UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo de micrositios para una empresa constructora

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

José Manuel Novas Santamaría

ASESOR(A) DE INFORME

Ing. Carlos Alberto Román Zamitiz

Ciudad Universitaria, Cd. Mx., 2019

Agradecimientos

En primer lugar, quiero agradecer a mis padres, ellos me ayudaron desde el principio de mi etapa escolar hasta esta instancia, me han dado todas las herramientas para poder culminar mis metas. A mi hermano que me ha acompañado siempre y con quien aprendo cada día algo nuevo. A mi madrina que me ha enseñado que la dedicación es un aspecto crucial a la hora de alcanzar metas. A mi padrino que ha sido un gran ejemplo que seguir. A mi familia que ha sido un gran soporte en esta nueva etapa de mi vida.

También quiero agradecer a mis compañeros de carrera, esa familia que me tocó elegir y que superamos juntos cada etapa de nuestra carrera profesional. A cada uno de mis profesores que me han inculcado valores en cada una de sus clases y me han dado valiosas herramientas para mi desarrollo profesional. A mi asesor de informe el Ingeniero Carlos Alberto Román Zamitiz por haber invertido su valioso tiempo en ayudarme a revisar el presente trabajo.

Por otro lado, quiero agradecer al Instituto de Ingeniería por haberme introducido a un ambiente más lejos de lo académico. A la Doctora Lourdes Angélica Quiñones Juárez por haberme invitado a ser parte de esta importante institución que ha aportado numerosos proyectos para el desarrollo científico de nuestro país y de la humanidad.

Por último, pero no menos importante, quiero agradecer a mis compañeros de trabajo, a mis jefes y directivos por haberme apoyado tanto con el presente proyecto. Quiero agradecer el darme la oportunidad de trabajar con proyectos de esta magnitud y por ayudarme a crecer profesionalmente una vez que terminé la licenciatura.

Objetivo

Desarrollar una plantilla web que permita agilizar el desarrollo de un máximo de 10 micrositios para una empresa constructora.

Desarrollar un CMS (Sistema de Gestión de Contenidos por sus siglas en inglés) que permita administrar el contenido y estado de los espacios que se encuentran en venta/renta de los desarrollos de la constructora.

Índice desglosado

Marco teórico

En sus inicios, el desarrollo de software fue considerado una actividad artesanal. Esto quiere decir que no existía una serie de estándares o metodologías que buscaran regular la generación de código fuente. Debido a las exigencias actuales, es necesario aplicar técnicas que permitan construir software de calidad regido por diversos estándares y requerimientos. La ingeniería de software busca lograr dicha meta, entregar proyectos de software de calidad en tiempo y presupuesto.

Antecedentes del tema

No es una novedad que el software sea un producto de vital importancia para las personas y corporaciones de todo el mundo. Esto ha provocado que se encuentre presente en la mayoría de las actividades laborales y cotidianas del hombre. Por ende, las exigencias del software se han vuelto tan complejas que es necesario entregar software de calidad al usuario final.

La necesidad de tener un software desarrollado a la medida para determinada empresa se debe a su capacidad de procesar grandes cantidades de información, con el objetivo de que pueda estar disponible para los usuarios finales. Esto ha llevado a que la industria del software sea un factor dominante en la economía de un país.

El software se comprende como una serie de instrucciones que se ejecutan en una computadora para mostrar resultados. A diferencia de cualquier otro producto, el software no es un elemento tangible, más bien es un elemento lógico. Dicho elemento no sufrirá un desgaste con el paso del tiempo, sin embargo, se va a deteriorar debido al cambio. Por ende, no puede ser administrado como un producto de manufactura.

Existen siete clases de software en la industria: de sistemas, de aplicación, de ingeniería y ciencias, incrustado, de línea de productos, aplicaciones web y de inteligencia artificial. El software del proyecto que desarrollaré en el capítulo siguiente es de la clase de aplicaciones web. Esta clase de software está centrado en redes, la cual comunica una serie de archivos de hipertexto que presenta información de manera gráfica y está integrado por bases de datos corporativas y aplicaciones de negocios.

Si bien una aplicación web o webapp es una clase de software, esta debe ser tratada de manera diferente a las demás clases. Esto es debido a que, al estar alojada en la red, se encuentra al alcance de prácticamente todo el mundo. Por lo tanto, debe satisfacer las siguientes características: uso intensivo de redes, concurrencia, carga impredecible, rendimiento, disponibilidad, orientada a los datos, contenido sensible, evolución continua, inmediatez, seguridad y estética.

Así como existe software por desarrollar, también existe software heredado, el cual comprende aquellos programas que se han estado desarrollando desde décadas y que en su momento cumplieron con los requerimientos de los usuarios. Sin embargo, los cambios del hardware y requerimientos de negocio hacen que el software necesite mantenimiento para que pueda evolucionar.

Ya teniendo claro el concepto de software y sus diversas clases, es preciso pasar a entender a la Ingeniería de Software, la cual es una disciplina tecnológica compuesta de varias capas: compromiso con la calidad, proceso, método y herramientas. La ingeniería de software se basa en los siguientes principios:

1. Se debe entender el problema antes de desarrollar una aplicación.
2. El diseño debe ser una actividad crucial.
3. El software debe tener una alta calidad.
4. El software debe ser fácil de mantener.
5. Debe ser aplicable en todas las formas y dominios del software.

La capa de proceso define un enfoque estandarizado que permite al equipo de software determinar las acciones y tareas para llevar a cabo el trabajo. La estructura del proceso de software consiste en las siguientes actividades. Estas se desenvuelven de manera iterativa o de manera lineal dependiendo de la metodología seleccionada:

1. Comunicación: entender los objetivos principales del proyecto.
2. Planeación: establecer las tareas, identificar riesgos, asignar recursos, establecer resultados y programar actividades.
3. Modelado: entender mejor los requerimientos y establecer un diseño para satisfacerlos.
4. Construcción: desarrollo de código y pruebas.
5. Despliegue: entrega y retroalimentación hacia el consumidor.

Por otro lado, se llevan a cabo actividades sombrillas, las cuales se encargan de administrar y controlar los avances en cualquier proyecto de software y son complementarias a la estructura del proceso vista anteriormente:

1. Seguimiento y control del proyecto de software: evaluación del progreso contra el plan del proyecto.
2. Administración del riesgo: todo aquello que pueda afectar el resultado y calidad del proyecto.
3. Aseguramiento de la calidad del software: actividades requeridas para garantizar la calidad del software.
4. Revisiones técnicas: evaluar los productos para identificar y eliminar errores antes de iniciar la siguiente actividad.
5. Medición: define y reúne mediciones en el proceso, proyecto y producto.
6. Administración de la configuración del software: identifica los efectos de un cambio en el proceso de software.
7. Administración de la reutilización: define los componentes que se pueden usar en un proyecto futuro y los que pueden ser utilizados en el proyecto actual.
8. Preparación y producción del producto de trabajo: se encarga de preparar el producto, así como la documentación del proyecto.

Ya teniendo definidas las actividades estructurales y las actividades sombrilla en el proceso de software, es preciso conocer cómo se van a aplicar en la práctica. Para ello se deben aplicar las siguientes actividades:

1. Entender el problema: se aplica en las etapas de comunicación y análisis. No siempre es fácil entender un problema y para ello se debe tener en claro quiénes son los participantes, identificar las incógnitas, saber si se puede fraccionar el problema y si es posible representar de manera gráfica el problema.
2. Planear la solución: se aplica en el modelado y en el diseño del software. Antes de codificar la solución del problema, es necesario analizar si existe algún software que cumpla con las características del desarrollo, identificar si es reutilizable lo que ya está hecho, conocer si puede definirse en problemas más pequeños y saber si es posible hacer un modelo de diseño.

