de trabajo que se lleva a cabo cada día en la empresa.

Flujos de trabajo en la empresa

La empresa recibe varias solicitudes para diseño y desarrollo web. Podemos enmarcarlas en 3 tipos:

- Sitios web empresariales.
- Desarrollos complejos.
- Implementación y adaptación de CMS.

Cada uno de estos tipos responde a un flujo de trabajo diferente.

Sitios web empresariales

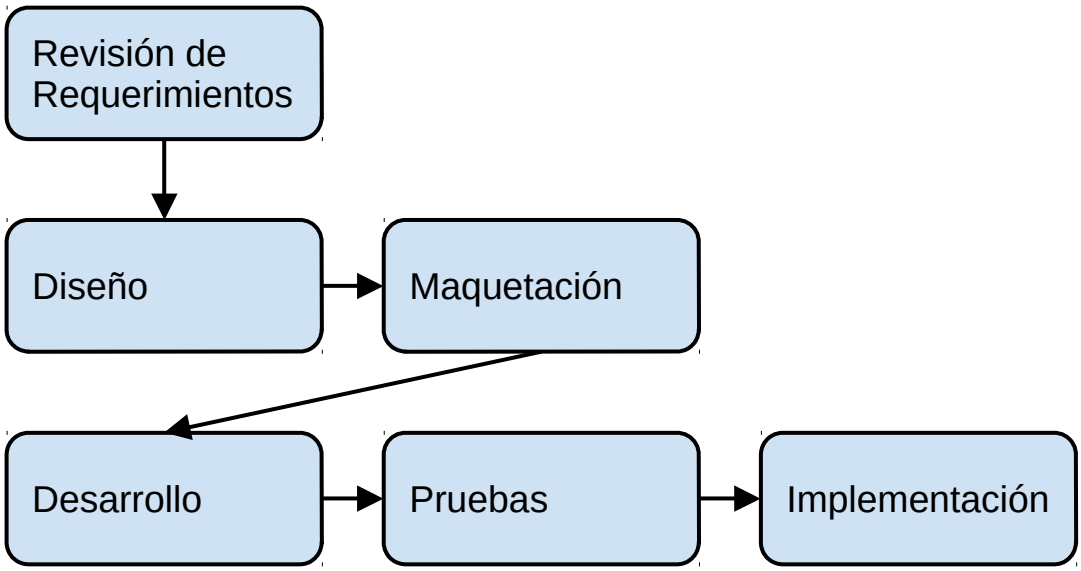

Figura 3: Flujo de sitios empresariales. Fuente: Elaboración propia.

En este tipo de solicitudes primero se revisan los requisitos para encontrar los detalles que van a requerir más trabajo y establecer si en ese caso se debería enmarcar en alguno de los otros tipos de solicitudes.

En la etapa de diseño un diseñador gráfico crea un diseño en Photoshop que es enviado en formato .psd al maquetador. En esta etapa el diseñador se encuentra en constante comunicación con el cliente para llegar a acuerdos sobre el diseño de interfaz y comportamiento de la misma.

En la etapa de maquetación un maquetador web convierte el diseño en un sitio web estático usando únicamente HTML + CSS + JS

En la etapa de desarrollo un desarrollador web crea el panel de administración y adapta la maqueta para que ésta sea dinámica y responda a los cambios realizados en el panel de administración.

La etapa de pruebas (cuando se realiza) generalmente consiste en una reunión entre el diseñador y el desarrollador para la revisión del producto final y definir los detalles que falten para cumplir los requisitos.

La implementación es generalmente un proceso sencillo que consiste únicamente en mudar los archivos y la BD del servidor de pruebas al servidor de producción. En raras ocasiones se necesita también realizar algún pequeño trabajo extra como migración inicial de unos pocos textos o imágenes.

Desarrollos complejos

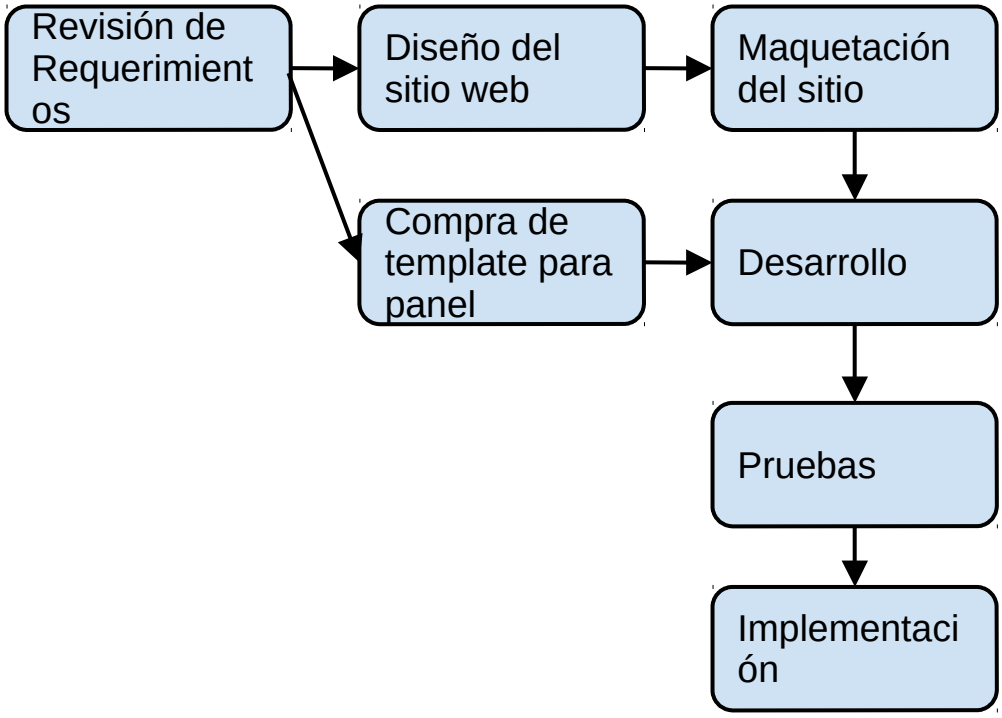

Figura 4: Flujo de desarrollos complejos. Fuente: elaboración propia

Desarrollos complejos son aquellos que requieren mucho más que formularios con campos tradicionales o que tienen relaciones complejas entre las tablas.

Algunas veces estos desarrollos son sitios web y otras veces son solo sistemas internos sin una interfaz abierta o accesible al público en general.

Para estos casos se compra un *template* para la parte de administración mientras que para el sitio web, si se requiere, se hace un diseño con maquetación a medida. Puede incluso darse el caso de que se requiera un diseño y maquetación a medida para el panel de administración, pero es una situación poco frecuente.

Por supuesto para estos tipos de soluciones la etapa de desarrollo es mucho más larga que para un sitio web empresarial.

Implementación y adaptación de CMS

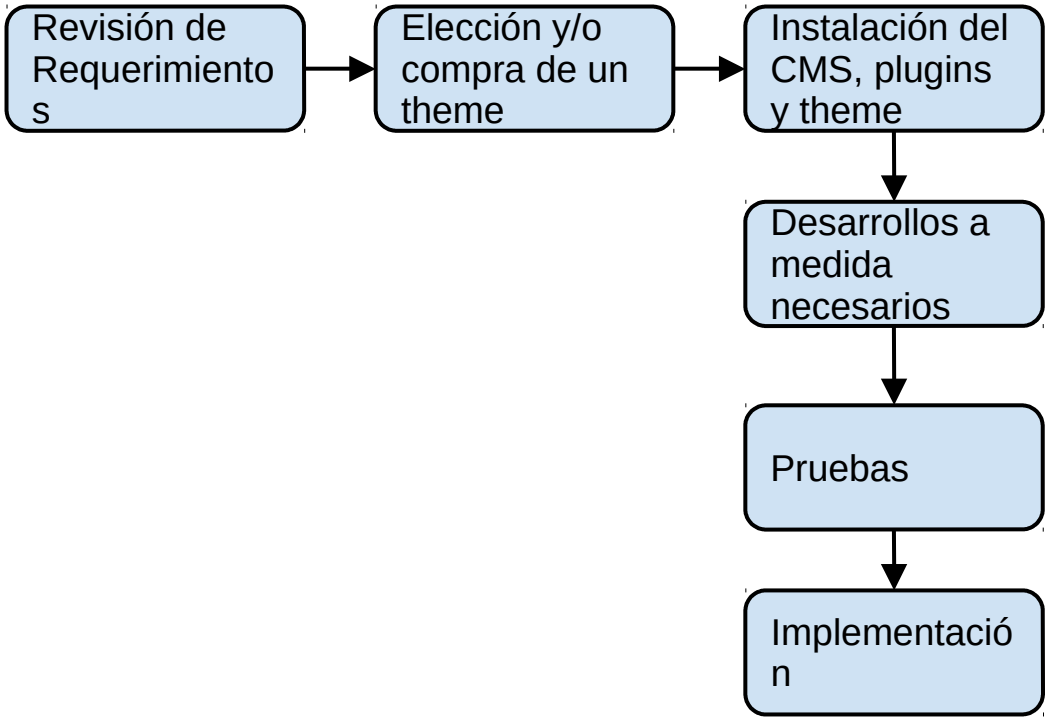

Figura 5: Flujo de CMS. Fuente: Elaboración propia.

En ocasiones se solicita la instalación de un CMS por variados motivos. En estos casos se compra un *theme* o se utiliza alguno gratuito. Se instala, configura y de ser necesario se hacen desarrollos a medida para cumplir con requerimientos que no puedan ser cumplidos usando extensiones.

Cabe aclarar que estos desarrollos son siempre cambios menores, al *core* (núcleo) del CMS, a los *plugins* o al *theme*.

Definición de problema

En la última situación no se presenta prácticamente ningún problema porque los CMS como Wordpress[7] solo se usan cuando es un requerimiento específico del cliente y por lo general este conoce las restricciones de usar un CMS, limitándose a usar solo lo que se pueda lograr con *plugins*.

