UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo de aplicación Web Onboarding de Factoraje Financiero

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Rogelio López Tendero

ASESOR DE INFORME

Ing. Jorge Alberto Solano Gálvez

Ciudad Universitaria, Cd. Mx., 2019

Índice de contenido

Introducción y objetivo

El presente informe tiene como objetivo explicar cómo se han aplicado los conocimientos teóricos y prácticos adquiridos en la carrera de Ingeniería en Computación para generar soluciones de software a problemas presentados en el ámbito laboral.

El proyecto de software utilizado como referencia en este documento lleva por título “Onboarding de Factoraje Financiero” y tiene como objetivo administrar y automatizar todas las actividades que realiza un prospecto interesado para convertirse en cliente de la financiera.

A lo largo del documento se presenta cada una de las fases del desarrollo de software en las que se ha tenido participación directa, desde el análisis de requerimientos hasta el despliegue productivo de la aplicación. En cada fase se explica a detalle las técnicas y procedimientos empleados para la obtención de una determinada solución.

El proyecto desarrollado tiene interacción parcial con otros procesos y desarrollos internos de la organización, de los cuales solo se presentan los aspectos fundamentales para comprender el contenido de este informe.

El documento no pretende ser una guía de como adquirir los servicios financieros de la empresa, tampoco pretende exponer ningún tipo de información sensible, por lo cual algunas partes del proceso, así como nombres técnicos han sido modificados u omitidos a petición de la empresa y la explicación de conceptos financieros presentados se limita únicamente a los necesarios para entender el proceso de Ingeniería involucrado.

Capítulo 1
La empresa

1.1 Historia

Flux Financiera es una institución financiera privada que nace en el año 2011 dedicada principalmente a proporcionar financiamiento a PYMES mediante un recurso llamado factoraje financiero, el cual consiste en dar liquidez inmediata a las empresas a cambio de los derechos de cobro de sus facturas para permitirles continuar su operación.

En nuestro país se creía que el factoraje solo estaba al alcance de grandes compañías, pues el sistema bancario se enfoca únicamente en ese sector, dejando a un lado a las Pymes, sin embargo es en estas últimas donde Flux Financiera ve un nicho de mercado potencialmente valioso con el cual puede entablar relaciones comerciales y es con estas con quien Flux inicia sus operaciones, siempre respaldada por un atractivo modelo de negocio que le permite ofrecer múltiples ventajas competitivas a sus clientes.

Con el paso de los años la cartera de clientes de Flux comenzó a crecer y con ello la cantidad de procesos y operaciones que debían gestionar, los cuales hasta ese momento se hacían de forma manual. Conscientes de que la era digital empezó hace ya varios años, la dirección decide iniciar una división de tecnología en la cual se desarrollarían todas las herramientas necesarias para soportar su crecimiento. A partir de aquí la empresa se fijó un nuevo objetivo: convertirse en una financiera tecnológica que revolucionará la forma tradicional de hacer negocios.

Al igual que otros sectores, el financiero es particularmente conservador y es por esta razón que llevar a cabo dicha transformación no fue una labor sencilla, pues algunas de sus empresas clientes tenían dudas respecto a la digitalización de sus procesos.

Al mismo tiempo que la financiera crecía, también lo hacía su división tecnológica y es en este punto donde se presenta la oportunidad de incorporarme a esta organización. El área continuó su crecimiento al punto de consolidarse como una empresa en el año 2017 llamada Prometheus dedicada a brindar soluciones de software para el sector financiero teniendo como principal cliente a Flux Financiera.

Hoy en día Prometheus le brinda todos los servicios de software a Flux Financiera necesarios para poder ofrecer una operación 100% online, sin papeleo innecesario y con respuesta inmediata.

1.2 Misión

Flux Financiera: Ser un actor relevante en el desarrollo de la economía y la competitividad de México. Impulsar la economía mexicana, transformando el sector financiero para convertirnos en socios estratégicos de nuestros clientes y proveedores.

Prometheus: Ofrecer soluciones tecnológicas a nuestras clientes diseñadas a la medida de sus necesidades y que les permitan incrementar su productividad y competitividad.

1.3 Visión

Flux Financiera: Generar valor al incrementar el flujo de capital a las PYMES por medio de dar certidumbre al proceso de factoraje y así empoderar las bases para la creación de un mercado en el cual haga eficiente la transacción de las cuentas por cobrar.

Prometheus: Convertirnos en partners estratégicos de nuestros clientes gracias a las soluciones tecnológicas que ofrecemos, siempre contribuyendo en su crecimiento y fortaleciendo relaciones comerciales duraderas.

Seguiremos construyendo nuestro futuro, siendo una empresa competitiva que ofrece servicios de TI de calidad, reconocida en México y con presencia en Latinoamérica, por las soluciones tecnológicas que entregamos, generando relaciones duraderas con nuestros clientes, proveedores y nuestra gente. [1]

[1] Misión y visión obtenidos de documentos internos de la compañía. Se omite información de estos documentos por motivos de confidencialidad

1.4 Organigrama

Anteriormente Prometheus era un área de Flux Financiera, sin embargo, ahora es considerada una empresa independiente. La estructura de las dos empresas se muestra en la imagen 1.1 y la imagen 1.2

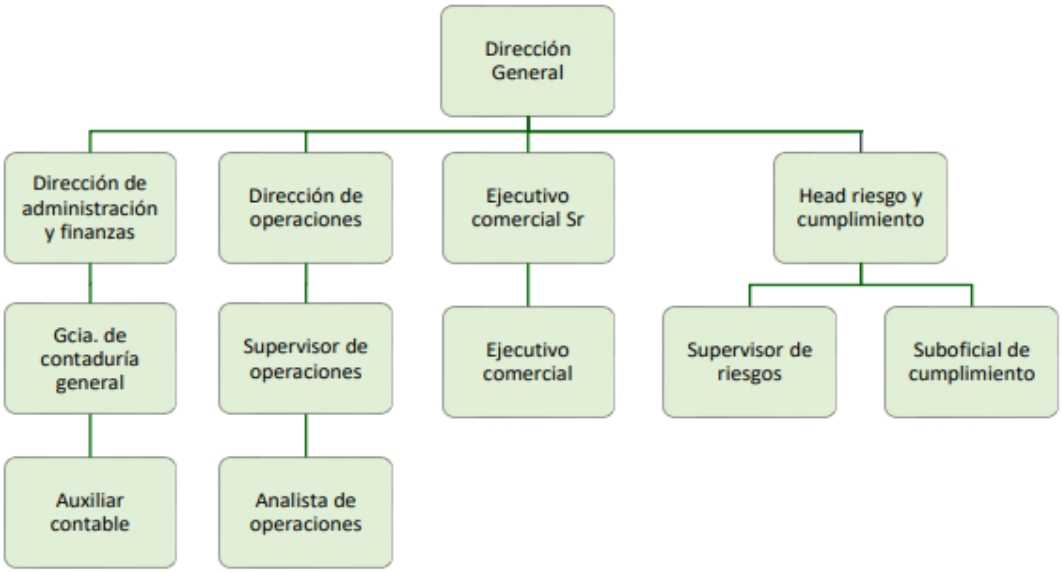


Imagen 1.1 Organigrama de Flux Financiera

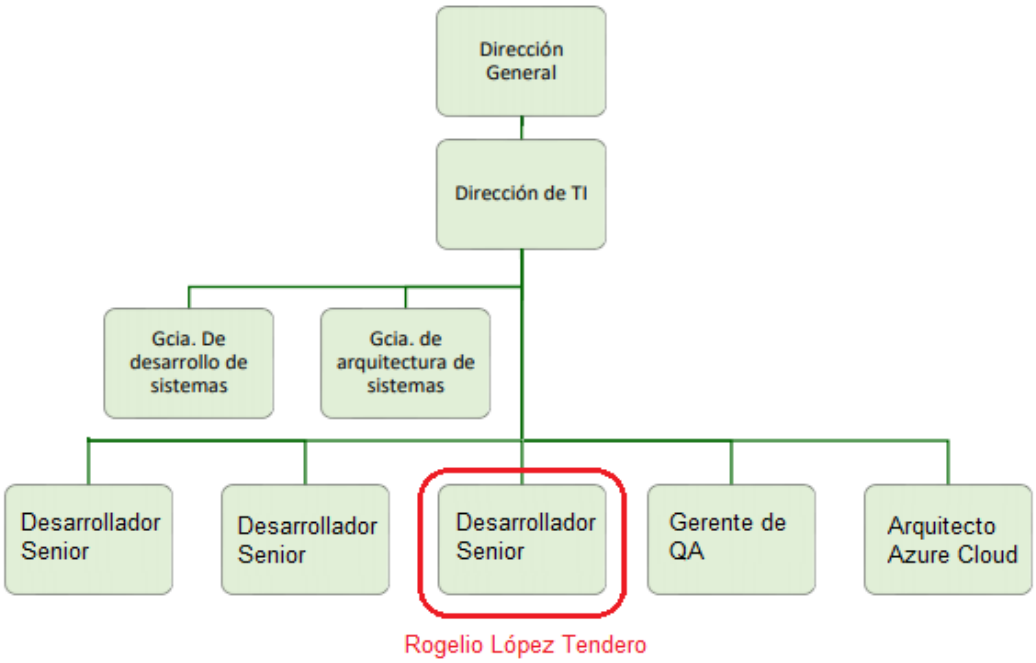


Imagen 1.2 Organigrama de Prometheus

1.5 Perfil del puesto

En este proyecto yo me desempeñé como desarrollador senior, cubriendo los siguientes requisitos y funciones:

- Construir o desarrollar aplicaciones de negocio basadas en los requerimientos funcionales de negocio.
- Proveer el correcto mantenimiento a las aplicaciones de negocio.
- Proveer soporte a las aplicaciones productivas.
- Identificar proactivamente los requerimientos NO funcionales de los aplicativos y dar soluciones integrales.
- Participar proactivamente en la generación de planes de trabajo y en el seguimiento de estos.
- Participación activamente en cada una de las fases del desarrollo de software.
- Promover la innovación tecnológica.
- Cuidar y mejorar los estándares de desarrollo.
- Liderar a equipos de desarrollo.
- Conocimientos sólidos para Programación en C# MVC.
- Entity Framework 6+.
- WCF, SOAP, Web Api.
- XML, JSON.
- REACT / REST.
- Manejo avanzado en base de datos: SQL Server, MySql.
- Manejo avanzado de HTML5, CSS3, JAVASCRIPT, AJAX, SOA.
- Conocimientos avanzados en Office y Visio.
- Conocimientos en Linux (deseable).
- Conocimiento en metodologías de desarrollo de software y metodologías ágiles.
- Microsoft Azure, Google Cloud.
- Experiencia en desarrollo de aplicaciones móviles (deseable).

Adicionalmente fue necesario demostrar otras habilidades:

- Buen comunicador entre las áreas de negocio y las áreas tecnológicas
- Buen administrador de tiempo para cumplir con los deadlines de los proyectos
- Comprensión del trabajo técnico a realizar a partir de los requisitos o
- solicitudes de negocio
- Trabajo bajo presión

Capítulo 2
Marco histórico

2.1 Factoraje y su impacto

Para comprender mejor el factoraje y cuál es su importancia es necesario poner más claro el contexto en el que se encuentra. En México cuando una empresa ofrece sus servicios y/o productos a sus clientes, éstos pueden realizar el pago correspondiente de forma inmediata o en su defecto aplazar la fecha de pago a una cierta cantidad de días posteriores a la fecha de entrega, por ejemplo 30, 60 o 90 días, este número puede variar, pero siempre es indicado en la factura de la transacción.

Debido a que la empresa no puede disponer de ese capital hasta la fecha pactada, ésta debe buscar alguna forma de financiar sus operaciones, pagar a sus proveedores, cubrir la nómina de sus empleados, entre otras cosas, sin embargo solventar esta demanda de gastos durante tanto tiempo no siempre es posible sobre todo si se trata de una Pyme, por lo cual se frena drásticamente su crecimiento y en muchos de los casos las lleva al extremo de tener que cerrar un negocio por impagos.

Afortunadamente existen diferentes métodos de financiamiento para las empresas, siendo el factoraje una opción muy atractiva. De acuerdo con la Comisión Nacional Bancaria y de Valores (CNBV) el factoraje es definido de la siguiente forma:

"*Adquisición de créditos provenientes de ventas de bienes muebles, de prestación de servicios o de realización de obras, otorgando anticipos sobre tales créditos, asumiendo o no sus riesgos. Por medio del contrato de factoraje un comerciante o fabricante cede una factura u otro documento de crédito a una empresa de factoraje a cambio de un anticipo financiero total o parcial. La empresa de factoraje deduce del importe del crédito comprado la comisión, el interés y otros gastos.*"

En otras palabras, el factoraje es un recurso financiero que permite a las empresas obtener liquidez de forma inmediata mediante la cesión de sus facturas a un tercero, por ejemplo, Flux Financiera, y utilizar estos recursos principalmente para cubrir gastos de operación, desarrollo y oportunidad de nuevos negocios.