3. Ejecutar el plan: se aplica en la generación del código. Ya teniendo el diseño de la solución, es preciso conocer si dicho diseño es aplicable a la solución del problema y saber si cada componente de la solución es aplicable.
4. Examinar la exactitud del resultado: se aplica en el proceso de pruebas y aseguramiento de la calidad. Si bien es cierto que la mayoría de las soluciones de software no son perfectas, se puede aplicar el número de pruebas necesarias para poder asegurar la calidad de un proyecto. Para ello se debe evaluar la solución final contra los requerimientos iniciales.

Para poder desarrollar software de calidad, David Hooker estableció siete principios, los cuales se pueden aplicar en la práctica de la Ingeniería de Software dentro de las actividades estructurales y las tareas técnicas. Estos principios buscan ayudar a crear herramientas mentales que ayuden a mejorar la práctica de la Ingeniería de Software.

1. La razón de que exista todo: buscar incluir en el desarrollo de software sólo valores agregados.
2. MSE: establecer un diseño simple del sistema hace que el mismo sea más fácil de comprender y, por lo tanto, sea susceptible de mantener.
3. Mantener la visión: si un diseño de software no tiene visión, por más de que este sea perfecto, en algún momento va a colapsar debido a la incompatibilidad de componentes o casos de uso no previstos.
4. Otros consumirán lo que usted produce: este principio va enfocado en las distintas etapas del desarrollo. Nunca se debe olvidar que un software será utilizado por personas, tanto en la ejecución como en el mantenimiento de éste.
5. Ábrase al futuro: en este momento ya no es factible diseñar software enfocado en el plan inicial. La evolución de hardware y software se mide en meses, por ende, nuestro desarrollo debe ser adaptable al cambio en periodos relativamente cortos.
6. Planee por anticipado la reutilización: el paradigma orientado a objetos promueve la reutilización de código a través de clases. Si un diseño tiene la capacidad de ser usado en otro proyecto, los costos de desarrollo disminuyen significativamente.
7. Piense: tal vez parezca obvio, sin embargo, es el principio que se pasa por alto la mayoría del tiempo. Ejecutar el pensamiento antes de pasar a la práctica evita errores en el desarrollo y logra mejores resultados.

Un obstáculo que impide el desarrollo óptimo de la práctica de la Ingeniería de Software son los mitos del software. La mayoría de los mitos provienen de la etapa inicial de la computación. Sin embargo, algunos pueden llegar a parecer ciertos incluso en nuestros días. Debemos tomar en cuenta que estos mitos no dejan de ser consideraciones erróneas por más verídicas que parezcan. Existen tres tipos de mitos: del administrador, del cliente y del profesional.

Por lo general los administradores, independiente de si el proyecto es de software o de otro tipo, se sienten bajo presión debido a que se debe cumplir en tiempo y presupuesto un determinado proyecto. Es bastante común que se sostenga la creencia de algunos de los siguientes mitos:

1. *Tenemos un libro de estándares y procedimientos para elaborar software. ¿No le dará a mi personal todo lo que necesita saber?*: puede que exista y que sea un libro muy completo. Sin embargo, los requerimientos del software cambian en cuestión de meses. Por ende, los estándares no siempre son adaptables.
2. *Si nos atrasamos, podemos agregar más programadores y ponerlos al corriente:* A diferencia de la manufactura, la producción del software es un proceso intelectual. Por lo tanto, si se agregan más programadores a un proyecto atrasado, los programadores actuales deben dedicar tiempo en instruir al nuevo personal.
3. *Si decido subcontratar el proyecto de software a un tercero, puedo descansar y dejar que esa compañía lo elabore:* la empresa que requiere el software debe conocer sus procesos internos a la perfección para transmitir a la desarrolladora sus requerimientos de software.

Debemos tener en cuenta que el cliente puede ser cualquier persona. Esto quiere decir que el cliente puede tener vastos conocimientos en el campo de la computación, como puede tener pocos conocimientos computacionales. Esto podría generar falsas expectativas, llegando a un desacuerdo entre desarrollador y cliente.

1. *Para poder comenzar a escribir programas, es suficiente el enunciado general de los objetivos* – podremos entrar en detalles más adelante: Un requerimiento ambiguo seguramente producirá un error en el sistema. Para evitar eso, la especificación de requerimientos debe quedar lo más detallada y clara posible.
2. *Los requerimientos del software cambian continuamente, pero el cambio se asimila con facilidad debido a que el software es flexible:* Para que el cambio de requerimiento sea fácil de manejar, este se debe presentar en la etapa de comunicación. Si este cambio se presenta en una etapa posterior, el proyecto pierde su flexibilidad y cualquier cambio se convierte en un riesgo.

En cuanto a los mitos del profesional, estos se han estado alimentando desde el inicio de la computación. Esto se debe a que la programación inició como un proceso artesanal y aún se ve como tal. A continuación, se presentan algunos mitos:

1. *Una vez que escribamos el programa y hacemos que funcione, nuestro trabajo ha terminado:* Entre un 60% y 80% de las fallas del software se generan después de la primera entrega con el cliente.
2. *Hasta que no se haga "correr" el programa, no hay manera de evaluar su calidad:* Se pueden aplicar filtros entre cada etapa del software a través de revisiones técnicas.
3. *El único producto de trabajo que se entrega en un proyecto exitoso es el programa que funciona:* No sólo se debe entregar un programa funcional, la ingeniería de software establece entregar modelos, documentos y planes del propio desarrollo junto con el software.
4. *La ingeniería de software hará que generemos documentación voluminosa e innecesaria, e invariablemente nos retrasará:* La prioridad de la ingeniería de software no supone producir documentos, establece entregar productos de calidad. Su objetivo principal es establecer tiempos de entrega más cortos.

Después de comprender todo lo anterior referente a la Ingeniería de Software queda la pregunta ¿cómo iniciar el proyecto? Para ello se debe tener en cuenta que cada proyecto de software parte de una necesidad de negocio, ya sea la corrección de un fallo, la inclusión de nuevas funcionalidades, la adaptación de un sistema heredado o simplemente la necesidad de crear un nuevo sistema.

Al final, la Ingeniería de Software busca entregar productos de calidad a los clientes. Por otro lado, el trabajo del ingeniero de software no termina en la primera entrega. Debido a las necesidades cambiantes de los usuarios del producto, éste va a ser susceptible a que reciba mantenimiento para que su periodo de vida sea prolongado y siga cumpliendo con las necesidades del cliente y de los usuarios.

Definición del problema

Antes de establecer la propuesta del desarrollo de plantillas para la constructora, se había trabajado en una página web estática e independiente para cada desarrollo. Dicha página web consistía en mostrar la ubicación, descripción del proyecto, la disponibilidad de las oficinas o departamentos, una galería con el avance de obra y un formulario de contacto.

Sin embargo, la información que se presentaba en estas páginas no se alimentaba en tiempo real. Esto provocaba que la disponibilidad y avance de obra no ofreciera el estado real en determinado momento. Para ello se debía pasar por un proceso de desarrollo por cada cambio que se presentara. Aumentando los costos de desarrollo para ambas partes.

Para evitar dichos costos, se presentó la propuesta de desarrollar plantillas genéricas que presenten la misma información que las páginas webs independientes de cada desarrollo. Adjunto a dichas plantillas, se propuso el desarrollo de un CMS o Sistema de Gestión de Contenidos (Content Management System) para que el cliente pueda manejar la información que se presenta en las plantillas en tiempo real.

Análisis y metodología empleada

Antes de comenzar con el análisis del desarrollo de las plantillas y del Sistema de Gestión de Contenidos, se deben listar los requerimientos de software. Empezaré por listar los requerimientos de la plantilla web.