El problema se presenta en las situaciones primera y segunda. Cuando el desarrollador está más involucrado en el proceso y requiere más tiempo.

El problema existe en la etapa de desarrollo. En especial cuando se solicitan soluciones complejas o cuando algún requerimiento en un sitio web empresarial sale de lo común.

Al definir los problemas que el presente trabajo pretende solventar en la empresa nos encontramos con dos situacione críticas:

- El software usado para desarrollar es obsoleto, ineficiente e inadecuado.
- No existen definiciones sobre la forma de trabajar y de llevar a cabo las tareas comunes para el desarrollo de software.

El software utilizado para el desarrollo

Para el desarrollo de sitios web empresariales, y en ocasiones para los desarrollos más complejos, se utiliza un sistema “Panel sin sistema de plantillas” como el descrito en la sección Soluciones alternativas.

Este software permite crear un panel de administración de contenidos siguiendo una serie de pasos:

1. Crear las tablas manualmente en la BD.
2. Configurar las distintas secciones que tendrá el sitio web desde la BD.
3. Modificar no menos de 3 archivos de configuración del propio software.

Este software tiene las siguientes desventajas y problemas:

- Está desarrollado sobre un framework que pocos conocen y por lo tanto pocos saben utilizar.
- Este *framework* a la vez tiene una documentación escasa y una comunidad muy pequeña, por lo que es difícil encontrar soporte.
- El *framework* es mantenido por muy pocas personas y por lo tanto es difícil que se solucionen los problemas del núcleo del mismo.

[7]"WordPress › Blog Tool, Publishing Platform, and CMS." 2004. <https://wordpress.org/>. Página vigente al 10 Nov. 2014.

- Este software fue desarrollado por personas que ya no trabajan para la empresa.
- A su vez fue desarrollado sin guías de estilo y sin seguir ninguna convención sobre la forma de desarrollarlo o los componentes a utilizar.
- Se crearon componentes para este software sin dejar documentación sobre cómo usarlos o como fueron desarrollados por lo que es muy difícil hacer modificaciones sobre esos componentes.
- El desarrollo se realizó sin usar ningún sistema de versionado de código por lo que es imposible saber quien realizo cada cambio, cuando y con qué propósito.
- Este software dejará de funcionar a partir del momento en que los *hostings* actualicen PHP a la última versión estable puesto que algunos componentes del software usan funciones obsoletas (*deprecated*) de PHP y por los puntos planteados anteriormente es muy difícil actualizarlos.

Proceso de desarrollo

En cuanto a la definición de las tareas a realizar en el proceso de desarrollo podemos decir que son completamente nulas. Las tareas se realizan según el criterio del desarrollador o grupo de desarrollo encargado de cada trabajo particular.

Estas son algunas de las tareas ausentes, sin definir o mal implementadas en la empresa:

- No existe una metodología de desarrollo a seguir.
- No se usa ningún sistema ni modelo de versionado de software.
- No existe sistema de *backups* para los sitios entregados.
- No existe un modelo para la implementación y puesta en marcha de los trabajos realizados.
- No se realiza planificación para las pruebas, no hay casos de prueba ni se realizan pruebas formales sobre los trabajos encargados.
- No existe una forma sistemática de planificar el tiempo y por lo tanto costos de desarrollo.
- No hay ninguna guía o manual para dar inicio al proceso de desarrollo que incluya los pasos para la instalación del software a usar, ni como configurarlo o hacerle modificaciones.
- La forma de trabajo de la empresa hace muy difícil aceptar y llevar a cabo solicitudes que se produzcan en el medio de desarrollo y incluso más difícil es cuando se producen al finalizar el mismo.

Objetivo

Agilizar el proceso de desarrollo web adoptando prácticas estándares para el mismo, más el desarrollo de un sistema de administración de contenido que sea sencillo implementar y configurar, para incluir en el proceso a desarrolladores *juniors* o *freelance*.

Objetivos específicos

Crear un gestor de contenidos que permita solventar las situaciones que se presentan en el campo de desarrollo web en la empresa.

Permitir que los programadores juniors se integren fácilmente y con una curva de aprendizaje rápida en los proyectos de desarrollo.

Integrar en el sistema de gestor de contenidos características actuales y necesarias como url amigables, administración flexible de imágenes y archivos.

Definir y hacer uso de herramientas y prácticas estándares que permitan agilizar el desarrollo, desde el *setup* de un nuevo proyecto hasta la puesta en producción de un sitio.

Alcance

La solución propuesta deberá contemplar los siguientes elementos:

- Un sistema que permita la creación de un panel de administración para el desarrollo de sitios empresariales (sencillos) y desarrollos más complejos también.
- Un proceso de desarrollo que abarque desde la maquetación hasta el despliegue de código en servidores de producción. Este proceso debe permitir a los desarrolladores *juniors* integrarse en el desarrollo de paneles de administración complejos en los que antes no podían participar por la dificultad de los mismos.
- Cambiar herramientas antiguas por herramientas más actuales, con mayor comunidad y aceptación. Además incluir en el proceso de desarrollo aquellas técnicas y herramientas modernas no utilizadas.
- Documentación detallada que agilice y facilite el proceso de preparación de desarrolladores juniors para el trabajo en la empresa.

Restricciones o límites del trabajo

El trabajo presentado no contempla:

- El desarrollo de herramientas y utilidades para la parte pública de los sitios desarrollados para los clientes.
- Documentación para configuración de servidores.
- Documentación para configuración de sistemas gestores de bases de datos.
- Sistema de extensiones o API para ampliar el sistema por parte de terceros.
- Comunicación con servicios externos (API, *web services*, *SOAP services*, RSS, etc).
- Sistema de administración gráfica o tipo *drag&drop* para la creación de paneles de administración.

SOLUCIÓN PROPUESTA

La solución que se propone es la creación de un sistema de gestión de contenidos sin administración de plantillas.

Además se propone un cambio en el flujo de trabajo integrando en el mismo técnicas y herramientas estándares y aceptadas en la comunidad de desarrollo web, priorizando su definición de forma clara y simple pero completa, de manera tal que pueda ser aprovechada por desarrolladores de todos los niveles.

Todo esto acompañado de una documentación clara que permita cumplir con los requerimientos y que ayude a alcanzar los objetivos planteados.

A continuación se especifican detalles sobre las 2 partes de la solución: el gestor de contenidos y la adopción de herramientas y técnicas estándares para el proceso de desarrollo.

Desarrollo de un sistema de gestión de contenidos sin administración de plantillas

El software propuesto deberá responder a las siguientes condiciones:

- Ser de fácil instalación.
- Tener una documentación completa.
- Ser fácilmente modificable y adaptable a nuevas características.
- Permitir una configuración sencilla con un máximo de un archivo de configuración.
- Poder instalarse en los *hostings* habituales con los que trabaja la empresa.
- Permitir actualizar de manera sencilla el software.
- Tener una interfaz gráfica moderna y amigable.

El desarrollo del software seguirá algunos procedimientos y conceptos de la metodología KANBAN. Ver anexo KANBAN para más detalles.

Mantenimiento del sistema de gestión de contenidos

Se prevé la realización de un mantenimiento preventivo al software cada 10 meses. Periodo que se tomó en consideración como una fecha razonable en la cual los principales *frameworks* usados para su desarrollo alcanzan nuevas versiones estables. Ej: jquery[8], laravel[9], bootstrap[10].

Este mantenimiento tiene como objetivos:

- La actualización de los *frameworks*, librerías y archivos que componen este software.
- La prueba sobre entornos actuales (sistema operativo del *hosting*, versión de mysql, versión de php, etc).
- La prueba de la interfaz sobre navegadores estándares modernos.

[8]"jQuery Release Notes · GitHub - Gist." 2014. <https://gist.github.com/teppeis/9264080>. 17 Ago. 2016.

[9]"Releases · laravel/laravel · GitHub." 2013. <https://github.com/laravel/laravel/releases>. 18 Ago. 2016.

[10]"Tags - GitHub." 2013. <https://github.com/twbs/bootstrap/tags>. 18 Ago. 2016.

Determinación de las mejores prácticas a aplicar

Como parte de la solución se propone un cambio fundamental: la adopción de algunas de las mejores prácticas en el proceso de desarrollo. Las prácticas a adoptar se han determinado en base a la interpretación de los problemas que se presentan en la etapa de desarrollo y la existencia o no de una solución a esos problemas que la empresa esté dispuesta a implementar como parte del proceso de desarrollo habitual.

A los efectos de que el lector pueda visualizar los diferentes problemas que se suscitan en la operación diaria de la empresa y seguir una línea en su lectura, la solución y el detalle de propuesta para cada uno de ellos se especifican en cada Anexo.

Problema: No existe una metodología de desarrollo a seguir.

Solución: Se propone el uso de los principios básicos de la metodología Kanban. Esta metodología es flexible, fácil de entender y aplicar, y con características que la hacen perfecta para la empresa en cuestión. En el Anexo, sección Kanban, se especifican los aspectos básicos de esta metodología, la razón para elegirla y la forma de aplicarla en la empresa (Ver Anexo, sección Kanban).

Problema: No existe una forma sistemática de planificar el tiempo y por lo tanto costos de desarrollo.

Solución: Se propone la realización de una tabla que sirva para cálculo de tiempos de desarrollo del panel usando el nuevo software. Misma que deberá ser ajustada con la práctica y a medida que se soliciten nuevos trabajos. (Ver anexo, sección Tabla de tiempos de desarrollo).

Problema: No hay ninguna guía o manual para dar inicio al proceso de desarrollo que incluya los pasos para la instalación del software a usar, ni como configurarlo o hacerle modificaciones.

Solución: Se propone la creación de un manual completo, que sirva como base para cualquier desarrollador de cualquier nivel en el que se expongan todos los pasos necesarios, desde la instalación del software de desarrollo, su configuración, modificación y expansión. Se propone la creación de un manual de uso del nuevo software. (Ver Manual de uso de Software de desarrollo).