Aunque existen algunas variantes entre la forma de ofrecer estos servicios, el flujo convencional del factoraje se puede representar en la imagen 2.1

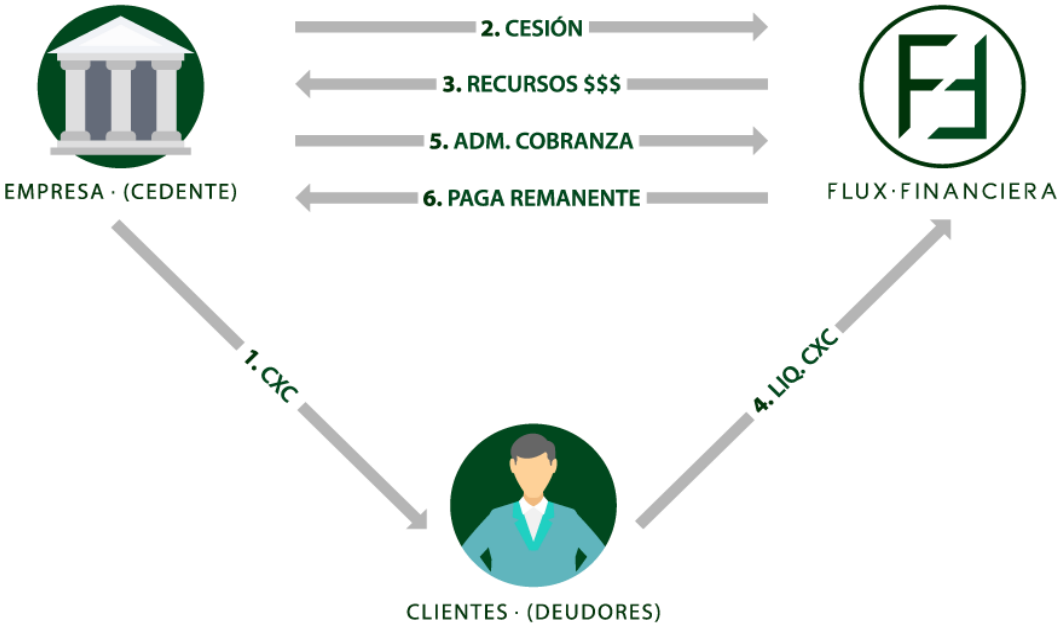


Imagen 2.1 Flujo normal de transacción de factoraje

1.-La empresa (cedente) emite facturas a sus clientes (deudores) por algún concepto de productos o servicios, la factura tiene un plazo de pago y se generan cuentas por cobrar.

2.- La empresa cede los derechos de cobro de sus facturas a la financiera mediante un contrato de factoraje.

3.- El cedente obtiene los recursos correspondientes a sus facturas, reteniendo una porción como seguro en caso de un impago, adicionalmente se descuentan algunos montos correspondientes a comisiones y gastos de operación.

4.- El deudor liquida la totalidad de las cuentas por cobrar a la financiera después de transcurrir el plazo pactado.

5.- El cedente indica a la financiera cómo debe gestionar los recursos proporcionados para administrar la cobranza de facturas.

6.- La financiera regresa al cedente todos los montos restantes que pueda haber, incluyendo el monto tomado como seguro.

2.2 Proceso Onboarding

Cuando una empresa está interesada en utilizar los servicios de factoraje de la financiera es necesario que pase por un proceso de revisión en el cual debe proporcionar cierta información a la financiera para saber si es posible otorgarle un financiamiento. Durante este proceso a la empresa solicitante se le llama **Prospecto** y el proceso de incorporación a la empresa es llamado **Onboarding.**

El proceso Onboarding incluye diferentes etapas las cuales se irán explicando más a detalle a largo de este documento. A continuación, se describe de forma general las etapas que lo conforman.

Registro de Prospecto: La empresa interesada en los servicios de factoraje debe acceder al portal de la financiera para llenar un formulario de registro con sus datos generales y de contacto. Con esto queda registrado y puede comenzar su proceso de Onboarding

Documentación Inicial: El prospecto proporciona algunos documentos básicos para comenzar un análisis y para probar su identidad, por ejemplo, el RFC de la empresa prospecto, el comprobante de domicilio y una identificación del representante.

Proporcionar CIEC: El prospecto autoriza a la empresa para hacer uso de su CIEC (Clave de Identificación Electrónica Confidencial) la cual es una firma electrónica que sirve como llave de acceso en los servicios electrónicos que brinda el SAT. Este acceso es únicamente con fines de consulta.

Análisis de Facturas: La empresa hace un análisis de las facturas que el prospecto tiene ante el SAT, basado en su modelo de negocio, empieza a revisar cuales de los clientes del prospecto son atractivos y pueden ser considerados.

Revisión PLD: Con los datos proporcionados hasta el momento se procede a realizar una búsqueda de la empresa y de sus representantes referente a PLD (Prevención de lavado de dinero) para corroborar que sea una empresa sin actividades ilícitas.

Consulta Buró de Crédito: El prospecto autoriza a la empresa la revisión de su buró de crédito, con el objetivo de conocer más las relaciones crediticias del prospecto y poder conocer su posible comportamiento en una nueva relación de crédito.

Investigación Interna: Complementa la información recabada por el buró de Crédito y ayuda a conocer mejor al prospecto.

Solicitud de Crédito: Si los puntos anteriores se completan de manera correcta y se ha observado que el prospecto tiene clientes potenciales y no presenta problemas legales o de cualquier índole el prospecto procede a llenar una solicitud de crédito.

Documentación Complementaria: El prospecto debe proporcionar documentos adicionales como son los estados financieros, declaraciones anuales, entre otros.

Autorización del Comité: Con toda la información previamente analizada por las diferentes áreas de la empresa y una vez que se acordó que se trata de un cliente potencial, el caso del prospecto es sometido al comité de la empresa el cual determina si su solicitud es aprobada o rechazada.

Contrato de Factoraje: En caso de ser aprobada la solicitud del prospecto se procede a generar el contrato de Factoraje en el cual se estipulan todos los términos y condiciones de la operación, comisiones, etcétera. Al firmarse el contrato el prospecto se convierte formalmente en cliente.

Los pasos descritos anteriormente, así como su orden dentro del proceso se muestran en el diagrama de flujo de la imagen 2.2.

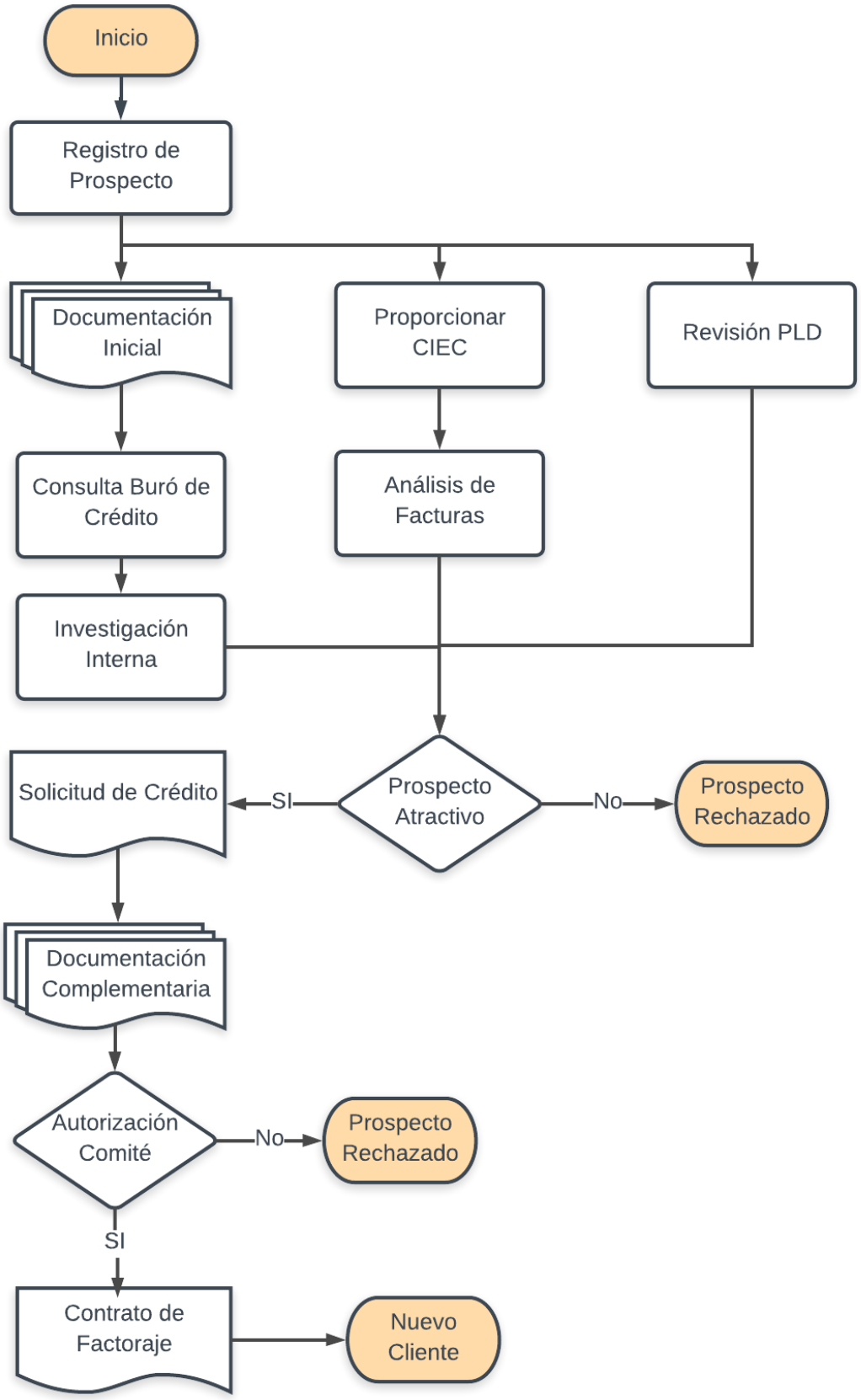


Imagen 2.2 Diagrama de flujo de proceso Onboarding

2.3 Necesidad de una plataforma

Al igual que muchas otras compañías la empresa tenía una administración de sus prospectos de forma manual debido a que las herramientas utilizadas no eran lo suficientemente robustas y personalizadas para llevar a cabo la gestión de tareas requeridas en el proceso de forma eficiente.

El mayor problema que se podía observar era el posible error humano al realizar ciertas tareas, por ejemplo, al descargar facturas del portal del SAT o en su defecto al hacer el análisis sobre éstas.

Otro problema que constantemente estaba presente era la generación de documentos ya que podía ocurrir un error en los datos al capturar información manualmente, adicionalmente no se contaba con herramientas digitales para comprobar la autenticidad de la firma.

Desde un punto de vista administrativo era muy complicado medir el progreso del prospecto entre las diferentes etapas del proceso, así como tomar los tiempos que tomaba cada paso y no se contaba con visibilidad clara de en qué fase se encontraba.

Con la firme idea de que toda empresa que busque ser competitiva en el mercado debe pasar por una transformación digital se toma la iniciativa de construir una plataforma digital que pueda subsanar todas estas deficiencias, que se adecue complemente a las necesidades de negocio, que sea funcional, escalable, robusta y que les permita llevar todo el seguimiento de los prospectos de forma eficiente.

2.4 Alcance del proyecto

El proyecto de la plataforma digital es una solución integral de software hecha a la medida, la cual abarca más de un sistema y requiere de algunas dependencias desarrolladas por terceros, por lo cual el documento se centra en explicar los componentes elaborados directamente por mí.

Mi participación dentro del proyecto abarca diferentes etapas del desarrollo de software como son el análisis de requerimientos funcionales y no funcionales, el diseño de una solución de software incluyendo la elección de tecnologías a utilizar, la implementación de la solución, algunos tipos de pruebas, liberación y posteriormente el soporte y mantenimiento.

A lo largo del documento se pretende detallar mi participación en cada una de las fases mencionadas y cómo se aplicaron los conocimientos adquiridos de la carrera de Ingeniería en Computación para llegar a la solución del problema.

Capítulo 3
Fase de análisis

3.1 Procesos de desarrollo de software

Un proceso de software es un conjunto coherente de políticas, estructuras organizativas, tecnologías, procedimientos y artefactos que se necesitan para concebir, desarrollar, implantar y mantener un producto de software.

Dentro de este proceso se contempla todo el ciclo de vida de un desarrollo de software, es decir, el periodo de tiempo que comienza cuando se toma la decisión de desarrollar un producto de software y que concluye cuando se entrega el software.

Existen diferentes modelos con los cuales se puede representar un proceso de software, cada uno ofrece un enfoque diferente tanto del proceso como de las actividades a realizar y su periodicidad, por lo que hay que elegir el modelo que más se ajuste a la naturaleza del problema que se deba resolver. Algunos ejemplos son el modelo en cascada, modelo en V, modelo basado en prototipos y modelo en espiral.