1. Generar una plantilla genérica que tenga las siguientes secciones:
    a. Inicio con la información general del desarrollo en particular.
        i. Banner principal.
        ii. Descripción.
        iii. Sección de servicios.
        iv. Sección de amenidades.
        v. Descripción de Redes.
        vi. Sección de disponibilidad.
        vii. Banner de publicidad.
        viii. Contacto y ubicación.
        ix. Sección de otros desarrollos.
    b. Calculadora de espacio.
    c. Simulador.
    d. Vista rápida de planes de pago.
    e. Galería con el avance de obra.
2. Funcionalidad para enviar un formulario de contacto con información básica desde el inicio.
3. Funcionalidad para solicitar una cotización desde el simulador y calculadora de espacio.
4. Funcionalidad para presentar el estado de disponibilidad de oficinas o departamentos en tiempo real.
5. Funcionalidad para presentar el avance de obra actualizado de cada desarrollo.
6. Funcionalidad para presentar el simulador de espacios cuando alguno se encuentre disponible desde la sección de disponibilidad.

A continuación, se presentarán los requerimientos del Sistema de Gestión de Contenidos:

1. Contar con el acceso de un súper usuario que maneje la información del sitio.
2. Manejar la información general de la plantilla:
    a. Nombre.
    b. Encabezados.
    c. Ubicación.
    d. Teléfonos.
    e. URL externo (páginas web anteriores)
    f. Colores de la página.
    g. Ubicación geográfica del desarrollo para su despliegue en Google Maps.
    h. Banners.
3. Edición de la descripción e imágenes del desarrollo. En esta sección se pueden incluir imágenes o video.
4. Editar el listado de los servicios que ofrece el desarrollo.
5. Editar el listado de amenidades que ofrece el desarrollo.
6. Editar y listar las certificaciones que ofrece el desarrollo.
7. Editar u ocultar el contenido de la sección “Conexión”.

8. Manejar la información de disponibilidad del desarrollo.
    a. Listar y dar de alta pisos.
    b. Listar y dar de alta espacios.
    c. Administrar el estado de cada espacio
        i. Disponible
        ii. Apartado
        iii. Vendido
        iv. En renta
9. Editar el banner de publicidad que se presenta en el sitio.
10. Listar y dar de alta los otros desarrollos que se presentan en el sitio.
11. Listar y dar de alta noticias internas que se deseen presentar a los clientes de la empresa constructora.
12. Administrar la galería de avance de obra que se presenta en el sitio para cada desarrollo.
    a. El avance de obra se puede dividir en periodos, ya sea:
        i. Trimestrales.
        ii. Mensuales.
13. Manejar los links a las redes sociales del desarrollo.

Para cumplir con dichos requerimientos se procederá a diseñar una plantilla que satisfaga dichos contenidos y que sea aplicable para cualquier desarrollo que se quiera presentar al público en general. Para presentar el estado de disponibilidad de los espacios de cada desarrollo, se utilizarán puntos de colores para identificar a cada uno.

El desarrollo se basará en el framework interno de la empresa desarrolladora de software. Dicho framework está basado en el patrón de diseño “Modelo Vista Controlador”. En cuanto a la vista, cabe destacar que se utiliza el patrón de diseño Mobile First, el cual establece que se debe partir de un diseño responsivo orientado a móviles y terminar en la versión de escritorio.

Para poder desarrollar estas vistas se utilizará el lenguaje de etiquetas HTML5 junto con las hojas de estilo CSS en su versión 3. Por otro lado, para complementar la funcionalidad que se ejecutará en las máquinas del cliente, se utilizará JavaScript. Los puntos de disponibilidad se desarrollarán con CSS y su ubicación estará dada por coordenadas dentro de la imagen del piso. Para manejar los contenidos dinámicos en tiempo real se utilizará la tecnología Ajax y se integrará por medio de la extensión jQuery. Para manejar las animaciones y layers del sitio se utilizará Bootstrap.

Todo lo anterior se refiere a la vista en donde se presentarán todos los datos de los desarrollos en cuestión. Por ende, la estructura y diseño de datos es de vital importancia para presentar la información correcta en el sitio. Para ello se diseñará un modelo de datos basado en MySQL. Fue seleccionado dicho controlador debido a su rápida integración con el lenguaje de programación PHP, utilizado en el framework interno de la empresa, además de su vasto soporte con bases de datos relacionales. La conexión de datos es de suma importancia para el proyecto debido a que se deberá identificar cuáles datos son los que pertenecen a un determinado desarrollo.

Con el fin de digerir la información que se quedará almacenada en las bases de datos del servidor y evitar que el tiempo de respuesta de la página sea tardado, se utilizará el lenguaje de programación PHP. Dicho lenguaje se encargará de ejecutar en el servidor los scripts correspondientes al filtrado de información a través de controladores de cada vista. Para optimizar los tiempos de respuesta, se integrarán un servicio web por medio de llamadas Ajax a un script de ejecución PHP, el cual se encargará de mostrar sólo la información requerida para un determinado evento.

Al final, el sitio se encontrará almacenado en un servidor con sistema operativo Linux. Las tareas de administración estarán bajo la interfaz cPanel, la cual nos permitirá administrar las cuentas, el nivel de seguridad, accesos, bases de datos y versión de PHP. Por otro lado, el control de versiones del proyecto estará bajo la administración de la tecnología GitHub a través de la cuenta organizacional de la empresa.

La metodología empleada para este proyecto será un modelo de desarrollo iterativo basado en prototipos. Los pasos por seguir para el desarrollo del proyecto serán los siguientes:

1. Junta de arranque con cliente para establecer prioridades y fechas de entrega.
2. Análisis de requerimientos y planteamiento de la solución del problema.
3. Diseño de la solución del problema.
    a. Diseño de las vistas de la plantilla web y de la versión móvil.
    b. Diseño de datos del sistema.
    c. Análisis de los componentes del Framework para su aplicación en el diseño.
    d. Diseño de nuevas clases para cumplir nuevas funcionalidades.
4. Maquetado del diseño, primera entrega de prototipo con Cliente.
    a. Página web navegable y demostración de la funcionalidad de la disponibilidad.
5. Desarrollo de la funcionalidad de la plantilla web.
    a. Desarrollo de la estructura de la base de datos.
    b. Desarrollo del CMS para administrar el contenido de la base de datos.
    c. Desarrollo de los servicios web.
    d. Desarrollo de las funcionalidades dinámicas del sitio.
6. Pruebas del sitio.
    a. Pruebas de caja negra.
    b. Pruebas de caja blanca.
    c. Pruebas beta.
    d. Pruebas de estrés.
7. Liberación en ambiente de producción.
    a. Monitoreo del tráfico de red.
    b. Pruebas de vulnerabilidad en nuevo ambiente.
    c. Implementación del plan de medición para el proyecto.

Participación profesional

Después de entender el problema, analizar los requerimientos del cliente y definir la metodología para desarrollar la solución, sigue explicar cuál fue la participación profesional. Cabe destacar que el desarrollo se manejó en distintas etapas, tal cual se presentó en la metodología, y que dichas etapas se realizaron con distintos equipos.

1. Etapa de comunicación: equipo de Asistentes de Cuentas.
2. Etapa de planificación: todos los equipos.
3. Etapa de diseño: equipo de Diseño Gráfico y equipo de Desarrollo.
4. Etapa de maquetado: equipo de Desarrollo Front End.
5. Etapa de desarrollo: equipo de Desarrollo Back End.
6. Etapa de pruebas: equipo de Desarrollo.
7. Etapa de liberación: equipo de Asistentes de Cuentas y equipo de Desarrollo.

El equipo de desarrollo se encuentra dividido en las siguientes áreas:

1. Front End.
2. Back End.
3. Pruebas y calidad.
4. Administrador de Servidores.

Cabe destacar que mi participación profesional se centró en el equipo de Desarrollo Back End. Por ende, estuve presente en las etapas de planificación, diseño, desarrollo, pruebas y liberación. Para describir dichas actividades presentaré las siguientes etapas en el listado que se muestra a continuación:

1. Análisis de la arquitectura del sitio.
2. Análisis del framework interno de la empresa.
3. Análisis de los componentes reutilizables de la empresa.
4. Análisis del flujo de datos del sitio.
5. Diseño de la estructura de la Base de Datos del sitio.
6. Diseño de los componentes nuevos requeridos para la solución.
7. Desarrollo de código.
8. Pruebas y resolución de fallas.
9. Plan de liberación en producción.