Problema: No se usa ningún sistema ni modelo de versionado de software.

Solución: Se propone el uso del modelo de nvie en distintas formas según la complejidad del proyecto. (Ver Anexo, sección Versionado de Software).

Problema: No existe sistema de *backups* para los sitios entregados.

Solución: Se propone el uso de un sistema de *backups* automáticos usando un servicio en la nube como Dropbox. (Ver anexo, sección Backups de proyectos).

Problema: No existe un modelo para la implementación y puesta en marcha de los trabajos realizados.

Solución: Se propone la definición de un flujo de *deploy* que use parte del modelo propuesto por nvie, que optimice tiempos y disminuya la cantidad de problemas causados por la actual forma de realizar *deploys* en la empresa. (Ver anexo, sección Flujo de deploy).

Problema: No se realiza planificación para las pruebas, no hay casos de prueba ni se realizan pruebas formales sobre los trabajos encargados.

Solución: Se propone la incorporación de pruebas unitarias al proceso de desarrollo sobre los trabajos que se solicitan. Añadiendo la planificación de estas pruebas como parte del proceso habitual del desarrollo de software. (Ver anexo, sección Pruebas).

Problema: La forma de trabajo de la empresa hace muy difícil aceptar y llevar a cabo solicitudes que se produzcan en el medio del desarrollo y incluso más difícil es cuando se producen al finalizar el mismo.

Solución: La adopción y entendimiento de la metodología Kanban servirá de soporte para la resolución de esta problemática. (Ver Anexo, sección Kanban).

Justificación de la solución

La solución propuesta se ha elegido en primer lugar porque en la empresa no existe un proceso de desarrollo formalizado a seguir y básicamente se desarrolla cada trabajo nuevo en base a los deseos y experiencia personal de cada desarrollador individual, incluidos los desarrolladores novatos y los *freelance*.

Resulta importante para la empresa poder contar con desarrolladores temporales y desarrolladores *juniors* cuyo conocimiento no es completo, pero sin que esto suponga retrasos en las entregas de trabajos. Tener manuales y guías para cada etapa del trabajo resulta entonces en un beneficio muy grande.

Esto va a permitir 2 aspectos fundamentales:

1. Que todos los involucrados en un proyecto puedan avanzar de forma eficiente desde el inicio del proyecto hasta su entrega.
2. Que el mantenimiento y modificaciones posteriores sean igual de eficientes sin importar quien lo realice o cuando se realicen.

Sumado a este gran cambio está el desarrollo de una nueva herramienta (gestor de contenidos). El hecho de hacerlo internamente partiendo desde cero no es una decisión tomada al azar, sino la elección de la mejor de las alternativas, la menos costosa y la más eficaz para conseguir los objetivos planteados.

Desarrollar una herramienta interna permitirá, mientras se tiene en cuenta el futuro de la empresa y considerando los problemas aprendidos de experiencias pasadas, agilizar el proceso de desarrollo en gran medida, evitando la problemática planteada con respecto al software de desarrollo en apartados anteriores. Esto significa saber cómo se hizo, quien, cuando y porque se realizaron cambios en el software y buscar con su desarrollo lograr usabilidad a largo plazo.

Existen varias razones para encarar este desarrollo desde cero, pero teniendo en cuenta las alternativas se resaltan 2 razones principales:

1. No se podía mejorar y/o continuar con el desarrollo del panel actual, puesto que no se tienen registro de su desarrollo, o guia de estilo aplicada al código; se desarrolló de manera independiente y sin comunicación por varias personas a lo largo del tiempo. Además de que los cambios y actualizaciones en las versiones de los *plugins* de jQuery o librerías de PHP provocan muchas demoras y problemas, a cambio de pocas ventajas.
2. Usar un CMS de desarrollo como FrogCMS, PyroCMS o Wordpress como plataforma de gestión de contenidos requiere invertir tiempo en aprender como se ha desarrollado y en la generación de extensiones que satisfagan las necesidades de la empresa. Sumado a muchas otras desventajas como: imposibilidad de establecer un valor como marca usando herramientas de terceros, alta dependencia a la continuidad del mantenimiento del "*core*" de estos sistemas y sus extensiones, etc.

Formulación de proyecto

Alternativas tecnológicas de solución al problema

El presente proyecto plant ea el desarrollo de la solución de software usando tecnologías de desarrollo web como PHP, CSS, SaSS, Javascript, HTML5 y derivados. Pero existen otras alternativas tecnológicas que podrían haberse elegido y que estuvieron en discusión por parte de los involucrados. Estas alternativas fueron:

- Python, Django, CSS, Javascript
- Ruby, Ruby on Rails, CSS, Javascript
- NodeJS, bases de datos no relacionales
- Otros *frameworks* PHP

Lenguajes distintos a PHP se descartaron por los siguientes motivos:

- Los desarrolladores actuales y los freelance que se suelen contratar no conocen todos los mismos lenguajes excepto por PHP.
- No todos los proveedores de *hosting* soportan esas tecnologías.

Y la elección de los frameworks de PHP se hizo realizando una investigación detallada de las opciones. Teniendo en cuenta empresas patrocinadoras, tamaño de la comunidad, documentación y tutoriales disponibles, etc. Una de las herramientas que se usó para la comparación fue OpenHub[11] que recopila varios de estos datos y permite comparar proyectos de software libre.

[11]"Open Hub, the open source network." 2014. <https://www.openhub.net/>. Página vigente al 11 Nov. 2014.

Análisis Económico-Financiero

Se identifican 4 principales fuentes de retorno de inversión:

Ganancias directas por el uso del nuevo software: El software propuesto permitirá desarrollar sitios web y soluciones web más eficaz y eficientemente. Esto se traduce como un menor costo variable. Además, teniendo en cuenta que las características del nuevo software brinda a los clientes un valor agregado en calidad, tiempos y seguridad, este se puede traducir en mayores ingresos por proyecto para la empresa, al aumentar el presupuesto de los mismos.

Agilización del proceso de desarrollo: Más allá del uso de una solución eficiente en materia de software, la definición y adopción de prácticas estándares permitirá agilizar todas las etapas de desarrollo. Menos tiempo de desarrollo se traducen en más proyectos que se pueden aceptar.

Incremento en la cantidad de clientes: El producto de software ayuda a posicionar la marca de la empresa, pues ahora se convertirá en un estándar para los productos que desarrolla y entrega la empresa. Esto ayuda a que los clientes la reconozcan y por lo tanto, y de acuerdo a su conformidad, se genera una publicidad gratuita y de la mayor calidad posible como es el “boca a boca”.

Y a partir de la puesta en marcha de toda la solución se logra atender a más clientes en menos tiempo, evitando los cuellos de botellas que impedían el constante flujo de ingresos.

Ganancias internas: Se logra la participación de los desarrolladores *juniors* en soluciones que antes eran muy complejas para ellos, permitiéndoles invertir su tiempo en tareas productivas. Se logra reducir la tensión y estrés por requerimientos que salían de lo normal. Y por último se obtiene un incremento en la productividad, producto de la satisfacción de desarrollar usando algo que se sabe de antemano que puede ser modificado porque fue planeado para ser modificado.

VERIFICACIÓN EXPERIMENTAL

Para verificar la veracidad de lo expuesto en este trabajo se hicieron 2 experimentos separados, uno para probar el prototipo del software de desarrollo y otro para implementar las mejores prácticas al proceso de desarrollo expuestas en este trabajo y detalladas en el Anexo.

Experimentación con el software de desarrollo

Para este experimento se tomó como base el desarrollo de un panel de administración para un sitio web realizado anteriormente en la empresa con las técnicas que venía usando hasta ahora y cuyos tiempos de desarrollo se conocen. Se replicó la creación de este panel pero usando ahora el nuevo Software de desarrollo.

El panel en cuestión pertenece al sitio http://www.baobabestudio.com.ar

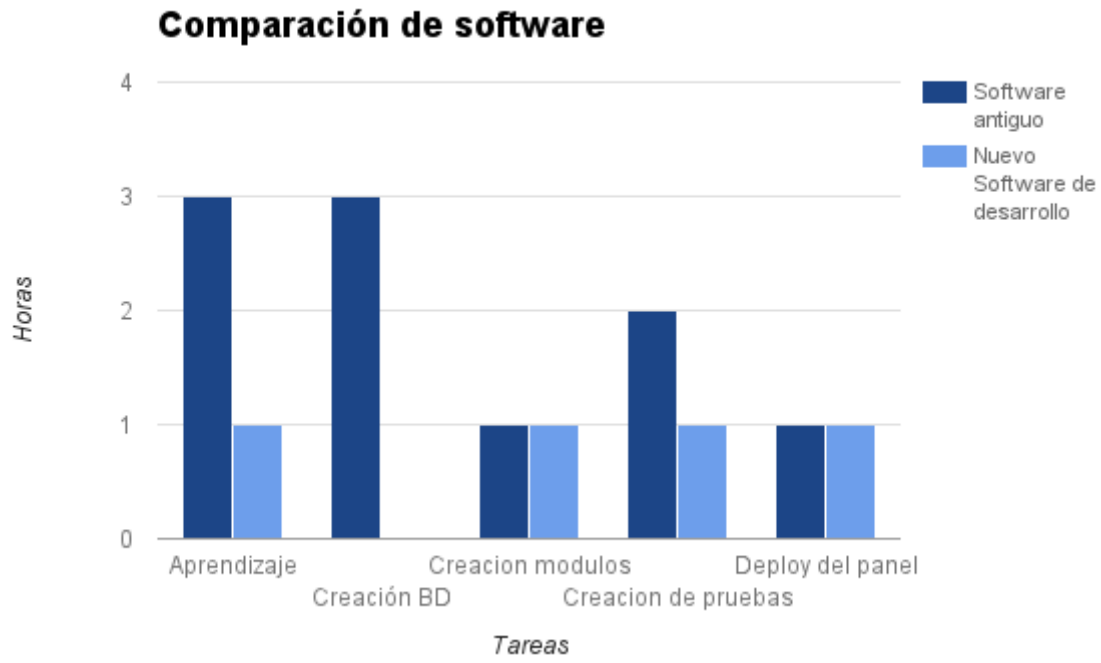

Figura 6: Comparación de software. Fuente: Elaboración propia.