Para el desarrollo del software que se describe a lo largo de este documento se eligió un esquema mixto entre el modelo basado en prototipos y el modelo en cascada, a continuación, se explican ambos modelos.

Modelo basado en prototipos

Un prototipo es un modelo ejecutable de un sistema futuro que implementa solo una parte de la funcionalidad, pero permite a los clientes, usuarios y desarrolladores familiarizarse con la arquitectura y la funcionalidad. Los prototipos permiten desarrollar en fases tempranas del proyecto un común entendimiento entre todas las partes involucradas con lo cual hay menos ajustes en el software en fases avanzadas del desarrollo. El flujo general del modelo de prototipos se muestra en la imagen 3.1.

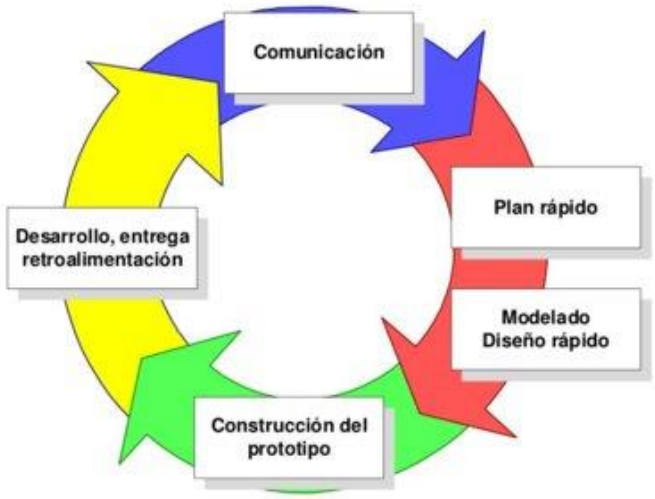


Imagen 3.1 Ciclo de trabajo en el modelo de prototipos

Existen dos tipos de prototipos: prototipo rápido y prototipo evolutivo. El prototipo rápido es muy básico, por lo cual considera mucho menos aspectos técnicos y usualmente no es completamente funcional, principalmente es usado para mostrar las interfaces del usuario en función de los requerimientos, por lo regular estos prototipos son desechados.

Por otro lado, el desarrollo basado en prototipos evolutivos se basa en la idea de desarrollar una implementación inicial, para exponerla a los comentarios del usuario y después refinarla a través de las diferentes versiones hasta que se desarrolla un sistema adecuado. Este desarrollo evolutivo se representa en la imagen 3.2.

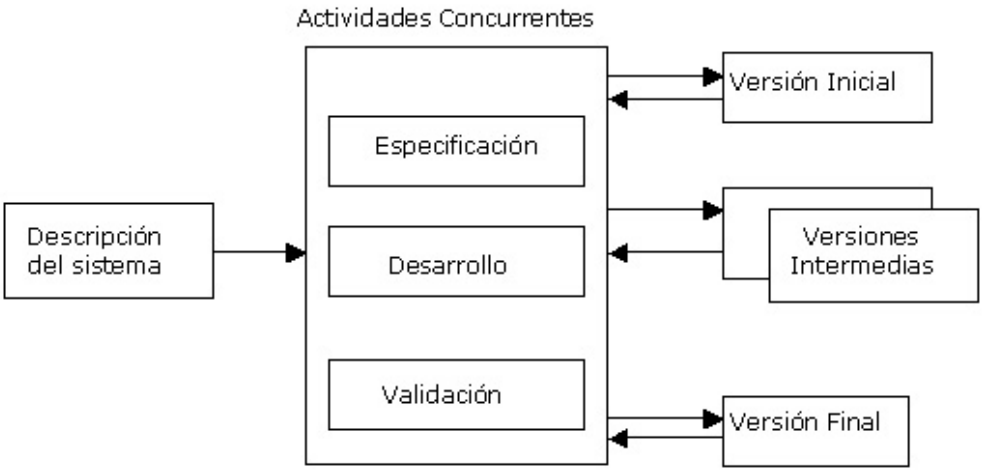


Imagen 3.2 Esquema de prototipos evolutivos

No obstante, este modelo implica algunas desventajas ya que los cambios continuos tienden a corromper la estructura del software haciendo que su arquitectura sea deficiente y que la tarea de realizar cambios sea más complicada a medida que va creciendo el software, adicionalmente es necesario hacer entregas regulares para medir el progreso del desarrollo y no es rentable producir documentos que reflejen cada versión del sistema pues estas cambian con mayor rapidez. Este tipo de fenómenos no suelen ocurrir en un modelo de cascada.

Modelo en cascada

El modelo en cascada es un proceso de desarrollo secuencial, en el que el desarrollo de software se concibe como un conjunto de etapas que se ejecutan una tras otra. Se le denomina así por las posiciones que ocupan las diferentes fases que componen el proyecto, colocadas una encima de otra y siguiendo un flujo de ejecución de arriba hacia abajo, como una cascada. Como se muestra en la imagen 3.3.

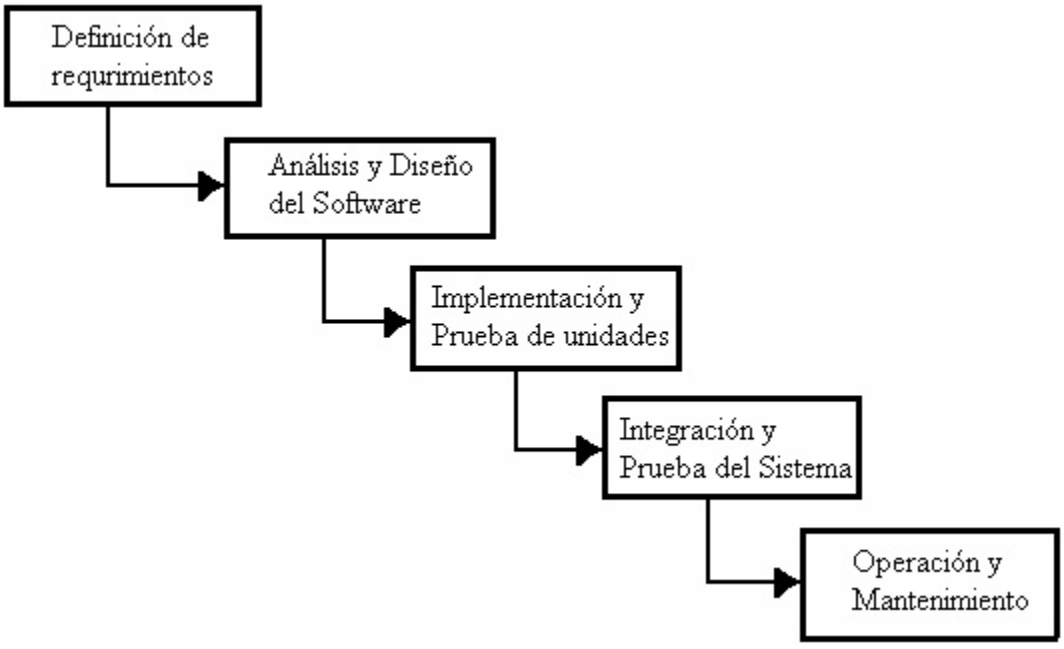


Imagen 3.3 Modelo en cascada

Análisis y definición de requerimientos. Los servicios, restricciones y metas del sistema se definen a partir de las consultas con los usuarios. Entonces, se definen en detalle y sirven como una especificación del sistema.

Diseño del sistema y del software. El proceso de diseño del sistema divide los requerimientos en sistemas hardware o software. Establece una arquitectura completa del sistema. El diseño del software identifica y describe las abstracciones fundamentales del sistema de software y sus relaciones.

Implementación y prueba de unidades. Durante esta etapa, el diseño del software se lleva a cabo como un conjunto o unidades de programas. La prueba de unidades implica verificar que cada una cumpla su especificación.

Integración y prueba del sistema. Los programas o las unidades individuales de programas se integran y prueban como un sistema completo para asegurar que se cumplan los requerimientos del software. Después de las pruebas, el sistema software se entrega al cliente.

Funcionamiento y mantenimiento. Por lo general (aunque no necesariamente), ésta es la fase más larga del ciclo de vida. El sistema se instala y se pone en funcionamiento práctico. El mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema y resaltar los servicios del sistema una vez que se descubren nuevos requerimientos.

Este modelo conserva un orden secuencial bien definido por lo cual es muy sencillo medir el progreso del desarrollo basándose en la fase en la que se encuentra. Este mismo orden permite generar un mejor diseño, así como una arquitectura más sólida y robusta, sin embargo, es mucho menos flexible a aceptar cambios de requerimientos del cliente, en especial cuando se encuentra en fases avanzadas. El éxito de un proyecto en cascada dependerá de la correcta ejecución de las fases previas empezando por una correcta definición de requerimientos.

Eligiendo un modelo

Para compensar las deficiencias de ambos modelos se implementó un esquema mixto de estos dos modelos, el cual contiene las mayores fortalezas de los enfoques de prototipos y de cascada.

El modelo de prototipos se eligió de referencia ya que se ajustaba mejor a la naturaleza del problema debido a que los usuarios (colaboradores de la empresa que realiza el proceso Onboarding) no tenían ninguna referencia de cómo debería ser el sistema en cuanto a los módulos o a las interfaces que tendría, es aquí donde los prototipos

presentados permiten establecer de forma clara la funcionalidad y empezar a familiarizar a los usuarios con el software.

Adicionalmente, al ser de distintas áreas (Comercial, Riesgos, Dirección, etc.), la comunicación se vuelve más complicada y existe el riesgo de entender de forma errónea los requerimientos del usuario por parte de los desarrolladores, pero los prototipos permiten perfeccionar los requerimientos en fases iniciales gracias a la retroalimentación del usuario.

Debido a que se tiene un común entendimiento de los requerimientos, así como del alcance del proyecto resulta muy bueno utilizar un modelo de cascada para crear una arquitectura robusta del software, pues el riesgo de que se presenten cambios en fases posteriores ya se ha reducido considerablemente y partiendo de los prototipos iniciales que han sido aceptados por los usuarios es mucho más sencillo avanzar entre las fases del desarrollo y pruebas. Si se analiza el desarrollo del software desde el punto de vista del modelo en cascada los prototipos participan principalmente en las fases de “Análisis y definición de requerimientos” y “Diseño del sistema y del software”.

La forma de presentar la información en el documento se apega a la estructura del modelo de cascada, haciendo énfasis en las partes del proceso en las cuales tuve mayor participación como son el análisis, diseño y la implementación, no obstante, mi participación se encuentra presente en todas las etapas del proceso, este proceso se describe en los capítulos siguientes.

3.2 Análisis de requerimientos

Un requerimiento de software es la propiedad que un software desarrollado o adaptado debe tener para resolver un problema concreto. Existen dos tipos de requerimientos, los funcionales y los no funcionales.

Los requerimientos funcionales definen el comportamiento del sistema que se va a desarrollar, incluyendo los procesos fundamentales que el software llevará a cabo. La mayoría de estos requerimientos provienen directamente del usuario (reglas de negocio).

Por otro lado, los requerimientos no funcionales tienen que ver con restricciones y exigencias de calidad del sistema, entre las que se encuentran requisitos de rendimiento, seguridad, tiempos de respuesta, mantenimiento, etc.

Dentro del software que desarrollé el proceso de obtención de requerimientos se dio principalmente por medio de juntas y reuniones en las cuales participaron los usuarios, los desarrolladores, el líder de proyecto y el director de la empresa. En estas reuniones los diferentes usuarios expusieron sus necesidades y expectativas sobre el software a desarrollar, este conjunto de información proveniente del usuario fue procesada para convertirse en un conjunto de requerimientos funcionales.

Como se mencionó anteriormente, es fundamental tener claridad de los requerimientos ya que éstos fijan los cimientos de cualquier desarrollo, por lo que los requerimientos fueron revisados y en algunos casos ajustados para poder cubrir las necesidades de los usuarios.

Dentro de esta parte del desarrollo me apegue más al proceso de desarrollo basado en prototipos para poder generar desde una fase temprana algunas representaciones visuales de los requerimientos que los usuarios pudieran entender y así dar su aprobación de los requerimientos o en su defecto indicar las respectivas correcciones.

Los prototipos iniciales permitieron el común entendimiento entre las personas involucradas y dejar claro cuál sería el alcance del software a desarrollar, como resultado de las reuniones y retroalimentación de los usuarios se obtuvo la validación de los requerimientos funcionales por parte de ellos y del área de desarrollo. Este proceso se muestra en la imagen 3.4.