Empezaré por presentar la arquitectura del sitio con el objetivo de entender los responsables en cuanto a las actividades de desarrollo del sitio.

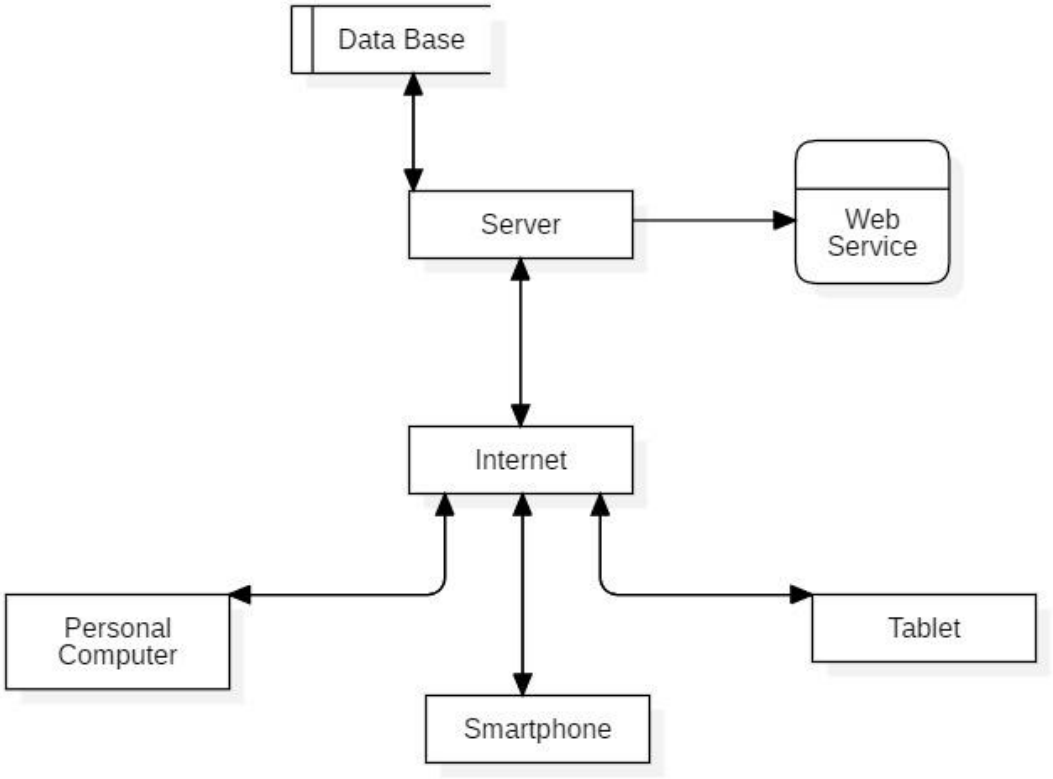

Imagen 5.1 Arquitectura del sistema

1. Base de datos: Back End.
2. Servidor: Administrador de servidores.
3. Web services: Back End.
4. Terminales (escritorio y dispositivos móviles): Front End.

Ya entendida la arquitectura de la solución del desarrollo, es preciso analizar cómo está compuesto el framework interno de la empresa. Este framework se basa en el patrón de diseño Modelo Vista Controlador. Dicho framework establece una interfaz para manejar los modelos de datos del sitio. También establece que cada vista debe tener un controlador asociado, el cual se encargará de obtener la información correspondiente de su respectiva vista.

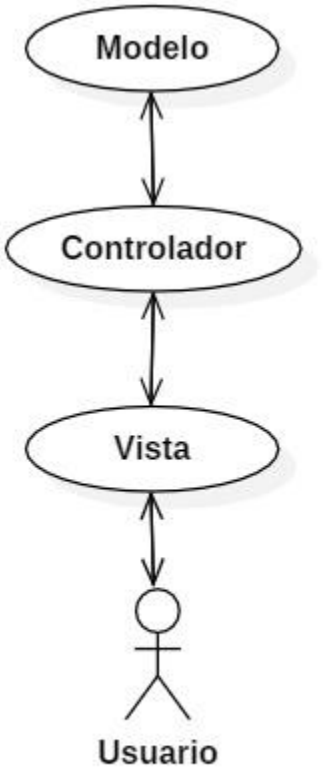

Imagen 5.2 Patrón de diseño.

Una vez teniendo en cuenta la arquitectura con la que se va a trabajar, pasaré a analizar los componentes del framework de la empresa. Para el CMS tenemos un manejador de subida de archivos. Para el sitio tenemos un manejador de envío de correos, una interfaz de comunicación hacia la base de datos y un manejador general para procesos comunes del sitio y CMS. Así como un manejador para la consola con el fin de depurar el código PHP en cualquier navegador. Los diagramas de clases se presentan a continuación:

Imagen 5.3 Diagrama de clases UML del Framework

Por otro lado, el framework cuenta con configuraciones predeterminadas de seguridad, de base de datos y navegación del sitio. Este tipo de configuraciones permiten identificar el protocolo por el cual el cliente se trata de conectar. Es capaz de identificar si es posible conectarse, idealmente, a través del protocolo https. Además, identifica el nombre del servidor en el que está alojado el proyecto con la finalidad de establecer un determinado ambiente: pruebas o producción. Ya teniendo establecido el ambiente determinado, define las variables de conexión a la base de datos, en este caso existen dos tipos de bases de datos: producción y pruebas. Tener ambientes separados nos ayuda en el proceso de mantenimiento, cuando se levante un nuevo requerimiento primero se desarrollará en pruebas, se validará y una vez aprobado, se libera en el ambiente de producción.

Después de analizar los componentes del framework y las configuraciones predeterminadas del software, es preciso analizar el flujo de navegación del proyecto. Para entender mejor el flujo final que tendrá el sitio web, procederé a presentar el mapa del sitio. Dicho mapa servirá para representar de una manera gráfica las relaciones en cuanto a navegación del propio sitio. Así como tener un bosquejo del flujo de datos del sistema.

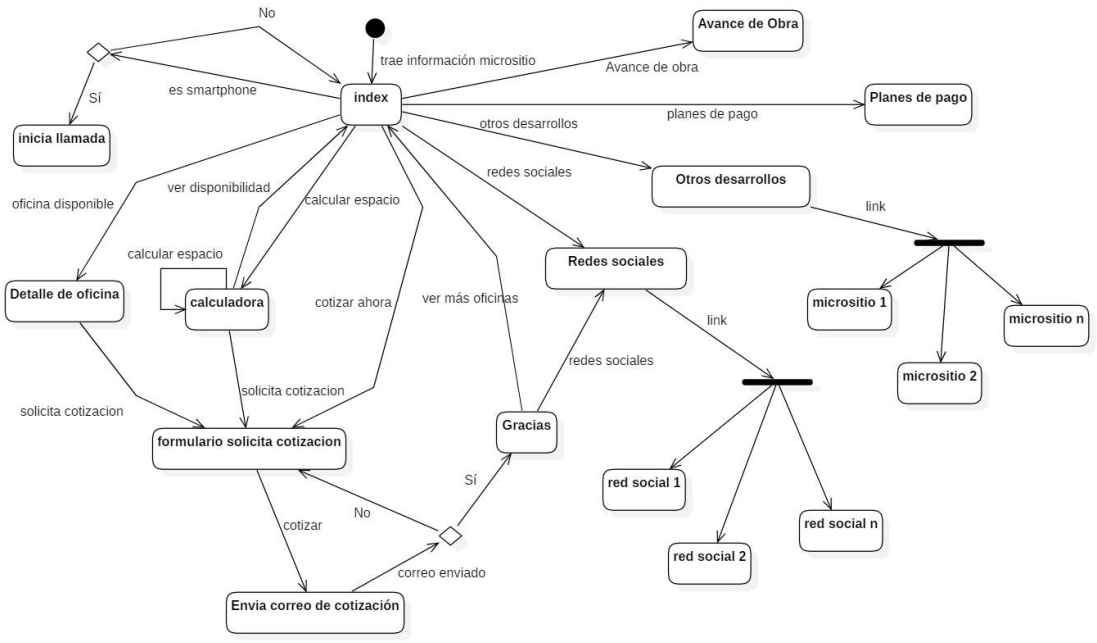

Imagen 5.4 Diagrama de flujo del sitio

Después de analizar el comportamiento del sitio, procederé a analizar el comportamiento esperado para el CMS. Esto con la finalidad de también tener una guía para empezar con el diseño de la base de datos del sistema, debido a que el CMS será la interfaz principal entre el administrador y la base de datos.