En este caso no se tienen los datos del tiempo que llevo hacer las pruebas unitarias para el software antiguo porque simplemente no se hicieron. Pero se ha estimado en 2hs dado que, aunque es un panel relativamente sencillo, el software que se usó para desarrollar no viene preparado y con todas las comodidades que se encuentran en el framework Laravel para realizar las pruebas unitarias.

Tenemos entonces el resultado expresado en horas para cada software:

Para el software antiguo: 10 horas

Para el software nuevo: 4 horas

Y las ventajas no quedan solo en una reducción de horas sino que además se ven reflejadas en muchas mejoras indirectas como ya se ha expuesto a lo largo de todo este trabajo.

Experimentación con el proceso de desarrollo

Para probar los resultados que se obtendrían al implementar las recomendaciones expuestas en este trabajo realizó una videoconferencia de 2hs con todos los involucrados donde se explicó las desventajas de la forma de trabajo actual, cual es la forma de trabajo

propuesta, cuáles son sus ventajas y cómo se implementarían en la empresa. A su vez se enviaron los anexos donde se exponen estos cambios de manera más detallada.

Finalizado un tiempo prudente de espera para la lectura del mismo, que en este caso fue de 3 días, se procedió a realizar un experimento, poniendo en práctica durante una semana los cambios propuestos.

Ante la dificultad de obtener resultados cuantitativos para este experimento se procedió a realizar una simple reunión, por videoconferencia nuevamente, para recibir la valoración de esta nueva forma de trabajo. El resultado fue una crítica positiva en el 100% de los casos, ánimos para seguir con este método y sugerencias para mejorarlo.

Los involucrados esa semana fueron 2 diseñadores gráficos, 2 desarrolladores senior y 1 desarrollador freelance.

CONCLUSIONES

El presente proyecto pretende “poner en orden” una empresa de desarrollo web que desde hace años funciona como una empresa artesanal, adoptando, en primer lugar, una metodología de gestión y otras buenas prácticas, y en segundo lugar creando una herramienta (el nuevo Software de Desarrollo) que elimine una gran cantidad de problemas que solo derivan en cuellos de botella.

Se considera que con este proyecto se ha logrado y que se han cumplido los objetivos planteados.

FUTURAS LÍNEAS DE INVESTIGACIÓN

Lo expuesto con este trabajo se considera que puede ser implementado en cualquier empresa de desarrollo web que se encuentre en una situación similar. Pero el proyecto puede ser ampliado para cubrir aún más necesidades y agregar nuevas características. A continuación se listan algunas de estas posibles líneas de avance.

- Definición de procedimientos que abarquen tareas habituales de una empresa de desarrollo web desde un punto de vista más amplio. Por ejemplo: gestión de clientes, negociaciones, definición de requerimientos, pruebas más exhaustivas, etc.
- Creación de interfaces gráficas para la creación del panel de administración. Por ejemplo: un sistema *drag&drop* para la creación del archivo de configuración.
- Adición de un sistema generador de API Rest para los módulos creados por el Software de Desarrollo.
- Creación automática de aplicación para *smartphones* utilizando herramientas como Ionic aprovechando la característica *‘responsive’* de la interfaz del panel de administración.

GLOSARIO

API: Interfaz de Programación de Aplicaciones. Capa de abstracción compuesta por funciones y procedimientos diseñados para ser utilizados por otros programas.

Asana: Servicio *online* para ser utilizado como un gestor de tareas y de fácil adaptación a casi cualquier metodología de desarrollo de software.

Backend: En desarrollo web hace referencia a la parte de un sitio web a la que solo tienen acceso los administradores y no los usuarios finales y que por lo general sirve para administrar el sitio en cuestión.

Bitbucket: Servicio *online* de almacenamiento para proyectos que utilizan el sistema de control de versiones GIT.

Blade: Motor de plantillas para el *framework* Laravel.

Bootstrap: *Framework* de CSS y JS para la maquetación rápida de sitios web.

Clonar: Término utilizado para referirse al acto de bajar una copia de un repositorio remoto de GIT.

CMS: Gestor de contenido. Sistemas para gestionar los contenidos (textos, imágenes, archivos, etc) de un sitio web.

Composer: Gestor de dependencias para proyectos de PHP. Permite instalar, desinstalar, actualizar paquetes y librerías para ser usadas en un proyecto de PHP.

CSS: Lenguaje de hojas de estilo en cascada. Se utiliza para mejorar visualmente la presentación visual de sitios web.

Deploy: Desplegar el software. Normalmente se entiende a esto como al acto por subir y poner online un sitio web.

Deprecated: Hace referencia a una función o librería obsoleta. Es decir, que ya no se puede usar o que al menos se recomienda altamente no seguir usando.

Desarrollador freelance: Desarrollador temporal. Normalmente trabaja en un o varios proyectos.

Desarrollador junior: Desarrollador novato. Puede incluso ser un desarrollador freelance a la vez.

Dropbox: Servicio de alojamiento de archivos en la nube.

Feature: Característica. Cuando se habla de ramas de GIT, la rama feature es una rama temporal para trabajar sobre una característica particular del software.

Framework: Marco de trabajo normalmente compuesto de un conjunto de librerías, funciones auxiliares, herramientas y utilidades que facilitan el trabajo sobre un determinado lenguaje de programación.

FrogCMS: Es un CMS.

FTP: Protocolo para transferencia de archivos. En desarrollo web es usado para transferir archivos entre la pc de desarrollo y el servidor de producción.

GTD: Es un método de gestión de tareas personales diseñado para optimizar el tiempo productivo de una persona.

HTML: Lenguaje de etiquetas que constituye la base del desarrollo web.

Ionic: Framework para JS que permite escribir código en HTML, CSS y JS y crear a partir de ello aplicaciones para *smartphones*.

jQuery: Librería para JS que añade una gran cantidad de utilidades para facilitar el desarrollo web. También es la base sobre la que se construyen muchos plugins para JS.

JS: Abreviación de JavaScript. Lenguaje de programación que se ejecuta en los navegadores de una pc.

Laravel: Framework para desarrollar con PHP.

Maqueta: Versión estática de un sitio web desarrollada usando solo HTML, CSS y JS. Es la primera versión web de un diseño realizado con herramientas gráficas como Photoshop.

Maquetación: Proceso de codificar una maqueta a partir de un diseño normalmente realizado en Photoshop.

Migraciones: Herramienta del framework Laravel que permite tener un sistema de control de versiones para la base de datos.

PhpMyAdmin: Herramienta desarrollada en PHP para la administración de base de datos MySQL.

PHPUnit: Framework de PHP diseñado para realizar pruebas sobre el código.

PyroCMS: Es un CMS.

Repositorio local: Copia local del código de un software que se sincroniza de alguna manera con el repositorio remoto usando algún sistema de control de versiones como GIT.

Repositorio remoto: Copia en un servidor del código de un software que se sincroniza de alguna manera con el repositorio local usando algún sistema de control de versiones como GIT:

Responsive: Hace referencia a la característica de un sitio web de poder adaptarse a distintos tamaños de pantallas como pueden ser: la de pc, la de tablet, la de un smartphone.

RSS: Formato basado en XML para compartir contenidos web.

SaSS: Preprocesador para CSS. Permite escribir código más limpio añadiendo nuevas características a CSS.

Scaffolding: Creación automática de formularios y tablas para la interacción con la base de datos.

Seeders: Funcionalidad del framework Laravel que permite crear clases para alimentar la base de datos con datos de prueba.

SEO: Optimización para buscadores.

SEO on-site: Optimización para buscadores que se realiza haciendo modificaciones a un sitio web.

Setup: Instalación y configuración.

SOAP: Protocolo que define cómo pueden comunicarse 2 objetos mediante el intercambio de datos en formato XML.

SSH: Protocolo para acceder a máquinas remotas de forma segura.

Template: Conjunto de archivos que sirven como base de diseño gráfico para un sitio web.

Themes: Template.

Wordpress: Es un CMS. El más popular de todos.

BIBLIOGRAFÍA

"Historia del Diseño Web - Dweb3d." 2013. <http://www.dweb3d.com/blog/diseno-web/historia-del-diseno-web.html>. Página vigente al 6 Feb. 2015.

"JeffreyWay/Laravel-4-Generators · GitHub." 2013. <https://github.com/JeffreyWay/Laravel-4-Generators>. Página vigente al 10 Nov. 2014.

"Laravel Administrator Documentation : Introduction." 2013. <http://administrator.frozennode.com/>. Página vigente al 10 Nov. 2014.

"ITIL: ¿qué es y para qué sirve? (parte 1) | Magazcitum." <http://www.magazcitum.com.mx/?p=50>. Página vigente al 4 Ago. 2016.

"Información general de CMMI - MSDN - Microsoft." 2015. <https://msdn.microsoft.com/es-es/library/ee461556.aspx>. Página vigente al 7 Ago. 2016.

"Qué significa CMMI: Qué es CMMI." 2013. <http://asprotech.blogspot.com.ar/2013/10/que-es-cmmi.html>. Página vigente al 7 Ago. 2016.

"WordPress › Blog Tool, Publishing Platform, and CMS." 2004. <https://wordpress.org/>. Página vigente al 10 Nov. 2014.

"jQuery Release Notes · GitHub - Gist." 2014. <https://gist.github.com/teppeis/9264080>. Página vigente al 17 Ago. 2016.

"Releases · laravel/laravel · GitHub." 2013. <https://github.com/laravel/laravel/releases>. Página vigente al 18 Ago. 2016.

"Tags - GitHub." 2013. <https://github.com/twbs/bootstrap/tags>. Página vigente al 18 Ago. 2016.

"Open Hub, the open source network." 2014. <https://www.openhub.net/>. Página vigente al 11 Nov. 2014.