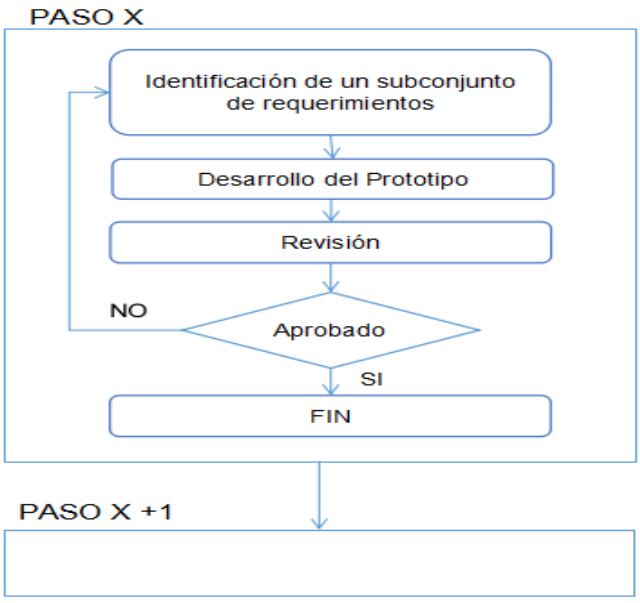


Imagen 3.4 Análisis de requerimientos

3.2.1 Requerimientos funcionales

A continuación se presenta una lista de los requerimientos funcionales más relevantes del desarrollo.

- El sistema permitirá capturar la información necesaria de la persona moral (prospecto), su representante legal y un contacto.

- El sistema permitirá gestionar la documentación relacionada con el prospecto. Estados financieros, acta constitutiva, RFC, Identificación Oficial, etc.

- La documentación podrá ser agregada por el prospecto o por el personal interno de la empresa.

- La documentación será aprobada únicamente por el área de riesgos.

- El sistema permitirá firmar de forma electrónica por el cliente y por la empresa toda la documentación que sea necesaria.

- Se tendrá comunicación con el SAT para poder descargar facturas de los prospectos mediante la clave CIEC.

- El sistema automatizará el análisis de facturas de acuerdo con el modelo de negocio de la empresa

- Se tendrá comunicación con Buró de Crédito para poder consultar a los prospectos en dicha institución.

- El sistema automatizará la investigación del prospecto en temas referentes a protección de lavado de dinero (PLD).

- Todos los documentos que sean elaborados por parte de la empresa durante el Onboarding deberán ser generados de forma automática por el sistema, por ejemplo, la solicitud de crédito, el contrato de factoraje financiero y la carta de aceptación de buro de crédito.

- El sistema permitirá la captura de datos adicionales de la personal, los cuales serán plasmados en el contrato de factoraje financiero.

- Se enviarán notificaciones a los usuarios sobre información relevante del prospecto y de su progreso en el proceso Onboarding.

- Se contará con diferentes roles de usuario, cada uno tendrá acceso limitado a funcionalidades específicas del sistema. Los roles en el sistema se crearán de acuerdo con los roles de la empresa: usuario prospecto, comercial, riesgos, finanzas y dirección.

3.2.2 Requerimientos no funcionales

Después de tener una correcta definición de los requerimientos funcionales es importante tener una serie de requerimientos no funcionales los cuales guiarán muchos aspectos del desarrollo del software posterior al análisis y ayudarán a la toma de decisiones referentes a las tecnologías empleadas. A continuación se presentan los requerimientos no funcionales más relevantes.

- El mecanismo de autenticación está basado en usuario y contraseña y se deben establecer directamente por los usuarios, deben de cifrarse antes de ser almacenados y deben de estar en un lugar diferente al resto de los datos.

- Se deberá permitir la gestión de usuarios, roles y permisos, y ésta debe ser sencilla, escalable y fácil de mantener.

- Se permitirán gestionar diferentes tipos de archivo para los documentos, imágenes y archivos comprimidos dando soporte a los formatos comunes como son pdf, jpg, jpeg, png, rar, zip.

- La documentación tendrá diferentes estatus: Documento no entregado, en proceso de validación, aprobado y rechazado.

- El sistema debe ser multiplataforma y poder ejecutarse en laptops, pc, MacBook y dispositivos celulares.

- Se debe contar con un mecanismo de respaldo continuo de la información, tanto de los documentos como de los datos de los prospectos.

- El sistema llevará registro de la actividad de los usuarios dentro de la plataforma, por lo cual se guardará la fecha, hora, así como el id del usuario de las acciones relevantes.

- El sistema conservará un histórico de la actividad de los usuarios para realizar mediciones del tiempo que se tardan los usuarios entre cada fase del proceso.

- La comunicación con proveedores externos y entidades gubernamentales para enviar y recibir información debe ser mediante e Servicios Web y/o API.

3.2.3 Representación de requerimientos

Como se explicó anteriormente estos requerimientos se obtuvieron gracias a la presentación de prototipos ante los usuarios para recibir la respectiva retroalimentación de su parte. Algunos de los prototipos elaborados para los requerimientos se muestran en las imágenes 3.5 a 3.9.