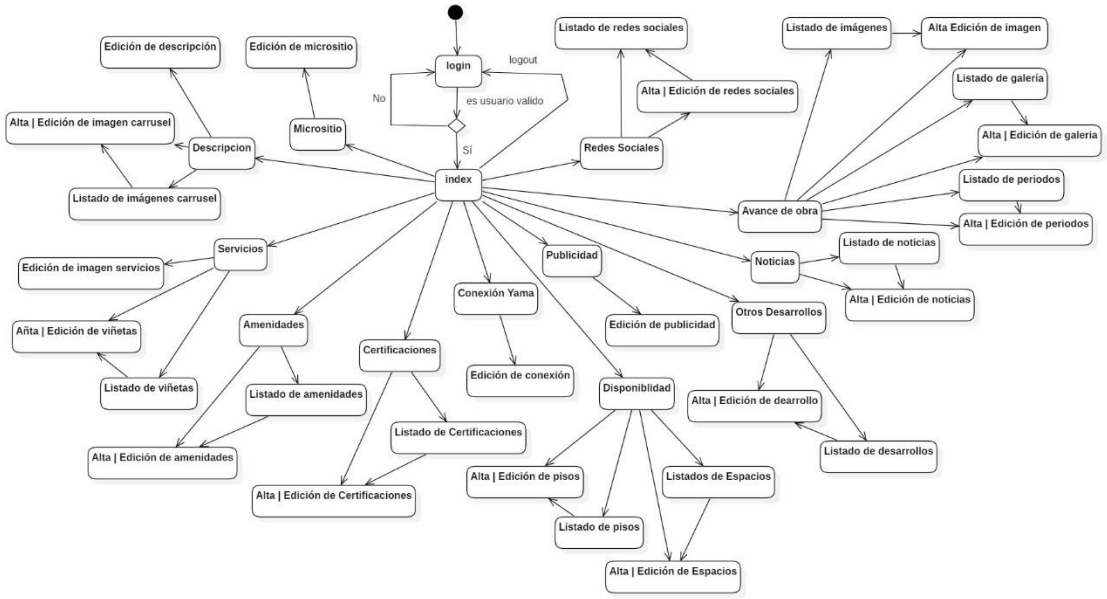

Imagen 5.5 Diagrama de flujo del CMS

Comparando los mapas, es notable la diferencia de tamaño entre el sitio y el CMS. Eso es razonable debido a que todo el manejo de datos, administración de contenido y administración de estado de disponibilidad se hace a través del CMS. El sitio solo se encarga de presentar dicha información y recabar datos de posibles clientes.

Entendiendo mejor el flujo de datos que se presentará en el sistema, ya es posible iniciar con el diseño del modelo de datos del sitio. Como ya se había comentado previamente, el modelo de datos se desarrollará para un manejador de base de datos MySQL. El diagrama se presenta a continuación:

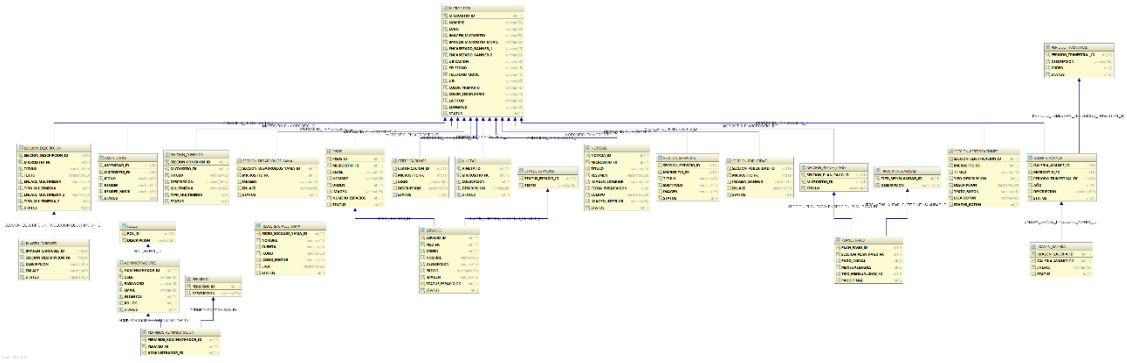

Imagen 5.6 Diagrama lógico de la Base de Datos

Estos diagramas me ayudaron a la hora de escribir el código fuente del sistema. Para resumir este proceso escribiré los pasos que se llevaron a cabo para realizar el proyecto en un pseudocódigo generalizado:

CMS:

1. Obtener login y password del usuario
2. Verificar información de la base de datos
    a. Login incorrecto, regresar a la pantalla de inicio con mensaje de error
    b. Login correcto, pasar a la pantalla index del CMS
        i. Inicializar CMS apuntando al registro de la base del micrositio actual
        ii. Detectar acción del usuario
            1. En caso de ser listado:
                a. Filtrar la información correspondiente de la tabla deseada
                b. Verificar los permisos del usuario y desplegar los botones de edición o eliminar
                    i. Editar: mandar a formulario de alta con id de registro.
                    ii. Borrar: actualizar el status del registro a 0
            2. En caso de ser alta de registro:
                a. Desplegar el formulario correspondiente para dar de alta el nuevo registro de base de datos.
                b. Si es necesario subir una imagen, guardar en el servidor con un nombre único para evitar sobreescribir imágenes.
                c. Guardar registro en base de datos.
        iii. En caso de click en acción de logout:
            1. Destruir la sesión actual.
            2. Regresar a pantalla de login.

Sitio:

1. Obtener información del micrositio de la base de datos.
    a. Obtener textos.
    b. Obtener links a imágenes.
    c. Obtener pisos.
    d. Obtener disponibilidad de espacios.
    e. Vaciar en maqueta.
2. Vacía información en la maqueta del micrositio.
3. Verificar variable de sesión para despliegue de layer:
    a. Existe variable
        i. Tiene valor verdadero: No despliega layer.
        ii. Tiene valor falso: Despliega layer y actualiza a verdadero.
    b. No existe variable
        i. Crea variable con valor verdadero y despliega layer.
        ii. Si sucede un error, actualiza la variable a falso.
4. Escucha eventos:
    a. Click en botón de teléfonos
        i. Verifica si existe aplicación para hacer llamadas
            1. Existe: realiza la llamada
            2. No existe: no realizar acción y regresar al index.

b. Enlace a calculadora:
    i. Navega a vista de calculadora
    ii. Pide los siguientes parámetros:
        1. Directivos.
        2. Gerenciales.
        3. Operativos.
        4. Salas de juntas
        5. ¿Tiene recepción?
    iii. Calcula espacio con los parámetros que nos proporciona cliente.
    iv. Identifica acción del usuario.
        1. Ver disponibilidad: regresa a index con ancla a la sección de disponibilidad.
        2. Solicita cotización:
            a. Abre formulario de cotización con el resultado del cálculo.
            b. Cotizar
            c. Mandar a llamar el servicio de envío de cotización.
c. Cambio de piso en la sección de disponibilidad
    i. Manda a llamar el servicio web para traer los datos del piso.
        1. Vacía en maqueta la imagen del piso.
        2. Vacía en maqueta los puntos de disponibilidad con su respectivo status.
d. Click en punto de disponibilidad en status de disponible.
    i. Manda a llamar servicio web para traer los datos de ese espacio.
    ii. Vacía en maqueta la información.
    iii. Crea enlace para el simulador de ese espacio.
        1. El enlace manda a llamar el formulario de contacto.
        2. Precarga la información del espacio.
        3. Cotizar.
        4. Manda a llamar el servicio de envío de cotización.
e. Click en el envío de cotización en index
    i. Manda a llamar el servicio de envío de cotización
f. Servicio de envío de cotización
    i. Valida que los campos no se encuentren vacíos
    ii. Obtiene el Template HTML con el cuerpo de la cotización.
    iii. Descarga los datos del formulario en la plantilla.
    iv. Envía el correo a los administradores del sitio.
        1. Error: manda alerta en el sitio avisando que hubo error en el envío.
        2. Enviado: manda a la pantalla de Gracias
            a. En la pantalla de gracias se tiene la opción de ir a las redes sociales de la empresa constructora.
            b. Regresar al inicio.
g. Sección avance de obra
    i. Manda a llamar el servicio para descargar la última galería dada de alta.
    ii. Descarga los encabezados de las galerías dadas de alta.
    iii. Si se da click en el encabezado del periodo requerido:
        1. Limpia las imágenes de la vista.
        2. Descarga las imágenes del periodo elegido.