"What is Kanban? - Everyday Kanban." 2012. <http://www.everydaykanban.com/what-is-kanban/>. Página vigente al 19 Ago. 2015.

"What is Kanban? - LeanKit." 2016. <https://leankit.com/learn/kanban/what-is-kanban/>. Página vigente al 19 Ago. 2015.

"Metodología Kanban | Kanban Tool." 2014. <http://kanbantool.com/es/metodologia-kanban>. Página vigente al 19 Ago. 2015.

"A Brief Introduction to Kanban | The Agile Coach - Atlassian." 2014. <https://es.atlassian.com/agile/kanban>. Página vigente al 19 Ago. 2015.

"What Is Kanban? An Introduction to Kanban Methodology - VersionOne." 2015. <https://www.versionone.com/what-is-kanban/>. Página vigente al 19 Ago. 2015.

"Metodologías ágiles. ¿Cuál elegir? - Project ManagementProject ..." 2014. <http://www.obs-edu.com/blog-project-management/temas-actuales-de-project-management/metodologias-agiles-cual-elegir/>. Página vigente al 19 Ago. 2015.

"Asana is the easiest way for teams to track their work · Asana." 2011. <https://asana.com/>. Página vigente al 19 Ago. 2015.

"Kasban: The instant kanban board for Asana projects." 2016. <http://kasban.io/>. Página vigente al 20 Ago. 2015.

"David Allen's Getting Things Done® Methodology." 2003. <http://gettingthingsdone.com/>. Página vigente al 20 Ago. 2016.

"A successful Git branching model » nvie.com." 2010. <http://nvie.com/posts/a-successful-git-branching-model/>. Página vigente al 20 Feb. 2016.

"Bitbucket — The Git solution for professional teams." 2008. <https://bitbucket.org/>. Página vigente al 20 Feb. 2016.

"foundations of software testing - School of Computing." 2015. <http://www.computing.dcu.ie/~ray/teaching/CA358/dorothy_graham.pdf>. Página vigente al 18 Sep. 2016.

"GitHub - nvie/gitflow: Git extensions to provide high-level repository ..." 2010. <https://github.com/nvie/gitflow>. Página vigente al 21 Ago. 2016.

KANBAN

La aplicación de la metodología Kanban en la empresa es fundamental para progresar hacia un esquema de desarrollo más profesional.

Kanban es una metodología que surgió como tal en la empresa japonesa Toyota, que la utilizó para mejorar el proceso de producción de automóviles, dividiendo el mismo en fases. Este mismo concepto de división en fases es lo que permite trasladarlo al desarrollo de software donde también se presentan fases claramente definidas (selección, análisis, desarrollo, prueba, entrega).

A continuación se exponen los principios más básicos de esta metodología, el porqué de su elección y como se aplica en la empresa.

Principios básicos[12] [13] [14]

Visualizar el flujo de trabajo: la idea general es que es más fácil y rápido procesar información visual que texto y listas de texto. Por lo tanto el método recomienda usar un tablero real con divisiones y notas pegadas en ese tablero para visualizar de forma sencilla cuáles tareas se están haciendo y quien las está haciendo, más otra información que se puede agregar a criterio de cada equipo.

Limitar WIP: WIP se traduce como “trabajo en curso”. Esta es una regla fundamental de Kanban y una de las pocas restricciones que el método propone para lograr el éxito y evitar cuellos de botella. Funciona definiendo un límite para cada una de las divisiones del tablero y respetando ese límite. Este límite puede modificarse a medida que se comprende mejor cual es aquella medida en la que el equipo funciona sin estrés y motivado a la vez que es productivo en las tareas que realiza.

Gestión del flujo: Visualizar y analizar en todo momento que exista un movimiento continuo y fluido de las tareas es muy importante en Kanban pues la metodología incentiva los cambios para mejorar el proceso mismo cuando se detectan problemas o cuellos de botella.

Establecer políticas claras: Para sacarle provecho a algo hay que entenderlo. No solo debe entenderlo uno mismo sino que todo el equipo debe entenderlo, para avanzar siempre en dirección positiva. Así, por ejemplo, para que una tarea avance de un estado a otro esta deberá cumplir requisitos claros. Esto evita confusiones, discusiones emocionales y subjetivas.

---

[12] "What is Kanban? - Everyday Kanban." 2012. <http://www.everydaykanban.com/what-is-kanban/>. Página vigente al 19 Ago. 2015.

[13] "What is Kanban? - LeanKit." 2016. <https://leankit.com/learn/kanban/what-is-kanban/>. Página vigente al 19 Ago. 2015.

[14] "Metodología Kanban | Kanban Tool." 2014. <http://kanbantool.com/es/metodologia-kanban>. Página vigente al 19 Ago. 2015.

Mejora continua: Kaban incentiva a los equipos a desarrollar una cultura de mejora continua, realizando aportes pequeños e incrementales en grupo. Esto se logra mediante experimentos y análisis pero sobre todo mediante la comprensión en equipo de los problemas.

Justificación de la elección

El hecho de que hacía falta utilizar una metodología para mejorar la gestión de las tareas de desarrollo en la empresa está claro.

A partir de allí se decidió que era necesaria una metodología ágil puesto que es imperativo tener este tipo de metodología por la naturaleza de la empresa, el equipo de trabajo y los trabajos realizados:

- Cambios permanentes en las tareas a realizar.
- Equipo pequeño.
- Gran cantidad de tareas de distintas dificultad para distintos clientes.

Y entre las metodologías ágiles se decidió por Kanban porque [15] [16] [17]:

Es sencillo de aprender y poner en marcha. Esto es muy importante para un equipo conformado muchas veces por trabajadores temporales.

Es altamente flexible en sus características. Es ideal como metodología inicial para cualquier proyecto puesto que se puede ir ajustando con el tiempo.

No requiere roles predefinidos. A diferencia de SCRUM que requiere roles como *Product Owner* o *Scrum Master*.

No existen sprints. A diferencia de SCRUM donde existen *sprints* (batch de tareas) para realizar en un periodo determinado de tiempo (2 semanas generalmente) en Kanban las tareas se manejan individualmente. En la empresa en cuestión esto es lo ideal puesto que el tiempo de cada trabajo solicitado nunca puede enmarcarse en un periodo fijo, algunos duran un par de días, otros varios meses.

Flexibilidad en la aceptación de nuevas tareas. Siempre que se cumpla con el límite de WIP se puede aceptar tareas en cualquier momento. Situación que en la empresa sucede casi cada día puesto que cada dia se solicitan nuevas modificaciones a trabajos previos o se detectan bugs que necesitan resolución de la manera más rápida posible.

[15] "A Brief Introduction to Kanban | The Agile Coach - Atlassian." 2014. <https://es.atlassian.com/agile/kanban>. Página vigente al 19 Ago. 2015.

[16] "What Is Kanban? An Introduction to Kanban Methodology - VersionOne." 2015. <https://www.versionone.com/what-is-kanban/>. Página vigente al 19 Ago. 2015.

[17] "Metodologías ágiles. ¿Cuál elegir? - Project ManagementProject ..." 2014. <http://www.obs-edu.com/blog-project-management/temas-actuales-de-project-management/metodologias-agiles-cual-elegir/>. Página vigente al 19 Ago. 2015.

Aplicación de Kanban en la empresa

Para aplicar esta metodología en la empresa se decidió usar la herramienta Asana[18] puesto que es altamente flexible y puede adaptarse a casi cualquier uso, además de eso es gratuita y los miembros de la empresa la usan desde hace años por lo tanto la conocen bien, aunque no la han usado con ninguna metodología sino más bien como una simple lista de tareas.

Se decidió acompañar esta herramienta con Kasban[19], otra herramienta pensada específicamente para integrar la metodología Kanban usando Asana. Esta agrega tableros visuales para acercarse todavía más a la recomendación de usar indicadores visuales para conocer el estado y flujo de las tareas.

Los estados propuestos para una tarea son:

- ***Uncategorized* o Sin categorizar o Bandeja de entrada.** Que haría las veces de bandeja de entrada para las tareas. Aquí habrá que anotar las tareas sea cual sea el servicio o medio usado para enviarla a la empresa.
- ***Accepted* o Próximas.** Donde solo estarán presentes aquellas tareas que deberán realizarse próximamente. Estas normalmente serán sacadas de *Uncategorized* o puestas aquí por casos de urgencia.
- ***Doing*.** Son aquellas tareas en las que efectivamente se está trabajando en este momento.
- ***Completed*.** Las tareas finalizadas.

[18]"Asana is the easiest way for teams to track their work · Asana." 2011. <https://asana.com/>. Página vigente al 19 Ago. 2015.

[19]"Kasban: The instant kanban board for Asana projects." 2016. <http://kasban.io/>. Página vigente al 20 Ago. 2015.

Ejemplo de tablero:

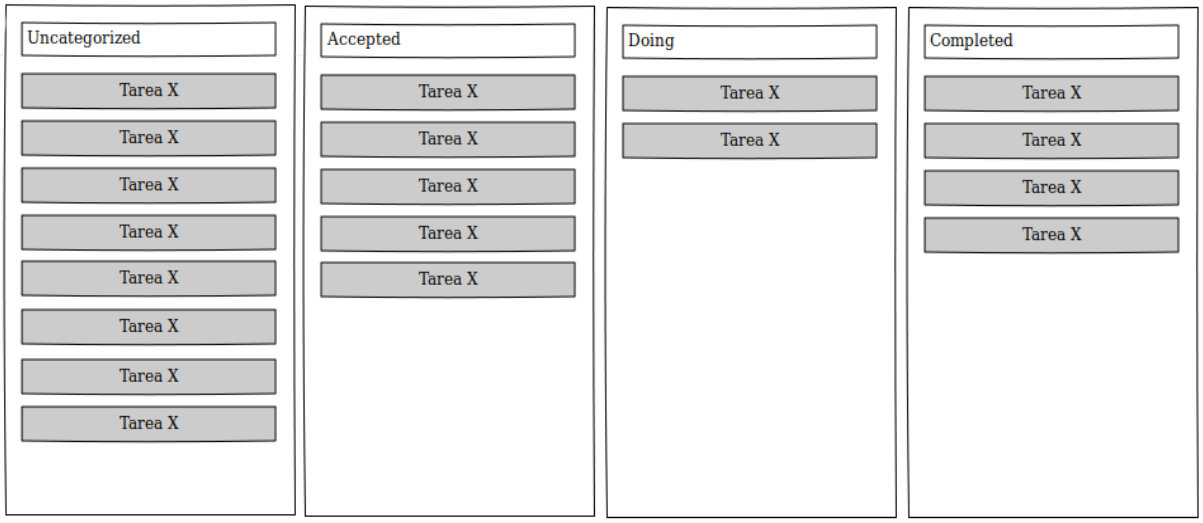

Figura 7: Ejemplo de tablero Kanban. Fuente: Elaboración propia.