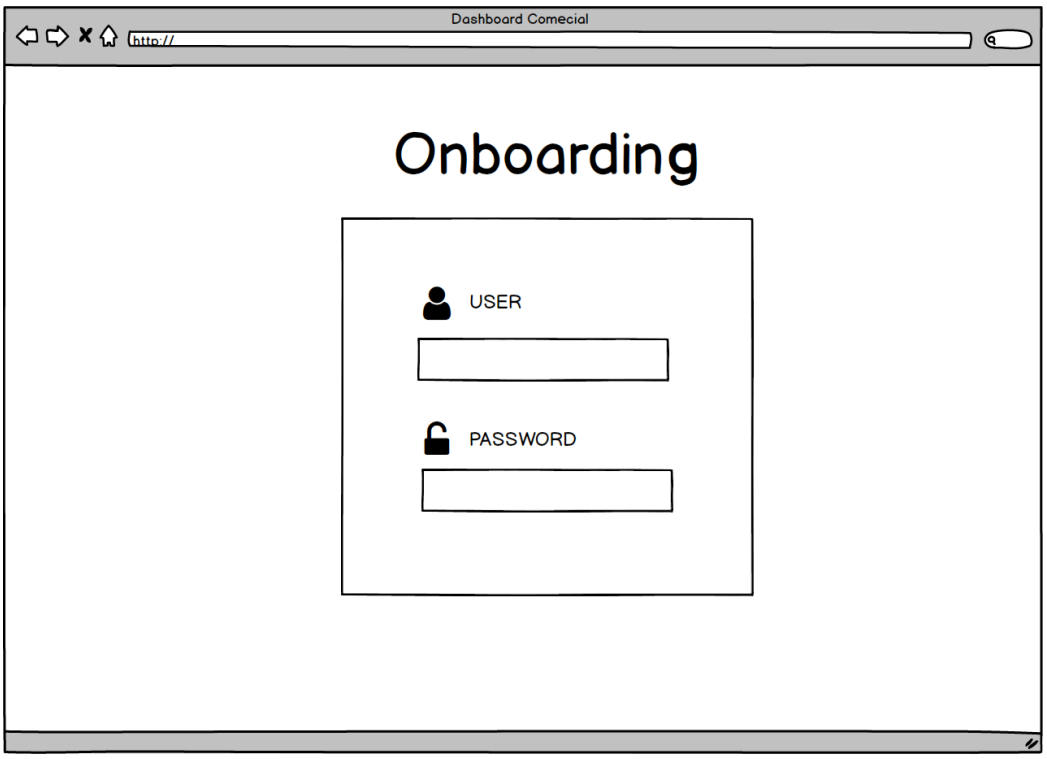


Imagen 3.5 Prototipo para inicio de sesión

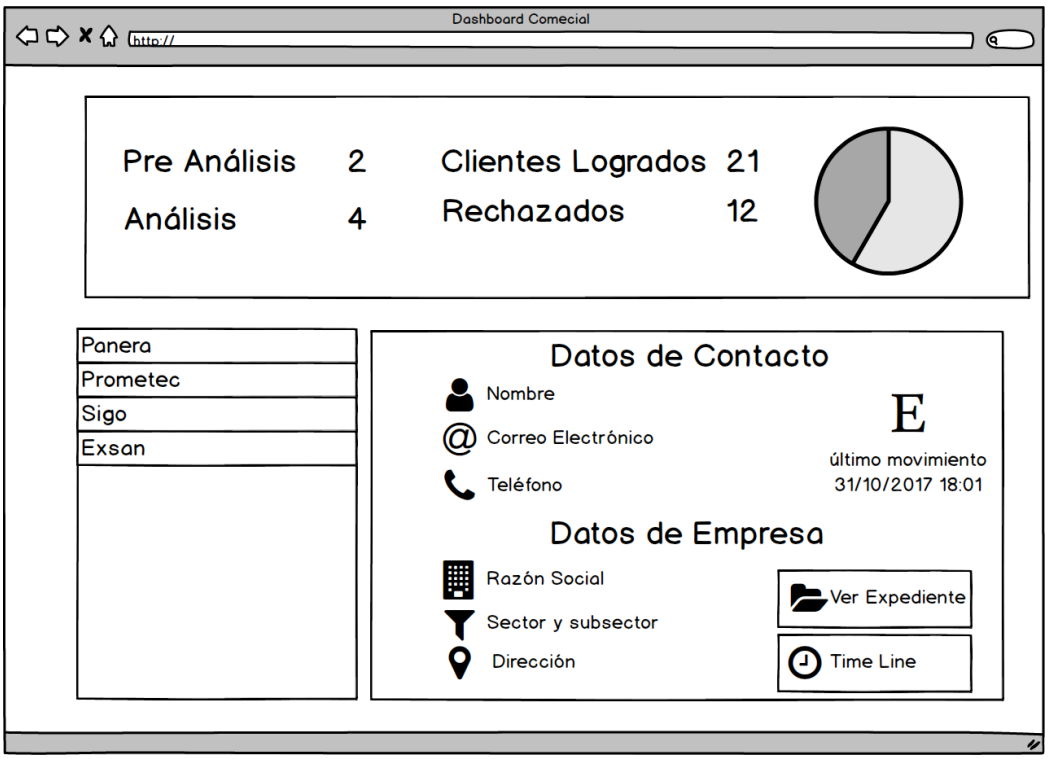


Imagen 3.6 Prototipo para pantalla principal de colaboradores

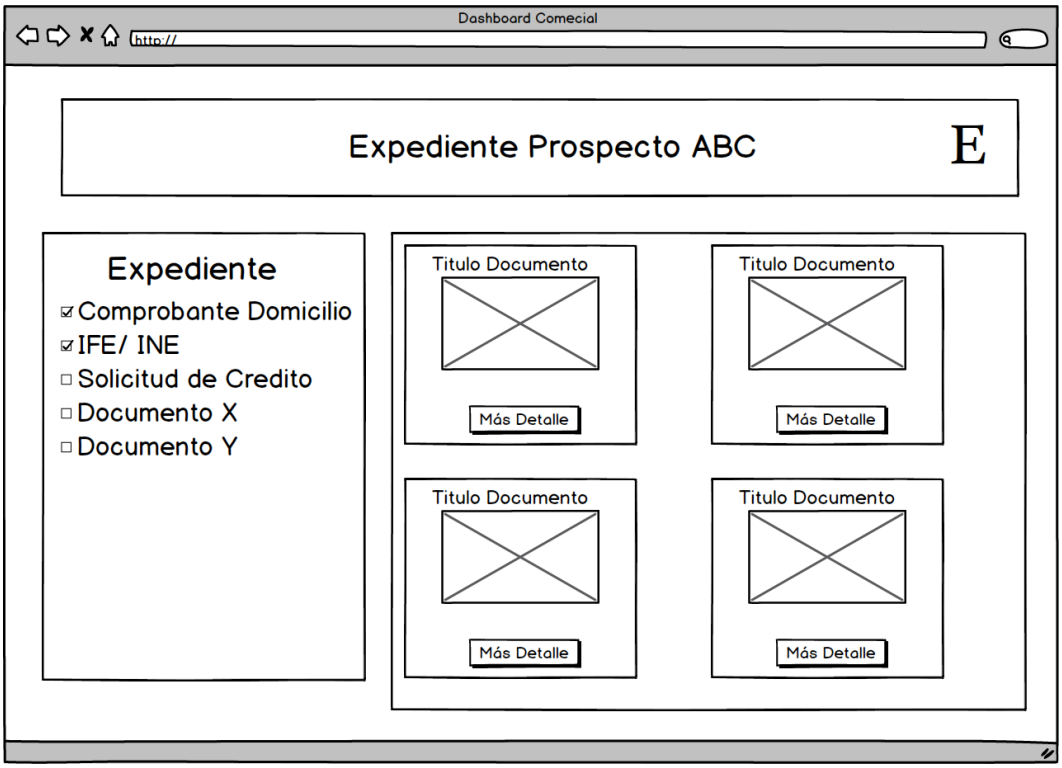


Imagen 3.7 Prototipo para visor de documentos

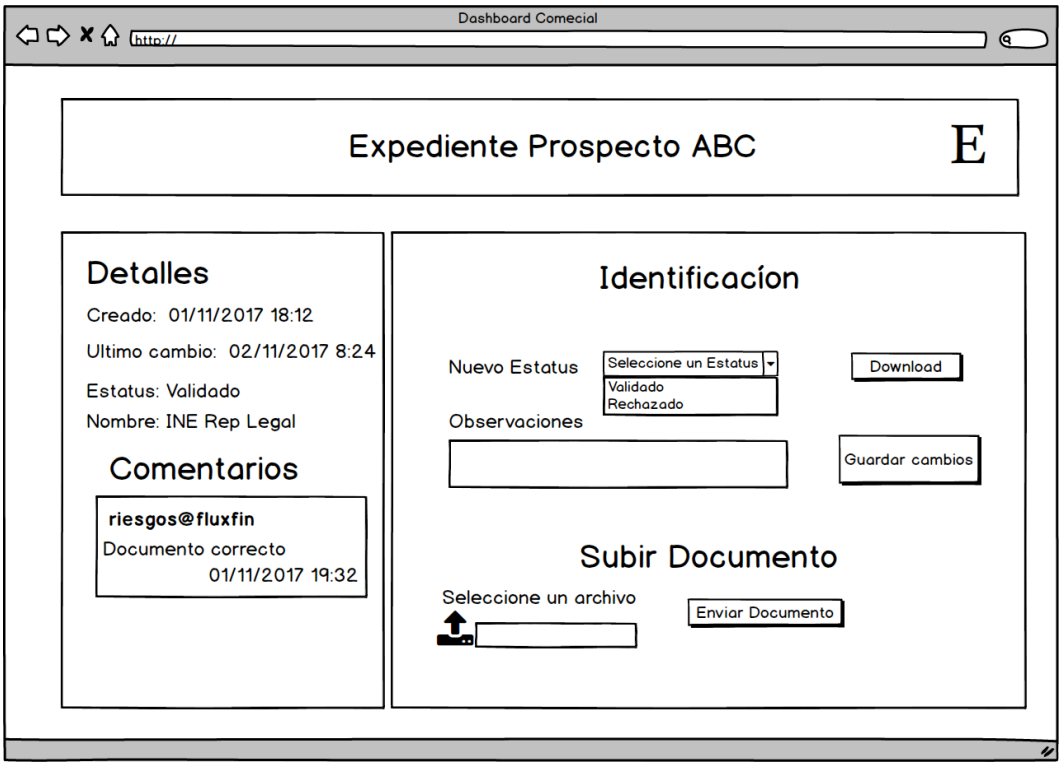


Imagen 3.8 Prototipo para administración de documentos

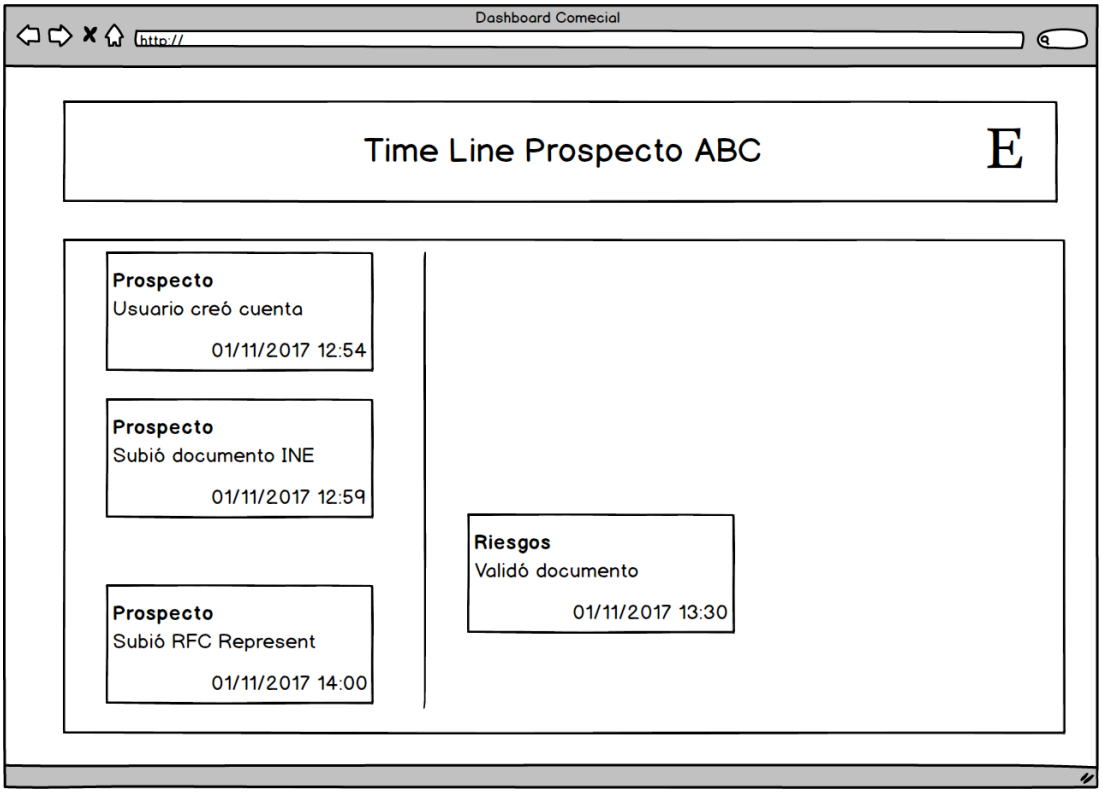


Imagen 3.9 Prototipo para la línea de tiempo del prospecto

Adicional a la elaboración de los prototipos es necesario apoyarse en diferentes diagramas que ayuden a establecer un común entendimiento de los requerimientos del sistema y para este desarrollo se utilizaron diagramas de secuencia.

Este tipo de diagrama se eligió porque se enfoca en el ciclo de vida de un proceso en específico dentro del software, muestra cuáles serán los objetos o componentes que interactúan en el proceso, así como el orden en el cual deben intercambiar mensajes. Algunos procesos pueden ser el inicio de sesión y el registro de usuarios, los cuales se muestran en la imagen 3.10.

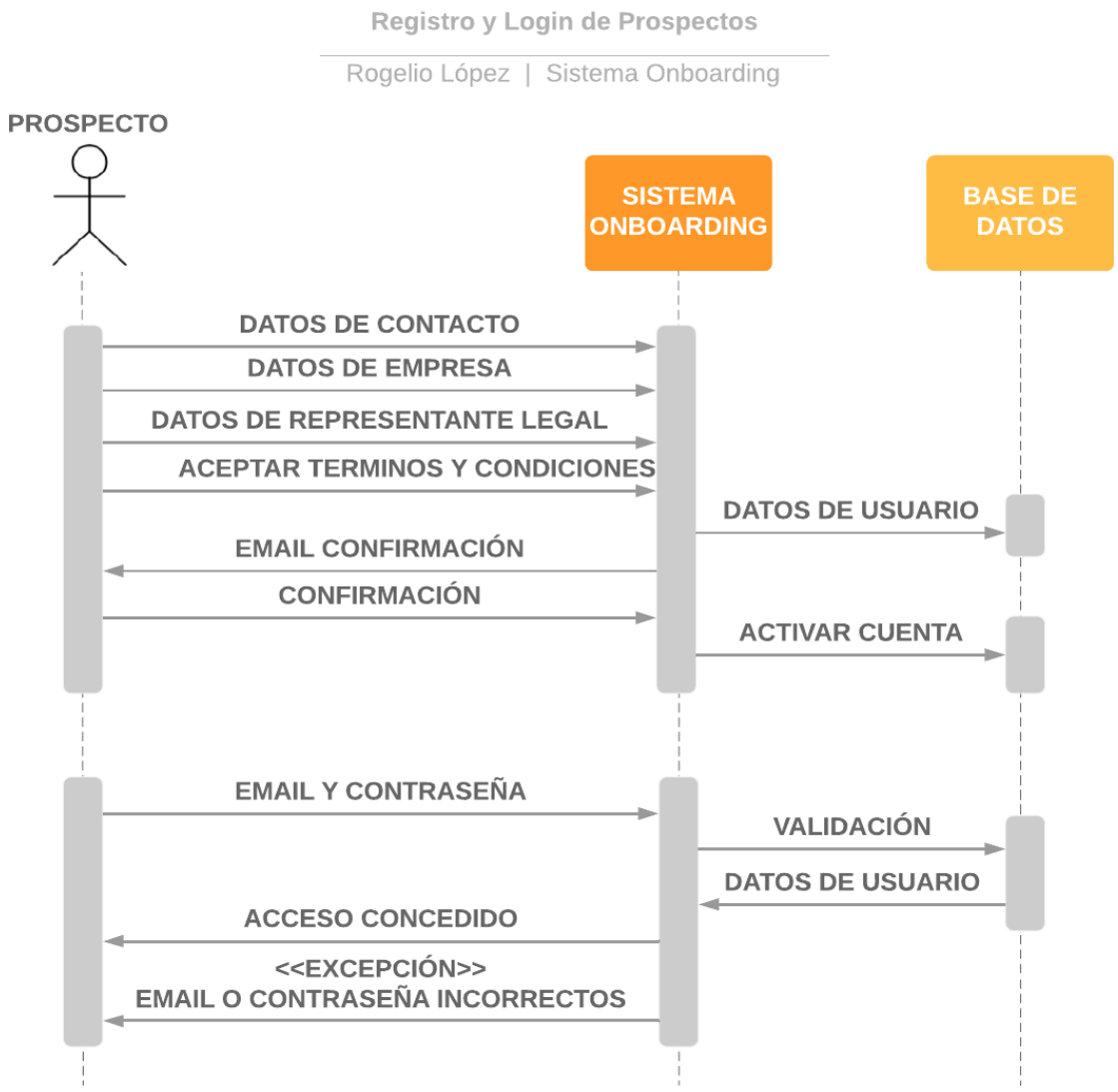


Imagen 3.10 Diagrama de secuencia para registro e inicio de sesión

Como se puede observar en el diagrama anterior se ilustra el proceso de registro de un prospecto en el sistema, podemos ver que existen 3 elementos: el usuario prospecto, el sistema Onboarding y la base de datos, adicionalmente se indica cómo interactúa cada uno con los otros, es decir se menciona cuáles son los datos que debe capturar el usuario en el sistema y cuando este debe enviarlos a la base de datos para ser almacenados, de igual forma se indican algunas acciones propias del sistema como el envío de correos y validación de datos.

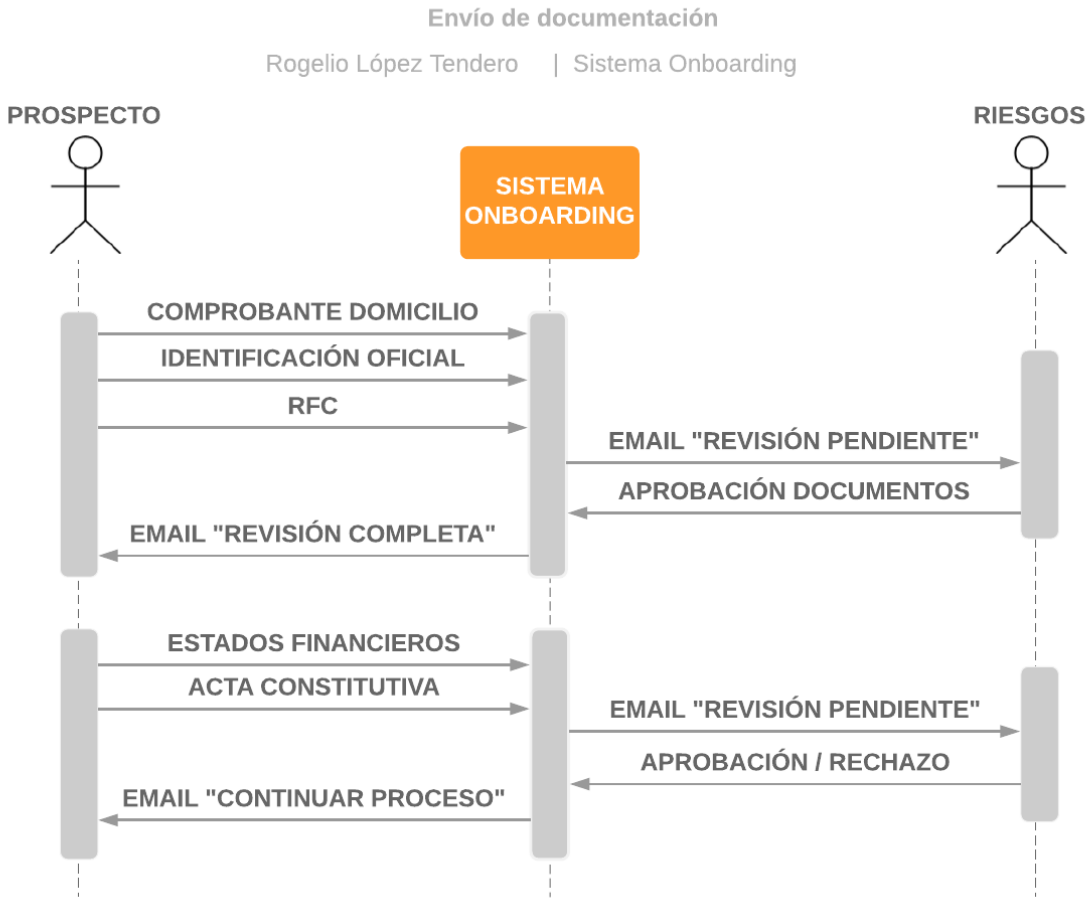


Imagen 3.11 Diagrama de secuencia para enviar la documentación en el sistema

En diagrama 3.11 se representan otro conjunto de requerimientos, por ejemplo, se detalla cuál será documentación solicitada al prospecto, el orden correcto en el cual deben irse cargando al sistema para posteriormente ser validados por un perfil de Riesgos, de igual forma se puede ver como el sistema deberá realizar notificaciones a los usuarios en el momento necesario.