Acabada la etapa de desarrollo, se procedió a iniciar con la etapa de pruebas. El equipo de pruebas realizó las pruebas pertinentes en distintos sistemas operativos y resoluciones. El sitio de pruebas se alojó en un servidor CentOS 6, con versión PHP 5.3 y base de datos MySQL 5.5. Los errores que se fueron encontrando se corrigieron de manera pertinente antes de liberar en producción.

Ya validado el sitio, se procedió a desarrollar un plan de medición con el equipo de Google Analytics. Ya que el plan fue desarrollado y aprobado, se procedió a incluirlo dentro del desarrollo. Dicha medición nos permite saber cuántas veces se dio click en cotizaciones, galerías, espacios, pisos, enlaces a redes sociales y enlaces a otros desarrollos. Así como el número de visitas a las vistas del sitio.

Para la liberación, el sitio se encontraría hospedado en los servidores del cliente. Para ello se nos pidió conceder todas las especificaciones técnicas del sitio. Para no tener mayores inconvenientes, se solicitó tener las mismas características que el servidor de pruebas. La transferencia de datos se manejó por medio de transferencia ftp entre el servidor de pruebas y el servidor de producción. Por otro lado, la migración de la base de datos se realizó con la herramienta MySQL Workbench.

Resultados obtenidos

El proyecto se culminó en un total de un mes de análisis, tres semanas de diseño, dos semanas de maquetación, dos semanas de programación, una semana de pruebas y una semana para la liberación. Actualmente, el sitio se encuentra operando y por nuestra parte seguimos ofreciendo mantenimiento.

En total se han puesto en producción 3 micrositios de la empresa constructora. Se espera que por mes se levanten 2 micrositios con ayuda de la plantilla, con la espera de que para la mitad del presente año estén en producción los 10 micrositios previstos. Sin embargo, debido a las exigencias de los clientes, el sitio sigue obteniendo mejoras constantes para satisfacer las necesidades del mercado.

A continuación, se presentan capturas de pantalla del micrositio de un determinado centro. Cabe destacar que los logos de la empresa, así como el logo del centro, fueron ocultaron en las imágenes de muestra.

Sitio:

Imagen 6.1 Index

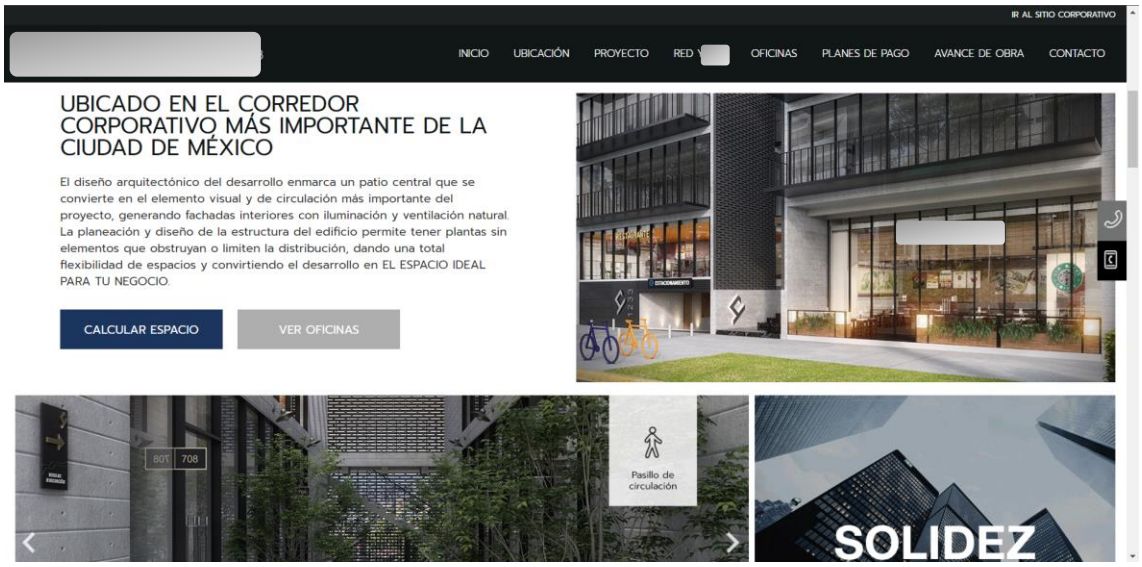

Imagen 6.2 Descripción

Imagen 6.3 Servicios

Imagen 6.4 Redes

Imagen 6.5 Disponibilidad

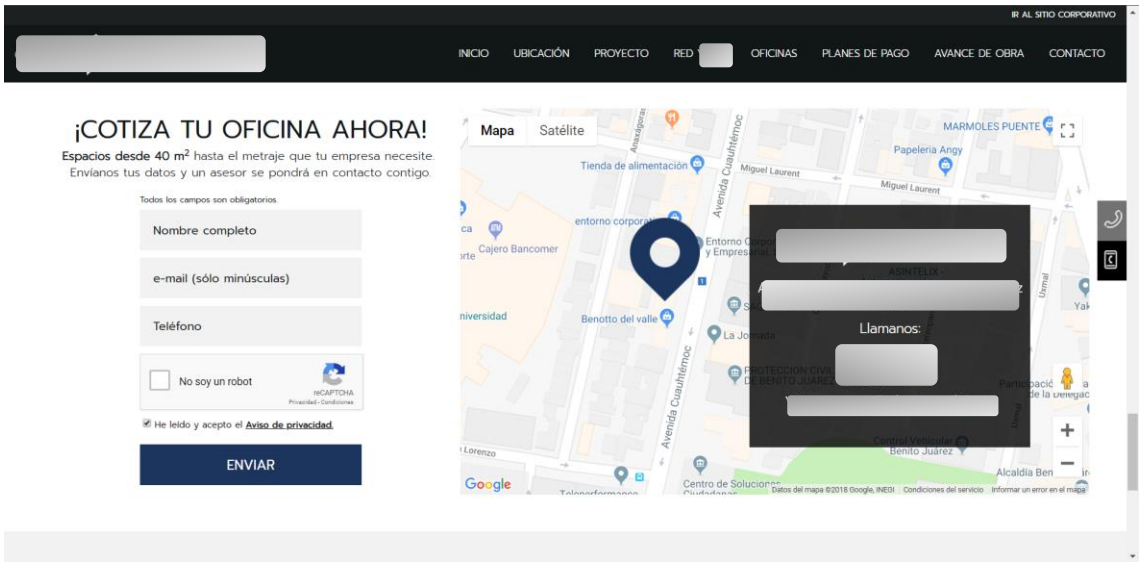

Imagen 6.6 Cotización rápida

Imagen 6.7 Otros Desarrollos

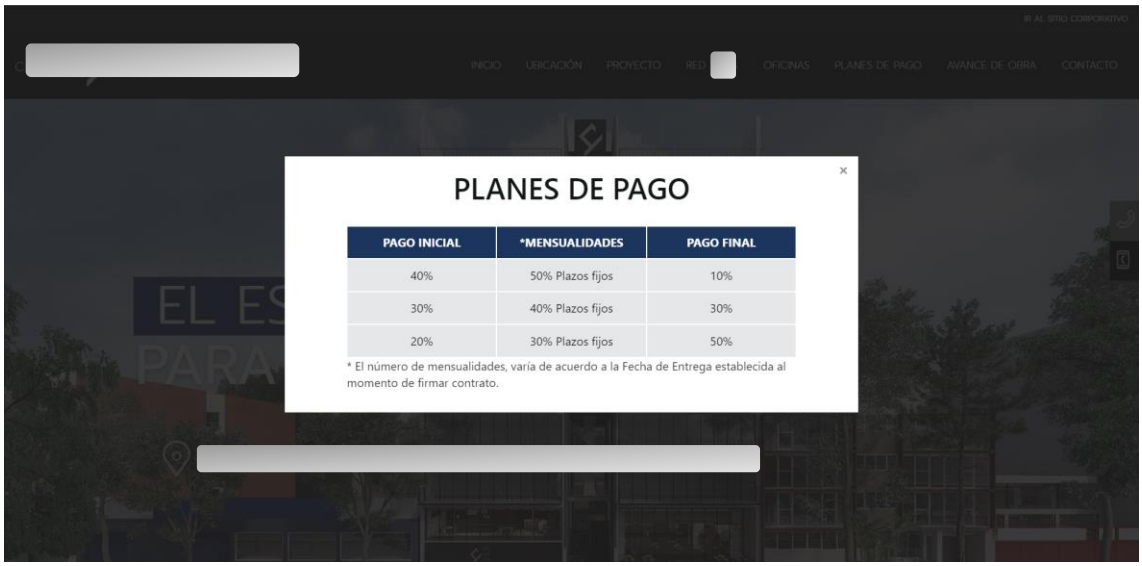

Imagen 6.8 Planes de pago

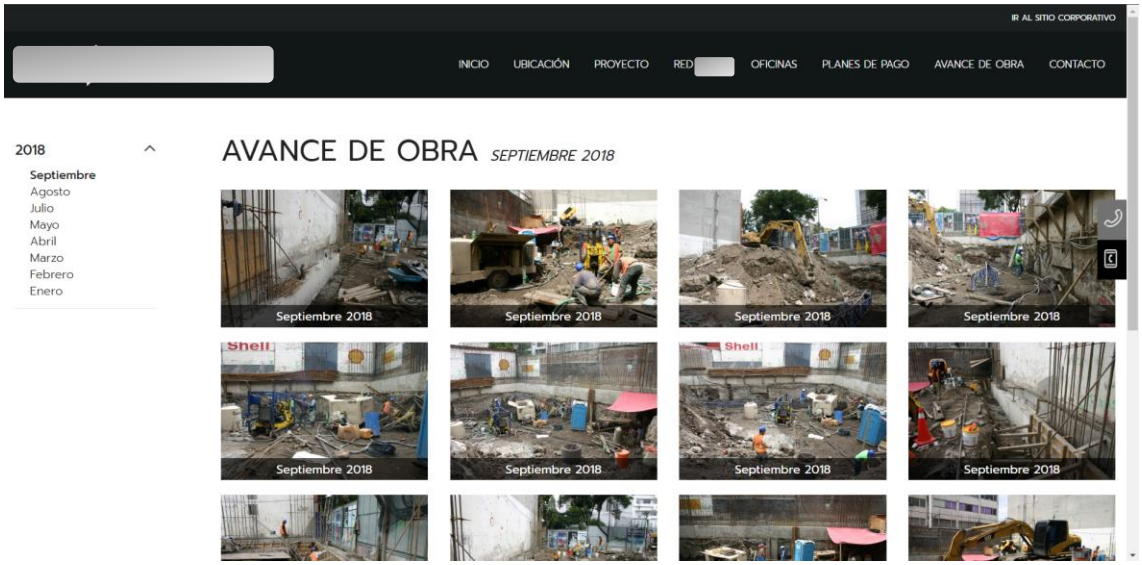

Imagen 6.9 Avance de obra

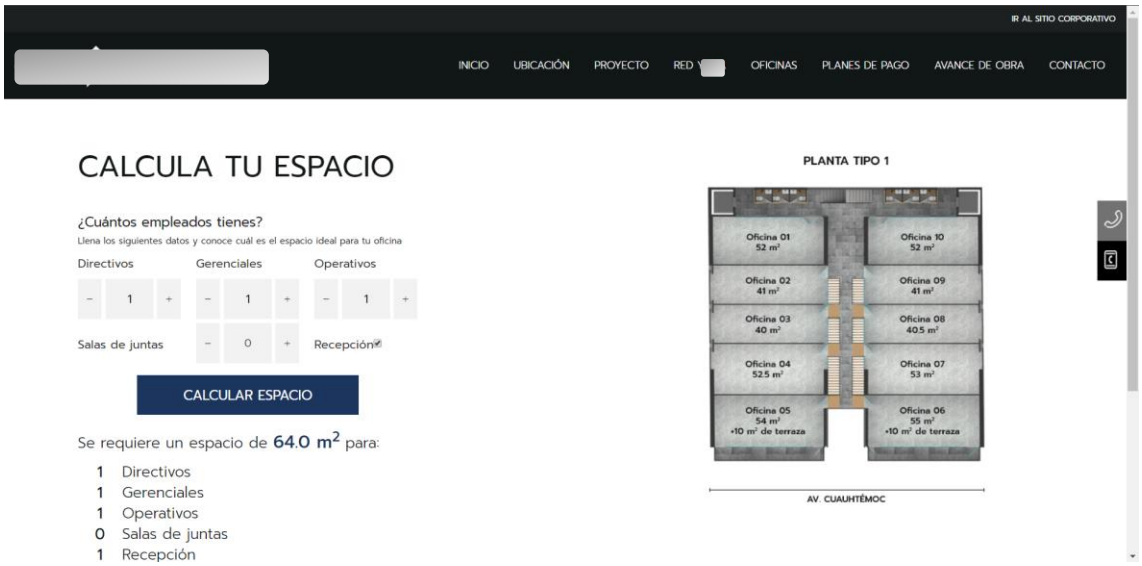

Imagen 6.10 Calculadora

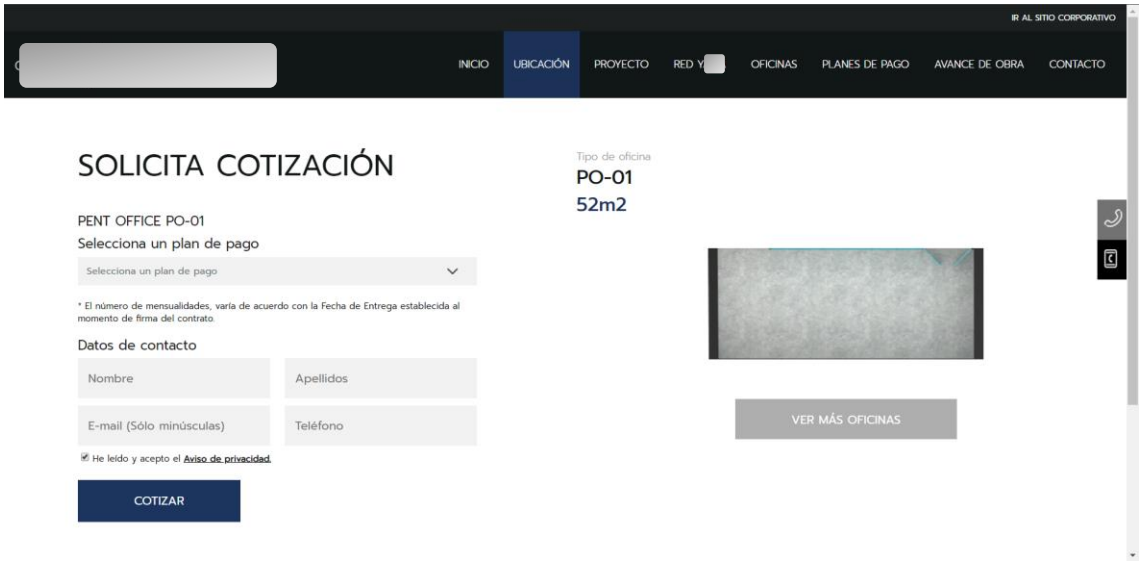

Imagen 6.11 Simulador

Imagen 6.12 Gracias

CMS

Imagen 6.13 Navegación del CMS

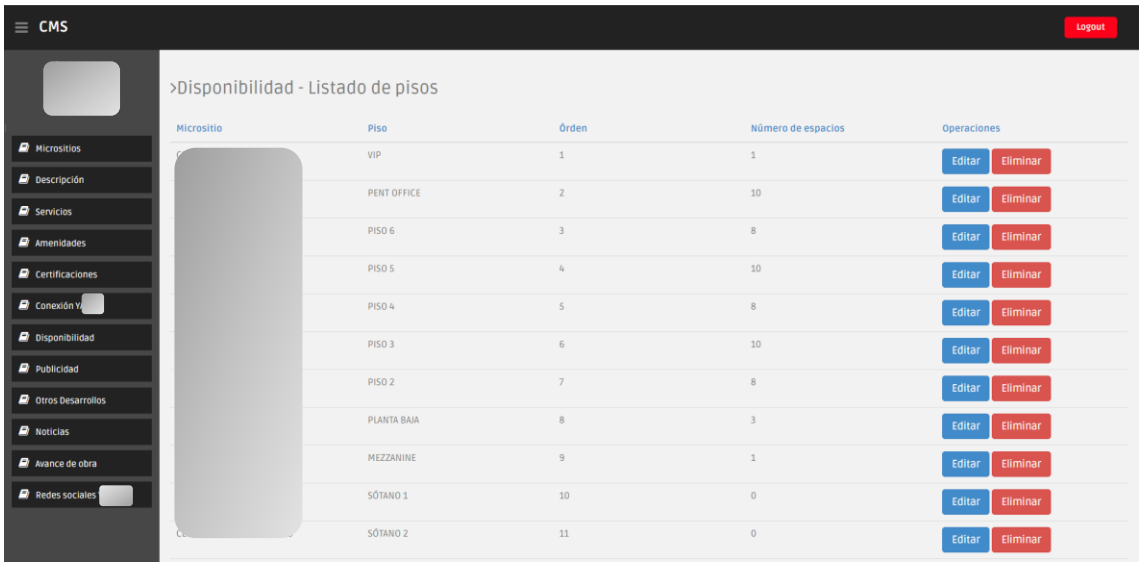

Imagen 6.14. Listado de secciones

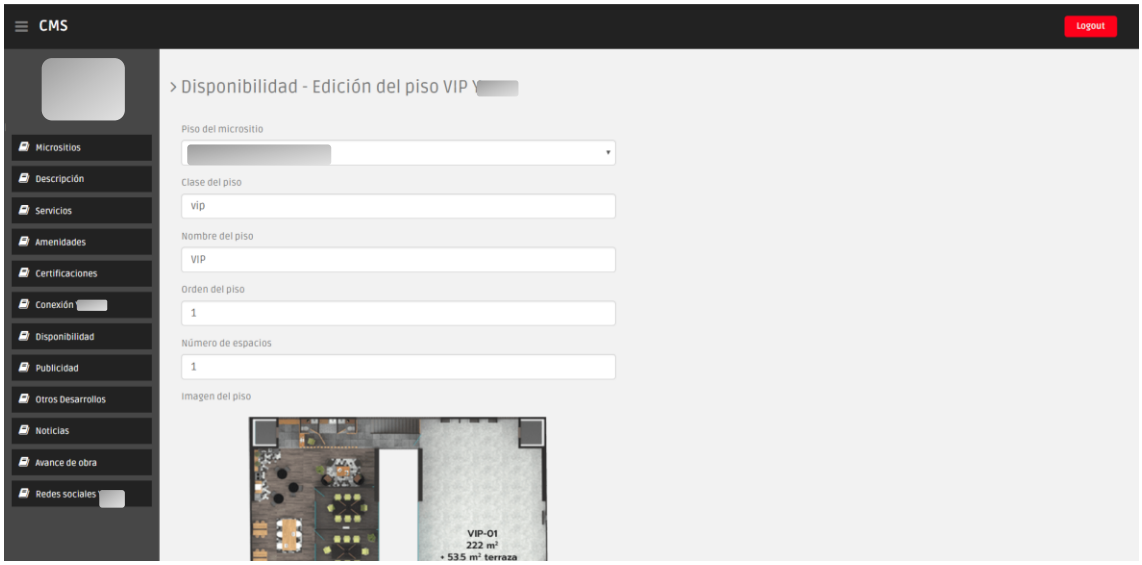

Imagen 6.15 Alta/Edición de sección

Conclusiones

Para poder entregar software de calidad en tiempo y presupuesto definido, es imprescindible emplear la metodología que declara la Ingeniería de Software. Como primer paso, es necesario entender las necesidades del cliente. Comprender los requerimientos del cliente no es una tarea sencilla, por ello es de suma importancia establecer una comunicación óptima entre el cliente y el equipo de trabajo.

Ya teniendo en claro las reglas de negocio, es posible establecer un plan para determinar la prioridad de las tareas para resolver las necesidades del negocio. Ya teniendo un plan, se puede diseñar la estrategia de solución para resolver los requerimientos establecidos. Con este paso es posible avanzar en el desarrollo del proyecto sin perder la línea temporal establecida en la etapa de comunicación.

Con las dos etapas anteriormente resueltas, la etapa de desarrollo se convierte en una tarea sencilla de llevar a cabo. Esto es debido a que se tiene una guía para empezar a escribir código fuente, sin la necesidad de generar retrasos en el proyecto debido a una mala administración. Cabe mencionar que es imprescindible detectar y resolver errores en cada etapa del desarrollo para evitar un mayor conflicto en los entregables.