Se recomienda realizar revisiones al principio de cada semana sobre el orden de las tareas, para mantener el flujo sin obstrucciones y darle la importancia que se merece cada tarea. Así por ejemplo, cada semana, se elegirán nuevas tareas de la sección *Uncategorized* o Bandeja de entrada para pasar a un estado de *Accepted* o Próximo. Y también pueden sacarse tareas de la sección de *Accepted* que ya no sean necesarias hacer por el motivo que fuere.

Los conceptos “Bandeja de entrada”, “Próximo” y “Revisión semanal” se integran a esta metodología extraídos desde el método GTD (*Getting Things Done*)[20], un método que pretende librar a la mente de tareas improductivas como recordar que es lo que se debe hacer, utilizando para ello conceptos como los presentados, logrando con esto un sistema eficiente y ordenado para la gestión de tareas. Se han elegido solo estos 3 conceptos para integrarlos en la metodología Kanban porque ayudan a crear una cultura organizacional en la que se confía en las herramientas utilizadas a la vez que mantiene el flujo de tareas en funcionamiento gracias a la revisión semanal.

[20]"David Allen's Getting Things Done® Methodology." 2003. <http://gettingthingsdone.com/>. Página vigente al 20 Ago. 2016.

TABLA DE TIEMPOS DE DESARROLLO

Aprovechando que la metodología Kanban incentiva la mejora continua mediante la revisión de estadísticas y la experimentación, se propone incluir en esas estadísticas una tabla que permita poco a poco analizar y entender cuales son los tiempos manejados por la empresa para la entrega de un trabajo.

En base al conocimiento sobre el funcionamiento del nuevo Software de Desarrollo y la experiencia obtenida en la misma empresa se presentan los siguientes campos para formar la tabla:

- Diseño (en horas).
- Maquetación (en horas).
- Nro. de módulos totales del panel (en números enteros).
- Nro. de módulos a personalizar en el panel (en números enteros).
- Desarrollo del panel (en horas).
- Integración con la maqueta (en horas).
- Pruebas (en horas).
- Implementación (en horas).

Completar estos campos con la información de cada proyecto nos permitirá obtener un aproximado de cuanto tiempo llevara el desarrollo de un sitio web completo.

Y teniendo en cuenta que, si se realiza un mantenimiento adecuado al panel, cada vez habrá menos módulos a personalizar (porque se irán agregando al Software de Desarrollo) entonces el tiempo total irá disminuyendo con el tiempo.

Por supuesto todo esto solo es posible llevando registros detallados y ordenados con herramientas como Excel u Hojas de cálculo de Google Drive.

Y aunque se acepta que el método propuesto no es el más robusto y preciso, se lo propone en base a la facilidad del mismo y en consideración de que las solicitudes que recibe la empresa suelen ser por lo general de trabajos de la misma naturaleza.

VERSIONADO DE SOFTWARE

El usar un sistema de versionado de software es una necesidad en la actualidad. Por ello resulta sorprendente que en la empresa no se utilizara o que se utilizara con objetivos distintos al de mantener un registro y control de cambios en el software, como ser: *backup* de proyectos.

La propuesta es avanzar hacia la correcta utilización de un sistema de versionado de software usando un modelo ampliamente conocido y aceptado como es el modelo de nvie[21].

Este modelo utiliza GIT como tecnología de versionado de código. Tecnología conocida por todos los miembros de la empresa, siendo esto la principal razón de su elección pero además de ser aquella tecnología sobre la que es más fácil encontrar documentación o solución a situaciones específicas.

Acompaña a este sistema de versionado de código el servicio bitbucket[22]. Servicio que permite repositorios privados de forma gratuita y con el cual la empresa lleva trabajando hace tiempo.

El modelo de nvie puede ser adaptado o usado en distintas situaciones. Luego de analizarlo se propone usarlo en 2 variantes dependendiendo del tamaño del proyecto.

Para proyectos pequeños

Un proyecto pequeño en la empresa será generalmente lo que se conoce como un sitio web empresarial, donde trabaja, hasta la primer entrega, un solo desarrollador. Luego, a la hora de realizar mantenimiento, puede o no intervenir otro desarrollador, aunque de preferencia siempre se busca que el mantenimiento siempre lo realice el mismo que desarrolló el sitio web la primera vez.

En esta situación, lo recomendable es usar solo 2 ramas, una para producción y otra para desarrollo. La rama de producción siempre representa aquello que se encuentre en producción (o sea en línea). Y la rama de desarrollo será la utilizada tanto por el desarrollador como para realizar montajes de prueba del sitio cuando se necesite.

[21]"A successful Git branching model » nvie.com." 2010. <http://nvie.com/posts/a-successful-git-branching-model/>. Página vigente al 20 Feb. 2016 .

[22]"Bitbucket — The Git solution for professional teams." 2008. <https://bitbucket.org/>. Página vigente al 20 Feb. 2016.

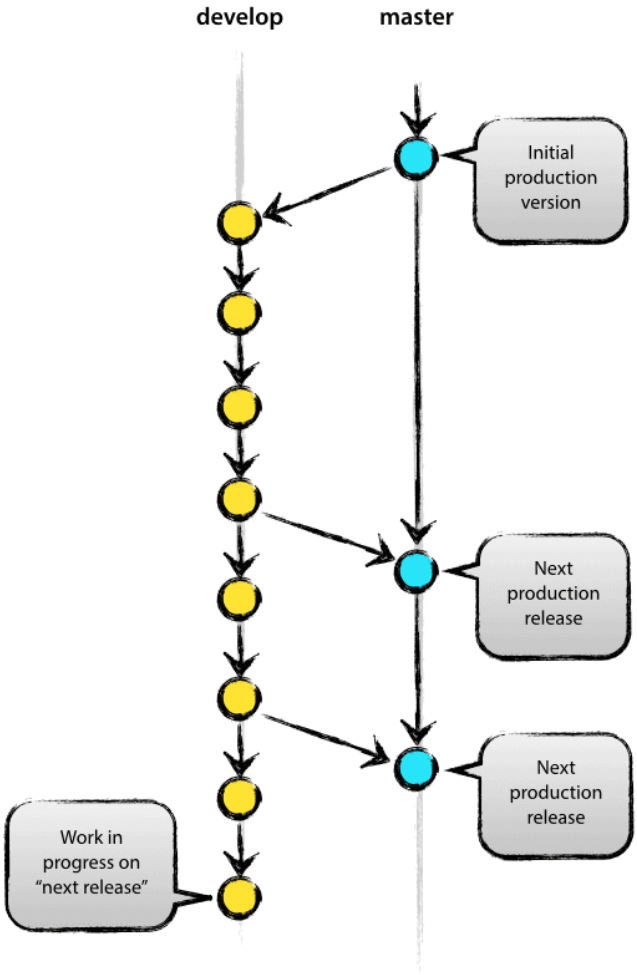

Figura 8: Modelo nvie 1. Fuente: http://nvie.com/posts/a-successful-git-branching-model/

Para proyectos grandes

Los proyectos grandes normalmente requieren pruebas más exhaustivas y más desarrolladores. En este caso lo ideal es trabajar las mismas 2 ramas que el caso anterior pero sumando las ramas de *'feature'*. Así, cada uno puede trabajar en funcionalidades distintas al mismo tiempo sin ser ningún impedimento para otros desarrolladores trabajando sobre el código de un software.

Cada rama *feature* siempre surge desde la rama *develop*.

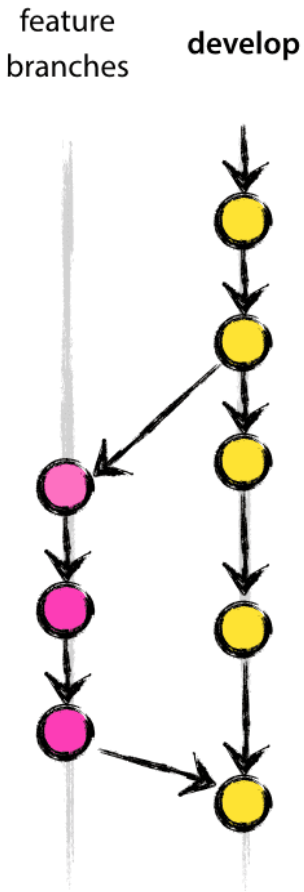

Figura 9: Modelo nvie 2. Fuente: http://nvie.com/posts/a-successful-git-branching-model/

BACKUPS DE PROYECTOS

Para los *backups* de proyectos se han integrado los paquetes necesarios para realizar dicha acción en el Software de Desarrollo.

Un *backup* estará conformado por 3 partes:

- El software.
- La base de datos.
- Los archivos subidos por el cliente.

Esto se ha pensado de esta manera pues se considera la mejor estrategia para recuperar rápidamente una, algunas o todas las partes que hayan sido afectadas por cualquier inconveniente.

Sobre el backup del software: Al empezar a usar un sistema de versionado de software y tener la última versión del mismo siempre disponible en la rama *master* de su repositorio, recuperar el mismo es tan sencillo como clonar el repositorio nuevamente en el *hosting*.