Este tipo de diagramas no indica las tecnologías que se van a utilizar, ni tampoco que componente o interfaz dentro del sistema realizará cada acción en concreto, no obstante resulta de mucha utilidad en una fase de análisis ya que establece el orden correcto del proceso, adicionalmente nos ayuda a ir definiendo qué componentes deberán existir en el sistema así como las acciones que cada uno debe realizar y al estar revisado y aprobado por el usuario final hace que la labor de programación en fases posteriores sea más sencilla y menos susceptible a cambios.

3.3 Tecnologías utilizadas

Una vez que se han establecido de forma clara los requerimientos de usuario y se conocen las funcionalidades que deberá ofrecer el sistema, el siguiente paso es elegir las tecnologías que se utilizarán para llevar a cabo el desarrollo del software, estas tecnologías se pueden agrupar fácilmente en tres categorías: Base de Datos, Lenguajes de programación y Servicios en la nube.

Hoy en día existe una gran variedad de productos para cada uno de estos tres grupos y mientras algunas desaparecen, otras evolucionan y otras aparecen para reemplazar a las antiguas, este constante cambio hace que los arquitectos y desarrolladores de software tengan más opciones, sin embargo, también hace que la tarea de elegir la tecnología sea más complicada. En este punto resulta muy importante resaltar que no existe una tecnología que sea mejor que las demás, no existe un lenguaje de programación o una base de datos perfectos para todo, más bien cada producto ofrece diferentes características que pueden ser consideradas como ventajas o desventajas dependiendo la naturaleza del problema que se busque resolver.

Aunque el objetivo del presente documento no es realizar un análisis detallado de las características de cada producto, ni realizar comparación entre las diferentes alternativas del mercado esta sección presenta el conjunto de los lenguajes y tecnologías empleadas para programar el sistema Onboarding dando una descripción de cada tecnología indicando cómo se integran en la solución y mostrando algunas ventajas por las cuales fue seleccionado.

3.3.1 Base de datos

Las bases de datos son herramientas fundamentales dentro de todo desarrollo de software pues estas almacenan toda la información de la aplicación, las hay de tipo relacional, donde la información se muestra a modo de tabla y en ella se integran un conjunto de registros y existen las bases de datos no relacionales que se emplean para grandes cantidades de información y aplicaciones de inteligencia empresarial.

Debido a la naturaleza del sistema Onboarding se eligió una base de datos relacional pues el volumen de datos, así como el número de transacciones no es alto por lo cual el rendimiento y la velocidad de respuesta entre una base relacional y una no relacional no representa gran diferencia, no obstante, el problema requiere que la información esté totalmente estructurada priorizando más en la consistencia e integridad de los datos.

Aunque es posible tener esta solución en una base no relacional como MongoDB es mucho más sencillo si se utiliza una base relacional al implementar diferentes mecanismos como son las transacciones, llaves primarias y foráneas para mantener la integridad referencial, especificando el tipo de dato de cada campo, así como su dominio, entre otros. Para el desarrollo del sistema se utilizó SQL Server 2014 y a continuación se describen algunas de sus características más relevantes

Microsoft® SQL Server™ es un sistema de administración y análisis de bases de datos relacionales de Microsoft para soluciones de comercio electrónico, línea de negocio y almacenamiento de datos, posee una gran variedad de funciones críticas, proporcionando un rendimiento, una disponibilidad y una facilidad de uso innovadores para las aplicaciones más importantes. Microsoft SQL Server ofrece nuevas capacidades en memoria en la base de datos principal para el procesamiento de transacciones en línea (OLTP) y el almacenamiento de datos.

SQL Server 2014 también proporciona nuevas soluciones de copia de seguridad y de recuperación ante desastres, así como de arquitectura híbrida con Windows Azure, lo que permite a los clientes utilizar sus actuales conocimientos con características locales que aprovechan los centros de datos globales de Microsoft.

3.3.2 Lenguajes de programación

De acuerdo con los requerimientos de negocio es necesario que el sistema sea multiplataforma, es decir, que pueda ejecutarse en diversos entornos o sistemas operativos, por ejemplo, una computadora, una Tablet o un teléfono móvil, por lo que los lenguajes utilizados deben ser de programación web. Cuando se habla de un desarrollo web es normal dividir la programación en FrontEnd y Backend.

FrontEnd es la parte de un programa o dispositivo a la que un usuario puede acceder directamente. Son todas las tecnologías de diseño y desarrollo web que corren en el navegador y que se encargan de la interactividad con los usuarios.

Backend es la capa de acceso a datos de un software o cualquier dispositivo, que no es directamente accesible por los usuarios, además contiene la lógica de la aplicación que maneja dichos datos. El Backend accede a recursos del servidor para atender las solicitudes hechas por el usuario a través de su navegador.

Para la programación del FrontEnd, los lenguajes por excelencia son HTML, CSS y JavaScript, de los cuales se desprenden muchos Frameworks y bibliotecas para expandir sus capacidades. En este desarrollo además de ser usados en su forma pura se han utilizado los Frameworks jQuery y Bootstrap.

jQuery es una biblioteca de JavaScript rápida, pequeña y con muchas funciones. Hace que cosas como la manipulación de documentos HTML, el manejo de eventos, la animación y el envío de peticiones Ajax sean mucho más simples con una API fácil de usar que funciona en una multitud de navegadores. Con una combinación de versatilidad y extensibilidad.

A pesar de haber diferentes alternativas en el mercado se ha elegido jQuery para el proyecto debido entre otras razones a su madurez ya que la librería cuenta con soporte constante y rápido, publicándose actualizaciones de manera regular. Además, la forma de escribir código es muy sencilla y es posible la integración de bibliotecas para diferentes objetivos del proyecto como subir documentos, trabajar con calendarios y realizar la validación de formularios de una forma transparente con lo cual el tiempo de desarrollo se reduce.

Por su parte Bootstrap es un conjunto de herramientas de código abierto para desarrollar con HTML, CSS y JS, el cual se ha convertido en uno de los Frameworks FrontEnd más populares ya que simplifica el proceso de maquetación de un sitio web.

La principal razón para incluir Bootstrap en el proyecto es que provee de todas las reglas CSS para que un sitio web se adapte de manera dinámica a la mayoría de las pantallas gracias a su esencia Mobile First. En esta forma de maquetar el sitio el diseño se centra en los dispositivos con pantallas más pequeñas y generalmente con menor ancho de banda disponible para la navegación, posteriormente el acomodo de contenido se escala a dispositivos con resoluciones más elevadas, de esa forma se garantiza que para cada resolución el sitio conservará una organización adecuada.

Por el lado de la programación Backend se tienen diferentes lenguajes de programación con los cuales es posible construir las aplicaciones, algunos ejemplos son Python, PHP, Ruby, C# y Java, para cada uno existen diferentes Frameworks que facilitan la programación, para el desarrollo del sistema Onboarding se utilizó el lenguaje C# y el Framework en específico fue ASP.net MVC 5.

El framework ASP.NET MVC está construido sobre el entorno ASP.NET para aplicaciones web desarrollado y comercializado por Microsoft, con el cual es posible desarrollar sitios web dinámicos, aplicaciones web y servicios web. Las siglas MVC se deben a que implementa el patrón de diseño Modelo Vista Controlador presente en casi todos los sitios web actuales con lo cual la estructura de la aplicación se mantiene muy bien organizada. Más adelante se hablará del patrón de diseño con mayor profundidad.

El framework MVC permite realizar un desarrollo controlado por pruebas (TDD) con lo cual es posible realizar pruebas unitarias de cada componente por separado de forma automatizada y de esa forma asegurar la funcionalidad y calidad del software.

El framework se benefician de la solidez y robustez del lenguaje de programación C# y esto es un factor importante para elegirlo pues como se ha mencionado el equipo Prometheus ha realizado diferentes proyectos previos al Onboarding usando tecnologías como Windows Forms las cuales están basadas en el mismo lenguaje C#, con lo cual es posible tener un paso más ágil entre aplicaciones de escritorio a aplicaciones web.

El uso de un lenguaje como C# permite construir componentes reutilizables en diferentes proyectos incluso si son de diferentes tipos como aplicaciones de escritorio, aplicaciones web o incluso aplicaciones móviles, cosa que no es posible con otros lenguajes como PHP que está enfocado únicamente a desarrollo web.

3.3.3 Servicios en la nube

Una vez que se ha seleccionado tanto la base de datos como los lenguajes de programación con los cuales se va a construir la aplicación es necesario definir donde quedarán alojadas estas dos. Entre las opciones disponibles para montar el sistema se encuentran los servidores locales de la empresa, los proveedores de Web Hosting y los servicios en la nube o Cloud Computing.

Sin embargo, no todas las opciones resultan convenientes para el proyecto a desarrollar, por ejemplo, al almacenar las aplicaciones en servidores locales de la compañía siempre existirá el riesgo de sufrir problemas o fallos tanto en los equipos como en las instalaciones, se tienen altos costos tanto en las licencias como en el soporte, mantenimiento y consumo de energía, el soporte es limitado y el equipo se deteriora con el paso del tiempo hasta quedar obsoleto.

Por otro lado, al tener las aplicaciones en un hosting se tiene el problema de que son muy rígidos en cuanto a su capacidad y procesamiento pues suelen estar en planes definidos en los cuales no hay posibilidad de personalizar el consumo de acuerdo con la compañía o aplicación haciendo que se eleve mucho el costo o que los recursos sean insuficientes para atender la demanda de la aplicación. Adicionalmente el soporte es limitado y no todos los proveedores soportan las bases de datos y/o lenguajes de programación deseados.

Esto nos deja la tercera alternativa, los servicios en la nube o cómputo en la nube. De acuerdo con Red Hat Cloud Computing se define de la siguiente forma

“Cloud computing es un conjunto de principios y enfoques que permite proporcionar infraestructura informática, servicios, plataformas y aplicaciones (que provienen de la nube) a los usuarios, a pedido y en una red. Las nubes son grupos de recursos virtuales (como el potencial de procesamiento en bruto, el almacenamiento o las aplicaciones basadas en la nube) orquestados por software de gestión.” (El concepto de cloud computing, (s.f.), Redhat.com, https://www.redhat.com/es/topics/cloud)

Es decir que gracias al cómputo en la nube es posible montar una infraestructura completa desde cero para poder alojar las aplicaciones, poniendo tantos recursos como sea necesario y pagando únicamente lo que se ha usado haciendo que este proceso sea mucho más sencillo tanto en la forma de implementar como de mantener pues desde un portal de administración es posible gestionar toda la actividad sin tener que estar físicamente junto al servidor.

Existen diferentes proveedores de servicios, pero los más grandes son Microsoft Azure, Amazon Web Services y Google Cloud. Para el proyecto de Onboarding se ha seleccionado Microsoft Azure.

Microsoft Azure es conjunto de servicios en la nube en constante expansión, la cual permite al usuario acceder a un catálogo de servicios estandarizados y responder a las necesidades de negocio, de forma flexible y adaptativa, en caso de demandas no previsibles o de picos de trabajo, pagando únicamente por el consumo efectuado.

Azure fue seleccionado ya que integra fácilmente las bases de datos de SQL Server, así como las aplicaciones de ASP .NET MVC y adicional a las bondades antes mencionadas de una nube, permite la implementación de diferentes ambientes como son desarrollo, pruebas y producción haciendo que los procesos de liberación sean sencillos y menos susceptibles a fallos. El tema de control y mantenimiento se abordará más adelante con mayor detalle. En la imagen 3.12 se ilustran las tecnologías más relevantes del proyecto.

Imagen 3.12 Stack de tecnologías

Capítulo 4
Fase de diseño

4.1 Patrón de diseño MVC

Antes de comenzar con la programación del sistema es fundamental tener un buen diseño de la solución que se va a construir, ya que esto encaminará los esfuerzos de los desarrolladores en una determinada dirección y servirá de guía en la forma de escribir el código de la aplicación. Afortunadamente no tenemos que crear un diseño desde cero pues existen los diferentes patrones de diseño.

Se puede entender un patrón de diseño como una solución que ya ha sido probada y documentada para resolver problemas de desarrollo de software con características similares a lo que nos enfrentamos, adicionalmente ya han demostrado que son capaces de obtener una solución y que los beneficios obtenidos compensan los costos de su implementación, en otras palabras, son el esqueleto de la solución.