Después, teniendo un producto funcional y operable, es necesario realizar pruebas para detectar errores antes de pasar a un ambiente de producción. Las pruebas nos permiten asegurar que se entregarán productos de calidad de acuerdo con los requerimientos iniciales del cliente.

Al final, después de la etapa de liberación, es preciso tener en cuenta que nuestro desarrollo será susceptible a recibir un mantenimiento después de la primera entrega oficial. Sabemos que las necesidades del negocio cambian con respecto al tiempo. Si nuestro software se comprendió desde un inicio, se diseñó de manera correcta, se construyó con la mentalidad de ser sencillo de entender y se detectaron los fallos a su tiempo, el software será fácil de mantener y seguirá siendo vigente por un largo periodo de tiempo.

La ingeniería de software logra su objetivo cuando tenemos una buena retroalimentación del cliente, cuando el sistema es intuitivo para el usuario o cuando las métricas del sistema indican que hubo un incremento de ventas. Se logra cuando los programadores posteriores entienden la mayoría del código fuente, cuando se encuentra un error antes de pasar a una etapa posterior o cuando el sistema no sólo funciona, aparte cubre las necesidades de los usuarios. Incluso, lo más importante, cuando surge una nueva necesidad de negocio, no es necesario hacer un rediseño de todo el sitio para satisfacerla.

Bibliografía

Pressman, Roger S. (2010). Ingeniería de software. México: McGraw-Hill.

Robertson, Suzanne. (2013). Mastering the requirement process: getting requirements right. USA: Addison-Wesley.

Fontela, Carlos. (2011). UML modelado para profesionales. México: Alfaomega Grupo Editor Argentino.

Kaufmann, Morgan. (2014). Relating system quality and software architecture. Netherlands: Elsevier.