Sobre el backup de la base de datos: Teniendo en cuenta que laravel usa migraciones para crear y administrar la estructura de la base de datos, solo es necesario hacer *backup* de los datos. Estos estarán guardados en Dropbox, se realizarán de manera diaria de manera automática gracias a un paquete de laravel, y recuperarlos es tan sencillo como bajar un archivo e importarlo desde PhpMyAdmin o cualquiera herramienta para administrar MySQL.

Sobre el backup de archivos subidos por el cliente: De forma predeterminada y automática se hará un *backup* diario, en Dropbox, de la carpeta "*public*" del proyecto donde estarán los archivos subidos por el cliente. Recuperarlos será otra vez tan sencillo como bajarlos y subirlos mediante FTP o similar. Esto también es posible gracias a la cuidadosa elección de un paquete para laravel que realiza esta función.

FLUJO DE DEPLOY

La actual forma de subir proyectos a producción es usando FTP para subir el proyecto completo y exportando/importando la BD. Esto por supuesto trae muchos inconvenientes a la implementación e incluso muchos más cuando hay que realizar mantenimiento a algún proyecto.

Lo que se propone es un flujo de *deploy* muy sencillo pero que permite mantener un orden adecuado a la hora de la primera implementación y los futuros cambios por mantenimiento.

Durante el desarrollo

Usar siempre la rama *'develop'* tal como propone el modelo de nvie (Ver sección de Versionado de software). Usar la rama *'master'* solo a la hora de hacer un *push* hacia el repositorio del proyecto del código que está listo para usarse en producción.

En ningún caso se usará FTP o cualquier medio de conexión para subir archivos directamente al *hosting*.

En ningún caso se harán cambios a la estructura de la base de datos que no sean mediante el sistema de migraciones y *seeders* de Laravel.

Durante la implementación

La implementación se hará clonando el repositorio del proyecto en Bitbucket directamente desde el servidor. A continuación se correrán los comandos por todos conocidos para instalar Laravel, sus paquetes y correr las migraciones.

Durante el mantenimiento

Al usar el modelo propuesto se vuelve imposible realizar cambios sin tener el repositorio local actualizado por medio del repositorio remoto en Bitbucket. Esto evita todos los problemas actuales de modificación de archivos de manera accidental, situación que se presenta especialmente cuando en un mismo proyecto trabaja más de una persona.

PRUEBAS

Pruebas unitarias

Teniendo en cuenta que en la empresa no se han realizado pruebas nunca sobre los proyectos que se entregan más allá de unas esporádicas pruebas manuales, y de que existe cierta resistencia al cambio se ha optado por la adopción de pruebas unitarias sobre los trabajos entregados.

Se han elegido este tipo de pruebas principalmente porque:

- Son el tipo de prueba más conocido entre los miembros del equipo.
- Son relativamente sencillas de escribir y llevan poco tiempo.
- Se pueden ejecutar de manera automática gracias a software como PHPUnit que viene integrado en el *framework* Laravel.
- Son una guía de desarrollo para cualquier desarrollador que se involucre en el proyecto luego de iniciado el mismo.
- Son repetibles.

Los beneficios que se esperan obtener son:

- Reducción de costos de mantenimiento.
- Reducción de tiempo necesario en realizar pruebas repetidas al realizar cambios en el software.

Implementación de pruebas

Para adoptar la creacion y ejecucion de pruebas en la empresa se deberán seguir algunos de sus principios básicos[23].

- Como **es imposible probar todo**, se recomienda probar aquellos módulos que se consideren más relevantes para el cliente.
- **Las pruebas se deben planificar**. Se recomienda que en el mismo momento en el que se definen las especificaciones para el proyecto se creen casos de prueba. Para esto también se usará la herramienta Asana y se integrarán estas tareas en la Bandeja de entrada del tablero de Kanban.
- **Las pruebas se deben realizar lo antes posible**. Esto quiere decir que se deberá escribir una prueba cuando el módulo al que hace referencia su definición se ha completado.
- Cuando se realizan modificaciones en el software durante el mantenimiento, estas modificaciones deben ir acompañadas de una **revision y modificacion de los casos de prueba** involucrados o de la creación de nuevos casos de prueba según sean necesarios.

[23]"foundations of software testing - School of Computing." 2015. <http://www.computing.dcu.ie/~ray/teaching/CA358/dorothy_graham.pdf>. Página vigente al 18 Sep. 2016.

MANUAL DE USO DEL SOFTWARE DE DESARROLLO

SCREENSHOTS

Se adjuntan algunas capturas para tener un ejemplo de como deberia quedar el panel luego de su instalación y configuración.

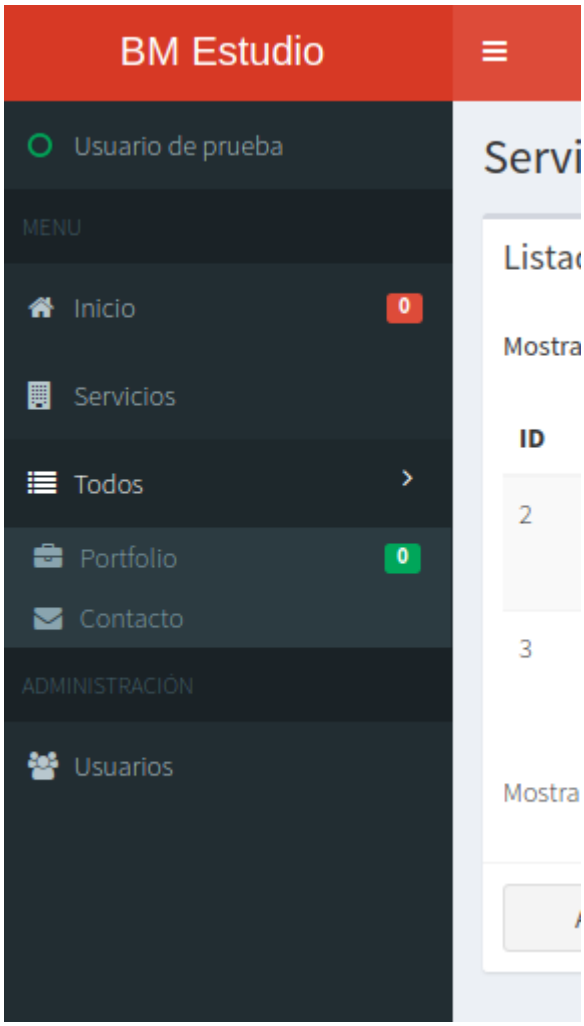

Figura 10: Captura de pantalla del menú. Fuente: Elaboración propia.

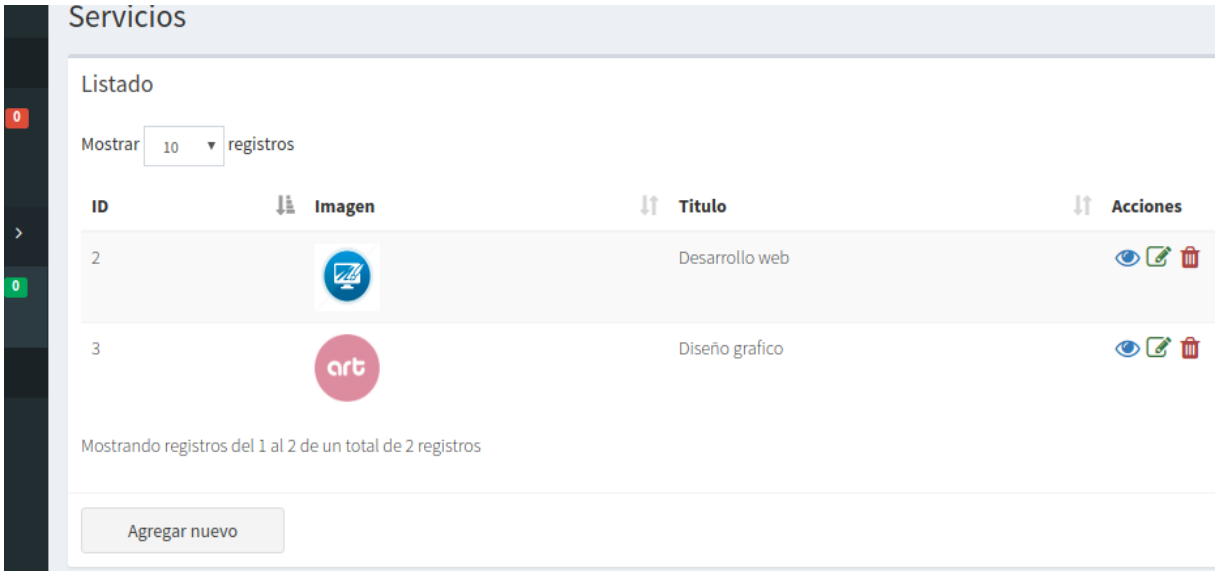

Figura 11: Captura de un listado del panel. Fuente: Elaboración propia.

REQUISITOS

PHP, Apache, Git, Git Flow [24], Gulp, NPM, Composer, editor de texto como Sublime Text o Atom, navegador moderno como Google Chrome o Mozilla Firefox.

INSTALACIÓN

La instalación del software seguirá los siguientes pasos:

Paso 1 - Clonar el repositorio desde github en una carpeta con permisos de ejecución para php (generalmente /srv/http si usas Linux) con el siguiente comando:

```
git clone https://github.com/sefsinalas/DogAdmin.git nuevo_proyecto
```

Paso 2 - Ingresar mediante consola a la carpeta del proyecto y ejecutar los siguientes comandos:

```
npm install
```

[24] "GitHub - nvie/gitflow: Git extensions to provide high-level repository ..." 2010. <https://github.com/nvie/gitflow>. Página vigente al 21 Ago. 2016.

```
composer install
gulp
cp .env.example .env
php artisan key:generate
sudo chmod -R 777 storage
composer dump-autoload
```

CONFIGURACIÓN

La configuración del software de desarrollo seguirá los siguientes pasos:

Paso 1 - Abrir una consola de sistema en la carpeta del proyecto e iniciar git flow con el siguiente comando:

```
git flow init
```

Aceptar las opciones predeterminadas de git flow.