Algunos de los patrones de diseño más conocidos son: Singleton, Builder y Modelo-Vista-Controlador o MVC por sus siglas, en el desarrollo del proyecto presentando se utilizó el patrón MVC, el cual se explica a continuación.

El patrón de diseño MVC separa una aplicación en tres grandes grupos de componentes o capas: modelos, vistas y controladores, como se puede ver en la imagen 4.1.

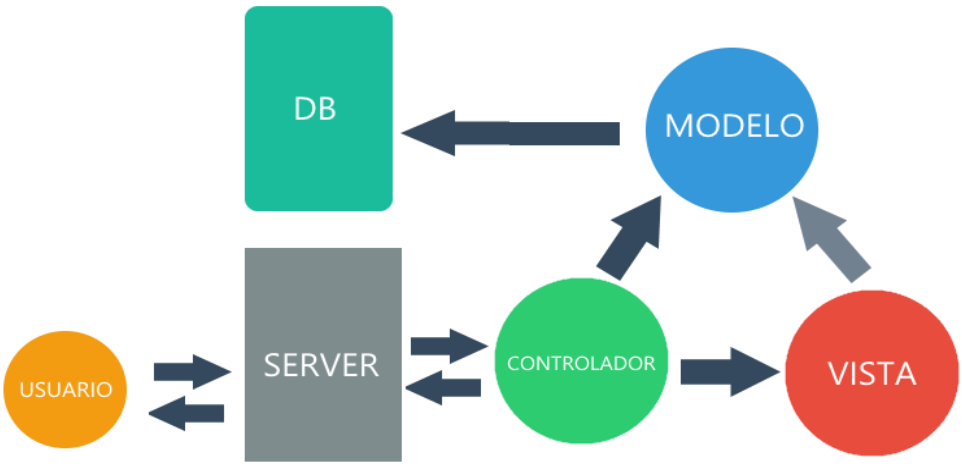


Imagen 4.1 Patrón de diseño MVC

El **modelo** representa la parte de la aplicación que trabaja con los datos, es decir, que es responsable de la recuperación de la información desde la Base de Datos, así como de su procesamiento, validación, asociación y cualquier otra tarea relativa a la manipulación de dichos datos.

El modelo contiene las operaciones CRUD (create, read, update, delete) para los datos las aplicaciones y encapsula la lógica de negocio junto con cualquier lógica para preservar el estado de la aplicación.

La **vista** es responsable de dar una representación de los datos obtenidos por el modelo previamente, es decir, contiene todo el código de la aplicación que describe la respuesta para el usuario.

Usualmente la capa de la Vista ofrece una representación en formato HTML de los datos, pero no está limitado a este formato en específico, sino que puede ser utilizada para ofrecer una amplia variedad de formatos en función de las necesidades de la aplicación por ejemplo texto, videos, música, documentos, JSON o XML.

El **controlador** es el encargado de gestionar las peticiones de los usuarios y ofrecer una respuesta al usuario, sin embargo, la responsabilidad del controlador no es manipular los datos, más bien se encarga de conectar al modelo con la vista.

Al recibir una petición del usuario el controlador comprueba su validez de acuerdo con las normas de autenticación o autorización de la aplicación, delega la búsqueda de datos al modelo y selecciona el tipo de respuesta más adecuado y delega este proceso de presentación a la capa de la Vista.

La interacción entre las 3 capas se puede expresar se puede ilustrar en la imagen 4.2.

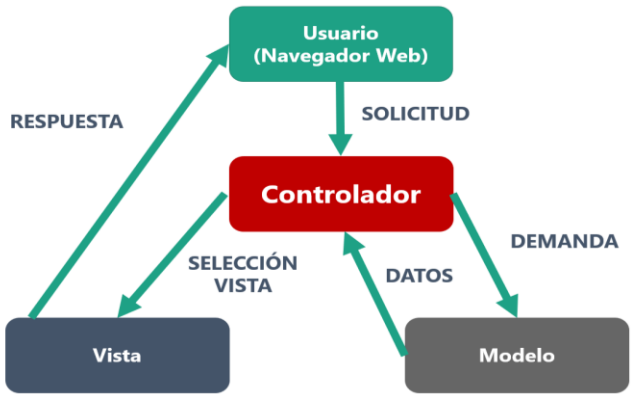


Imagen 4.2 Interacción Modelo-Vista-Controlador

Como se puede apreciar esta separación de componentes resulta muy efectiva ya que permite dividir las tareas de desarrollo según la capa a la que pertenece con lo cual se obtiene un paquete modular más fácil de mantener y de mejorar, incluso los errores pueden resolverse con mayor rapidez pues según su naturaleza estaría contenido en una cierta capa permitiendo a los desarrolladores hacer cambios en esa parte de la aplicación sin afectar a lo demás, a diferencia de una aplicación en la que estas tres funcionalidades se encuentran mezcladas.

4.2 Diseño de base de datos

La base de datos juega un papel fundamental en el desarrollo ya que contiene toda la información que genere la aplicación del Onboarding. Esta será la encargada de mantener los datos de una forma estructurada y ordenada por lo cual será responsable de definir la estructura de los modelos de la aplicación.

Como parte del diseño también es necesario llevar un proceso de normalización de bases de datos el cual consiste en aplicar una serie de reglas a las relaciones obtenidas tras el modelo entidad-relación con el objetivo de evitar la redundancia de los datos, disminuir problemas de actualización en las tablas y proteger la integridad de los datos.

Una base de datos bien diseñada facilita posteriormente la labor de programación pues los datos ya se encuentran perfectamente bien definidos tanto en estructura como en tipo de dato, además ya está establecida la relación con otras estructuras de datos y ya se han limitados los posibles valores que puedan tener, con lo cual su acceso es mucho más sencillo, de igual forma las consultas requeridas por la aplicación deben ser más sencillas de construir pues se partió de los requerimientos de la aplicación para formular el diseño.

Por el contrario, el tener un mal diseño puede implicar que las consultas sean mucho más complicadas, que se tengan datos duplicados o que no se tenga el control adecuado sobre ellos, un mal diseño se debe principalmente a que los requerimientos usados como guía están mal planteados y no reflejan la naturaleza de la aplicación

A continuación se muestra el diseño de la Base de Datos para la Aplicación. Es importante resaltar que es meramente ilustrativa, así que múltiples entidades y/o campos han sido omitidos del diagrama o renombrados para poder ser mostrados.

Como primer paso es necesario identificar aquellas tablas que servirán como catálogos para las demás entidades. En este caso se pueden distinguir cuatro de ellas.

Sector Económico: Define la actividad económica dentro del país, este catálogo es proporcionado por la CNBV de forma pública, por ejemplo *”Construcción”, “Servicios educativos”,* etc.

Subsector: Es una división de la tabla anterior ya que un sector puede estar dividido en subsectores más pequeños, por ejemplo, el sector *“Comercio al por menor”* puede estar dividido en *“Comercio al por menor en tiendas de autoservicio y departamentales”, “Comercio al por menor de artículos para el cuidado de la salud”*, etc.

Asentamiento: Catálogo proporcionado por el Servicio Postal Mexicano (SEPOMEX) con datos de todas las colonias, municipios, códigos postales y demás datos referentes a la localización de una vivienda dentro del país.

Documentación: Es el catálogo que define todos los pasos que deben ser cubiertos dentro del Onboarding de la empresa, por ejemplo: Consulta Buro de Crédito, Firma de Contrato, etc.

Estos catálogos se presentan en la imagen 4.3.

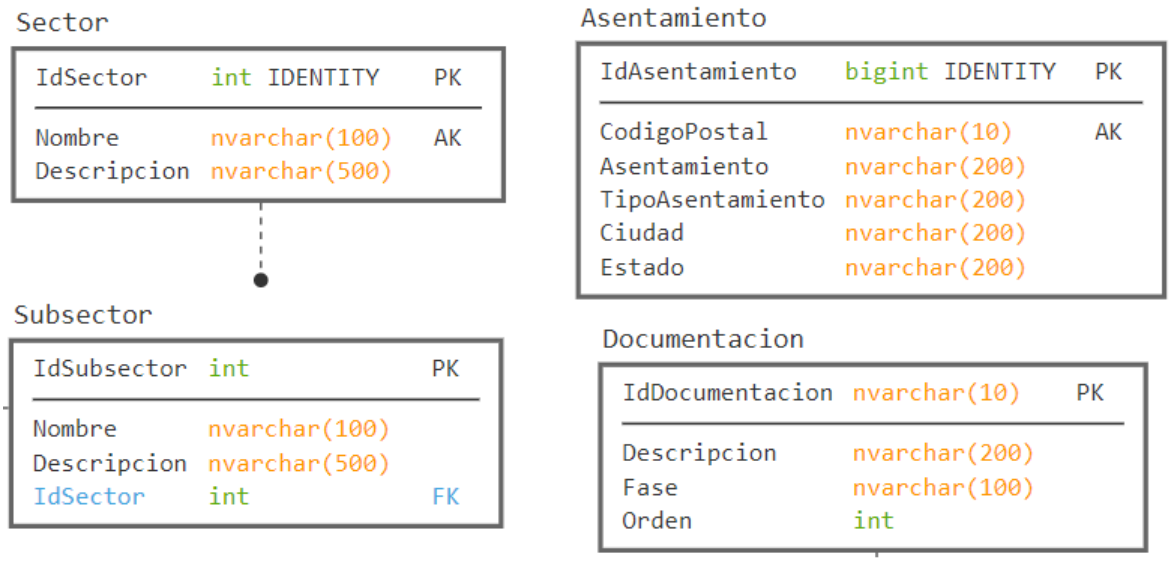


Imagen 4.3 Tablas de catálogos

Después es necesario definir algunas tablas propias de la aplicación.

Dirección: De acuerdo con los requerimientos es necesario capturar datos del domicilio de la empresa, así como múltiples datos referentes al domicilio como la delegación, estado o colonia. Algunos de estos datos son obtenidos de la tabla Asentamiento para evitar demasiados datos duplicados. Representada en imagen 4.4.

Expediente: Representa a cada uno de los expedientes creados para almacenar los archivos de un prospecto. Contiene la referencia de Google Drive para poder ubicarlo. Representada en imagen 4.5.

Archivo: Está asociado siempre a un expediente y contiene los metadatos necesarios para poder recuperar el archivo desde los servicios de Google Drive. Representada en imagen 4.5.

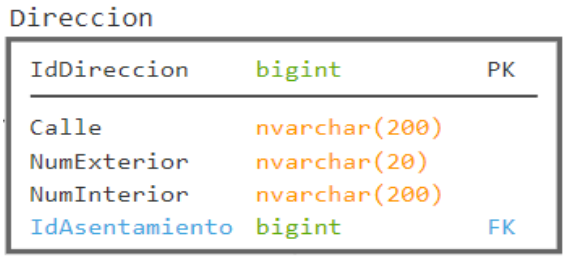


Imagen 4.4 Tabla Dirección

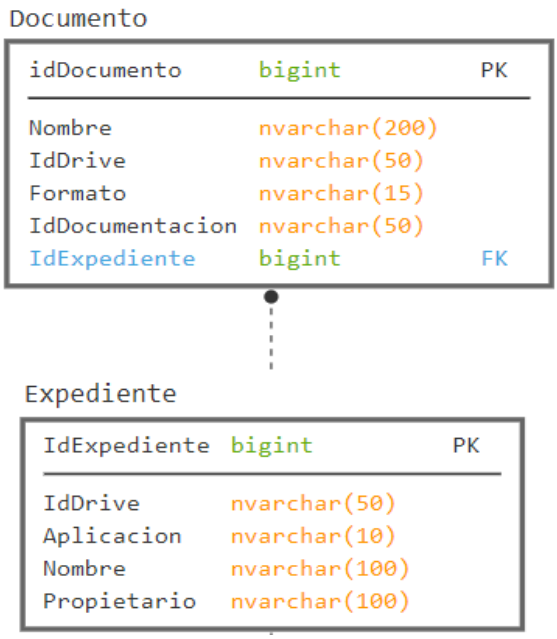


Imagen 4.5 Tablas de documentación

Una vez generadas estas tablas es posible construir la entidad central de la aplicación llamada prospecto, representada en la imagen 4.6 y que se usa como referencia de múltiples tablas mostradas más adelante

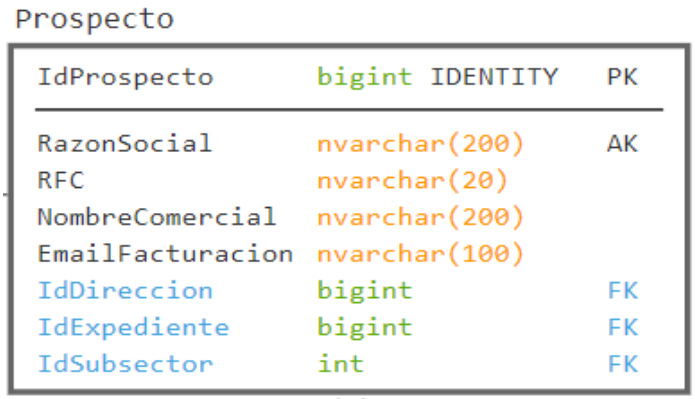


Imagen 4.6 Tabla Prospecto

Como lo indican los requerimientos es necesario contar con datos de los contactos, así como de los representantes de la empresa prospecto. Estos dos poseen una estructura similar sin embargo el representante contiene algunos datos de carácter legal como un número de escritura y el tipo de poder dentro de la empresa, lo cual sirve de guía para saber cuándo debe firmar o no un documento. Por otra parte, el contacto es meramente informativo y permite al área comercial tener una línea directa para comunicarse con la empresa. Ambas tablas se ilustran en la imagen 4.7.