Paso 2 - Cambiar el repositorio remoto por el repositorio del proyecto que se busca realizar con el siguiente comando

```
git remote add origin https://github.com/nombre_usuario/nombre_proyecto.git
```

Paso 3 - Modificar el archivo config.json que servirá para crear los módulos del panel.

Este archivo estará divido en 3 partes: ***General, Modules* y *Menus*.**

General serán propiedades para configurar distintos aspectos del software en sí, como ser: Nombre del proyecto, datos para conexión a la base de datos, color primario del panel, *layout* del *template*, etc.

Modules será un array de los módulos, donde cada módulo tendrá a su vez 2 propiedades: ***General* y *Fields***.

La propiedad ***General*** de cada *Module* estará conformada por propiedades específicas para ese módulo, como ser: tipo de módulo, estado, tabla principal, etc.

Fields será un *array* de los campos que tendrá cada módulo, y cada elemento tendrá sus propiedades como: nombre en la base de datos, etiqueta, tipo, etc.

Menus será un *array* los accesos directos que formarán el menú para acceder a cada módulo.

Toda la documentación sobre las propiedades de este archivo de configuracion estará disponible y actualizada en el repositorio github del proyecto (https://github.com/sefsinalas/DogAdmin) y en el archivo *README*.md del software, pudiendo visualizarse perfectamente formateadas con cualquier visor de archivos markdown.

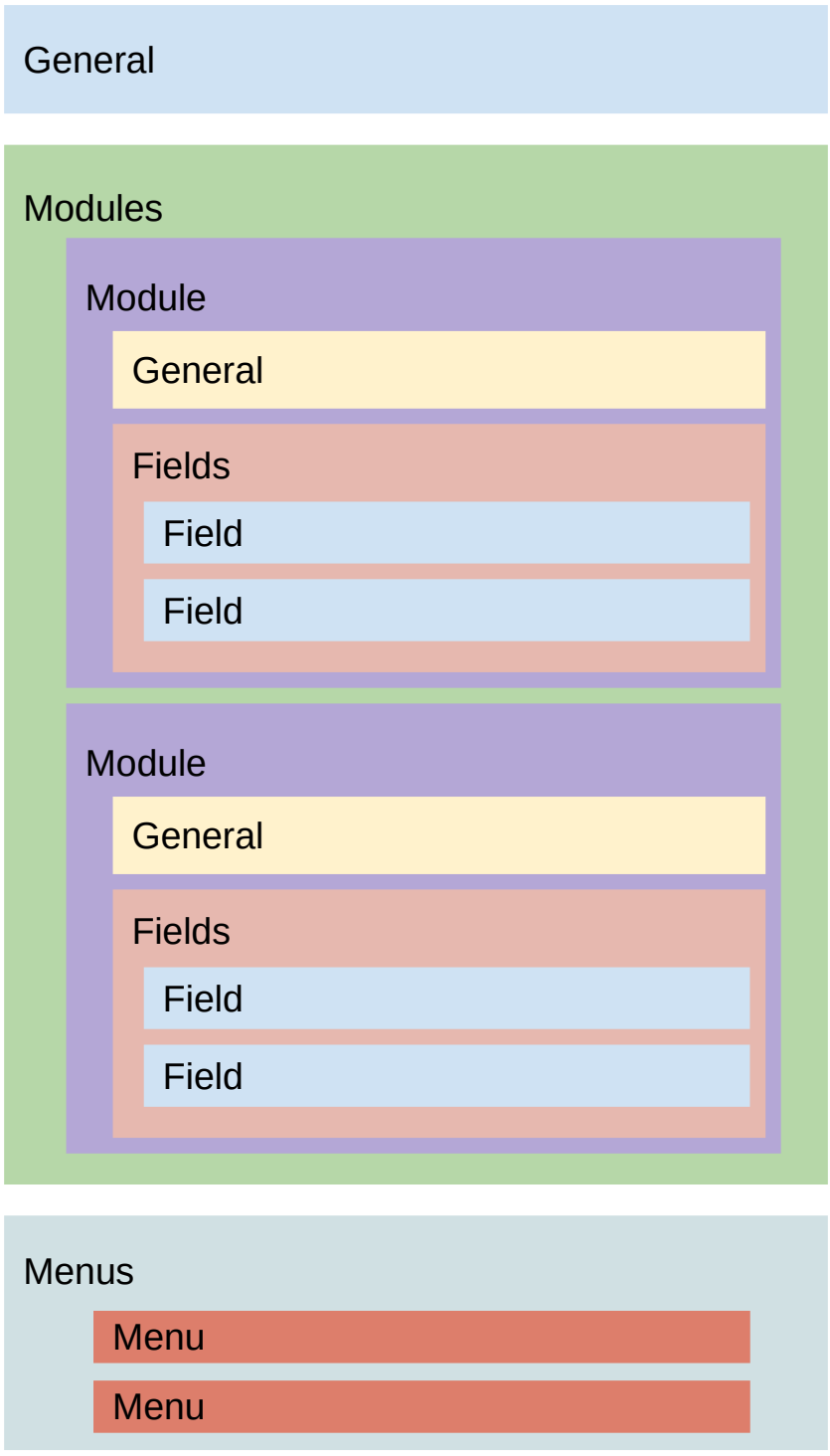

Figura 12: Diagrama de configuración. Fuente: Elaboración propia.

DESARROLLO

Cuando se ha terminado de configurar todo, crear un panel será ser tan sencillo como correr el comando:

```
php artisan dogadmin:install
```

Este comando ejecutará varias tareas en secuencia:

- Copiar configuraciones globales en el archivo .env
- Crear y correr las migraciones
- Crear un usuario de prueba
- Crear los modelos para los datos
- Crear los controladores para los módulos
- Crear las vistas para interactuar con los módulos
- Crear las rutas para acceder a los módulos
- Crear el menú del panel
- Crear los seeders para cargar la BD con datos de prueba

A continuación basta con entrar a la url del proyecto agregando al final /*login* para tener acceso al panel. Las credenciales para el *login* serán proporcionadas en la misma consola luego de ejecutar el comando anterior. Estos datos de prueba y exclusivamente para usarse durante el desarrollo son:

- **Email**: demo@demo.com
- **Password**: demo123

Luego de ejecutado el comando install se puede usar el comando:

```
php artisan db:seed
```

El anterior comando usarlo únicamente si se han definido datos falsos (*fake*) en el archivo config.json.

Se proporciona además el comando:

```
php artisan dogadmin:reset
```

Este comando volverá atrás todos los cambios, es decir, eliminará modelos, controladores, rutas, vistas y demás. Será usado exclusivamente durante el desarrollo para

agilizarlo y volver rápidamente atrás en caso de haber cometido un error en el archivo config.json

NUEVOS CAMPOS

Lo siguiente serán los pasos necesarios para agregar nuevos tipos de campos.

Paso 1- Crear una nueva carpeta cuyo nombre sea el tipo (*type*) de campo que se desea crear. Colocar esta carpeta en resources/stubs/fields.

Paso 2- Dentro de la carpeta colocar los *templates* necesarios. Estos serán:

- **form.stub:** Este define el *template* que se visualizará para ese campo en el formulario para agregar o editar un nuevo item de módulo.
- **listing.stub:** Este define el *template* que se visualizará para ese campo en el listado.
- **show.stub:** Este define el *template* que se visualizará para ese campo en la vista de detalle del item.

Paso 3- Crear una nueva carpeta cuyo nombre sea el tipo (*type*) de campo que se desea crear. Colocar esta carpeta en app/Includes.

Paso 4- Dentro de esta carpeta colocar archivos que definirán o extenderán la funcionalidad para cada campo.

- **data.php**: Este define tipos de campo que se usarán en las migraciones, validación en los modelos, etc.
- **properties.php**: Este servirá para agregar o modificar las propiedades que se usarán en los *templates* del paso 2. Y en el resto de los archivos de la carpeta *Include*.
- **store.php (opcional)**: El contenido de este archivo se incluirá en la función de guardado (*store*) de cada item.

En todos los casos, cada *template* (.stub) tendrá a su disposición todas propiedades definidas para ese campo en el archivo config.json más aquellos aquellas propiedades que se definan en properties.php.

PRUEBAS

Para realizar las pruebas se deberá subir el sitio mediante git a un subdominio de pruebas del proyecto. Ej: test.miproyecto.com

Se desaconseja totalmente el uso de FTP para subir los archivos y se recomienda en su cuenta ingresar al hosting usando SSH y correr los mismos comandos para la instalación, desde la clonación del repositorio.

Esto permitirá evitar una gran cantidad de inconvenientes provocados por la interacción humana en el proceso de deploy en flujos basados en FTP.

Se recomienda a su vez usar la rama develop para hacer las pruebas.

IMPLEMENTACIÓN

Para implementar cada proyecto se deberán seguir los mismos pasos que para la realización de las pruebas pero usando la rama *master* del repositorio.

MANTENIMIENTO

A la hora de realizar mantenimiento de cualquier tipo en el proyecto es cuando realmente se notara una gran ventaja en el uso de este sistema. Los archivos generados en la instalacion y configuracion simplemente serán vistas, controladores y modelos básicos de laravel y no será necesario en ningún caso conocer para nada como fue desarrollado este software, por lo tanto cualquier persona con conocimientos en laravel puede realizar las modificaciones.

Sumado a esto tenemos el hecho de que si se sigue el modelo sugerido de versionado de código de nvie entonces se podrá conocer exactamente lo que realizaron otros desarrolladores en cada proyecto.

A la hora de hacer alguna clase de modificación en las vistas se recomienda usar paquetes para el formateado de código puesto que el software genera este código sin indentación de ningún tipo.

Paquetes recomendados para Sublime Text:

- https://packagecontrol.io/packages/HTMLBeautify
- https://github.com/akalongman/sublimetext-codeformatter