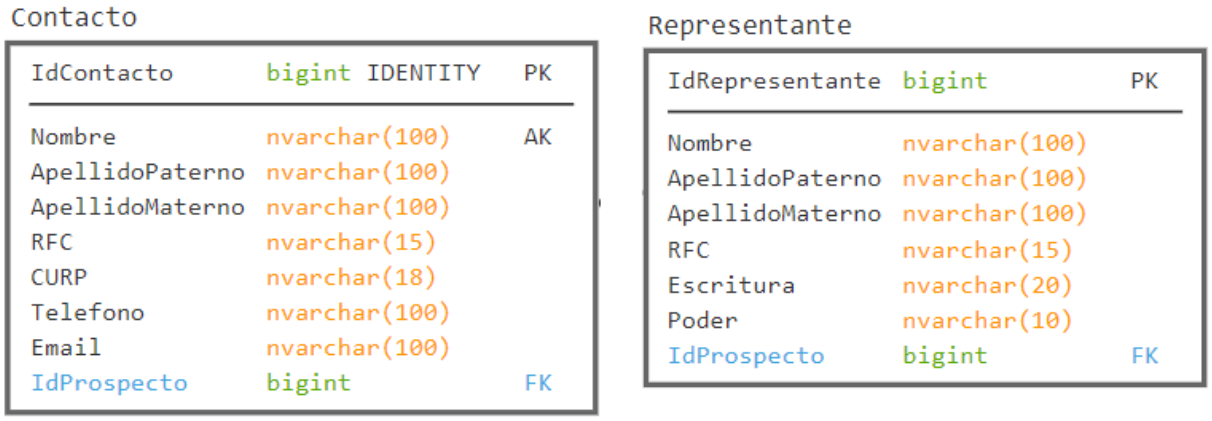


Imagen 4.7 Tablas de personas relacionadas

Por disposición legal estas personas deben ser analizadas para saber si están o no vinculadas con lavado de dinero o financiamiento al terrorismo mediante las llamadas listas negras. Esta información de consulta debe ser resguardada como evidencia del proceso. Se muestra la estructura en la imagen 4.8.

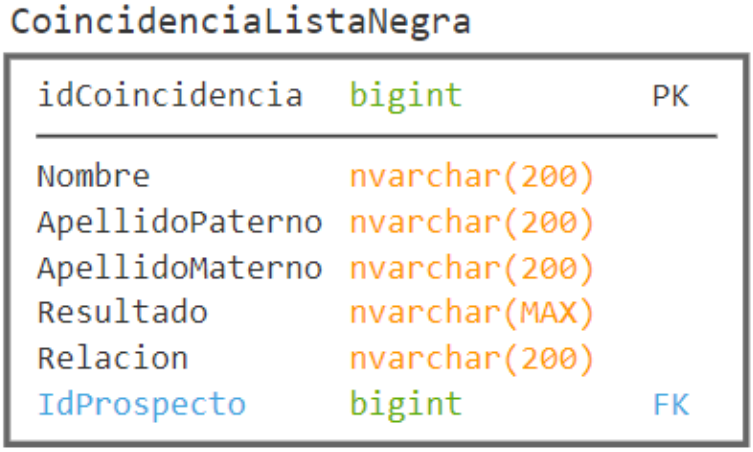


Imagen 4.8 Tablas de listas negras

Adicionalmente es necesario llevar el seguimiento del prospecto a lo largo del proceso de Onboarding por lo que se cuenta con dos estructuras, Control y Bitácora las cuales se ilustran en la imagen 4.9. La tabla de Control se encargará de tener el estado actual del cedente, referencia al Prospecto y a la Documentación para tener perfectamente bien ubicado que procesos ha cumplido y cuales están pendientes, así como su orden por otro lado la tabla Bitácora preservara un histórico de la actividad del prospecto, así como de los usuarios que han hecho cambios en alguno de sus procesos.

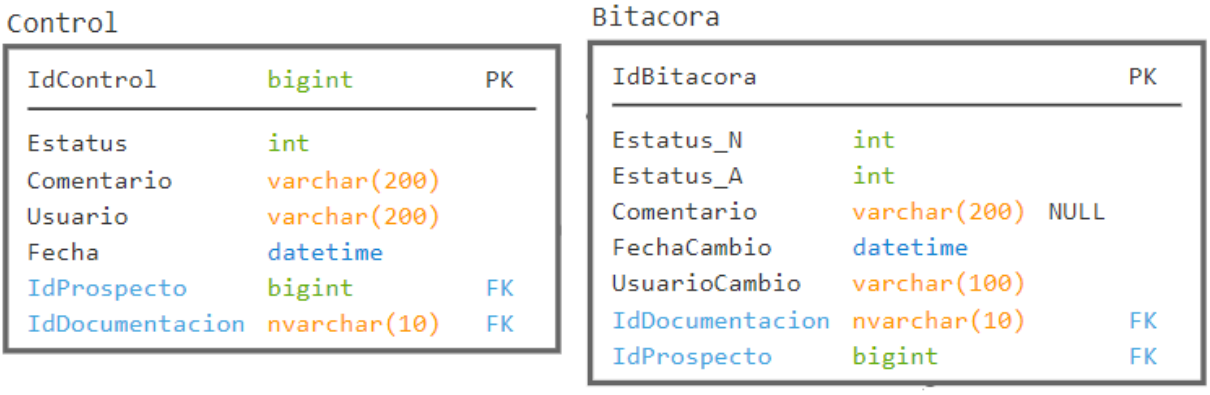


Imagen 4.9 Tablas de control

Estas tablas constituyen la parte más relevante del diseño de la Base de Datos, sin embargo, no muestran la definición completa. La forma en que estas tablas se relacionan se presenta en el diagrama de la imagen 4.10.

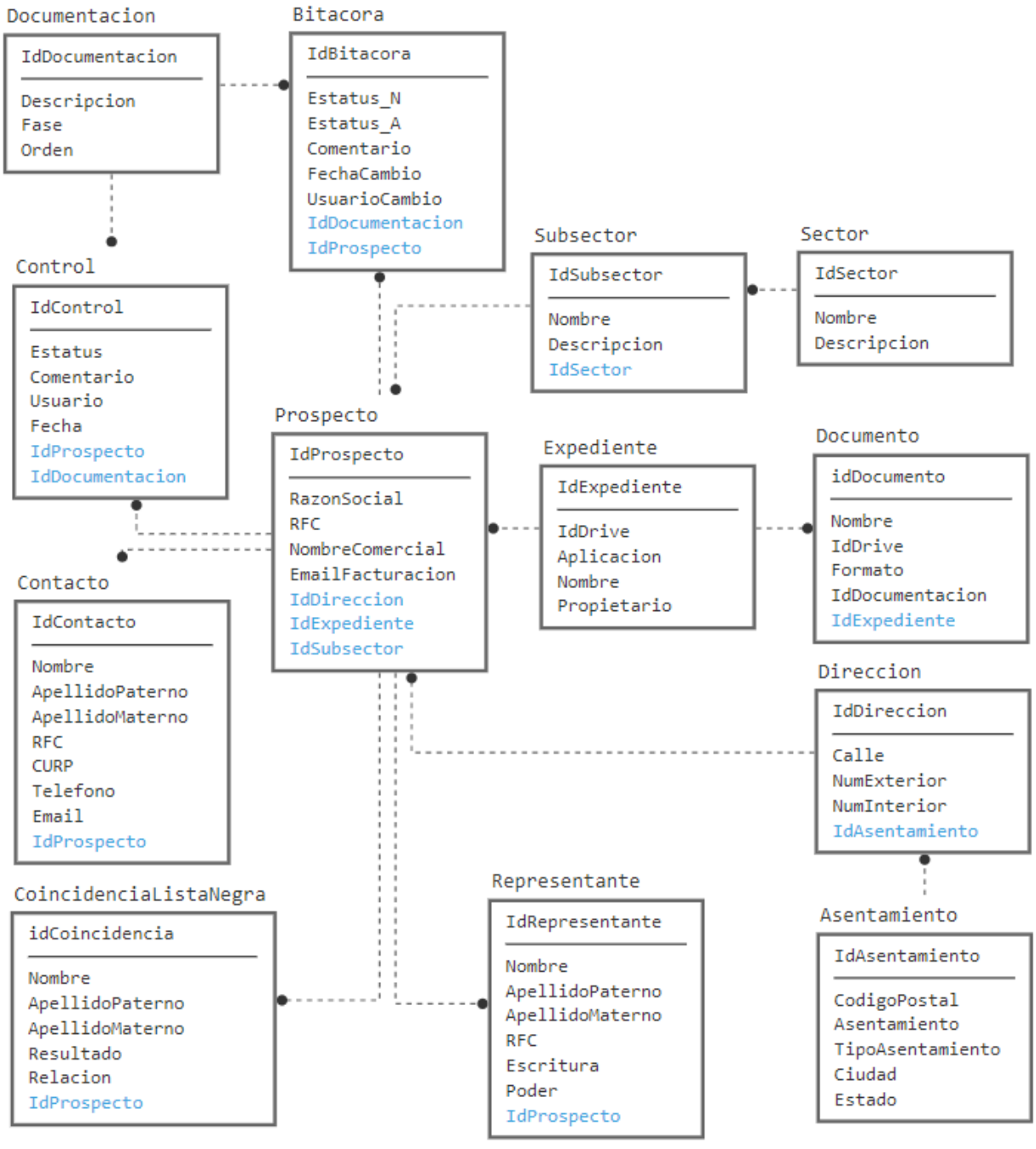


Imagen 4.10 Diagrama de base de datos

4.3 Componentes y dependencias

Una vez que se ha definido el patrón de diseño que se utilizará es importante diseñar los diferentes elementos que compondrán la solución de software y cuál será el nivel de dependencia que tendrán entre ellos, así como la dependencia de otros componentes externos, los cuales pueden ser de partners tecnológicos o bien de la propia empresa, pues como se ha mencionado anteriormente la empresa tiene diferentes servicios desarrollados previamente que deben estar incluidos en el proyecto. Los componentes empleados en el desarrollo de la plataforma Onboarding son los siguientes: Aplicación Web, Web API, Base de Datos y Servicio Externo.

Las **Aplicación web** son proyectos desarrollados con el Framework ASP.NET MVC y los lenguajes FrontEnd CSS, HTML y JS.

Las **Web API** están construidas únicamente con el Framework ASP.NET MVC, sin utilizar lenguajes FrontEnd.

Las **Bases de Datos** se encuentran dentro sistema de gestión de bases de datos relacionales (RDBMS) llamado SQL Server.

Los **Servicios Externos** son proporcionados por entidades ajenas a la organización en los cuales su arquitectura interna es irrelevante para el proyecto ya que su objetivo es proporcionar conectividad entre los servicios de dicho proveedor y la empresa.

En el diseño de la solución del Onboarding las API juegan un papel fundamental por lo cual es necesario profundizar un poco más en este punto. Un API (del inglés API: Application Programming Interface) es una capa de abstracción que permite implementar las funciones y procedimientos de nuestro proyecto, detallando solamente la forma en que son realizadas sin requerir una interfaz gráfica que mostrar al usuario, por lo que no utilizan los lenguajes FrontEnd.

En particular las API utilizadas son API RESTful es decir que las operaciones que realizan son solicitadas por peticiones HTTP entre las cuales las más importantes son POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar).

De esta forma es muy sencillo encapsular una determinada lógica sin la necesidad de repetirla en diferentes lugares, pues cada cliente que lo necesite, por ejemplo, una aplicación web o una móvil, puede intercambiar información usando las operaciones previas. Es importante decir que las peticiones se realizan en XML o JSON, ya que son los lenguajes de intercambio de información más usados.

Al tener un diseño basado en API´s volvemos nuestra aplicación más modular, más escalable y fácil de mantener ya que se tiene un conjunto de servicios enfocados a una labor en específico, permitiendo utilizarlos donde sea necesario.

De igual forma un cambio puede aplicarse de forma más sencilla pues este puede aislarse únicamente al API que lo contiene, haciendo que el impacto hacia los demás componentes sea mínimo o incluso nulo. Adicionalmente si las API se apegan al patrón de diseño Modelo Vista Controlador, como es el caso de este proyecto, el resultado será aún mejor.

En la imagen 4.11 se presentan los diferentes componentes de la plataforma Onboarding y su interacción.

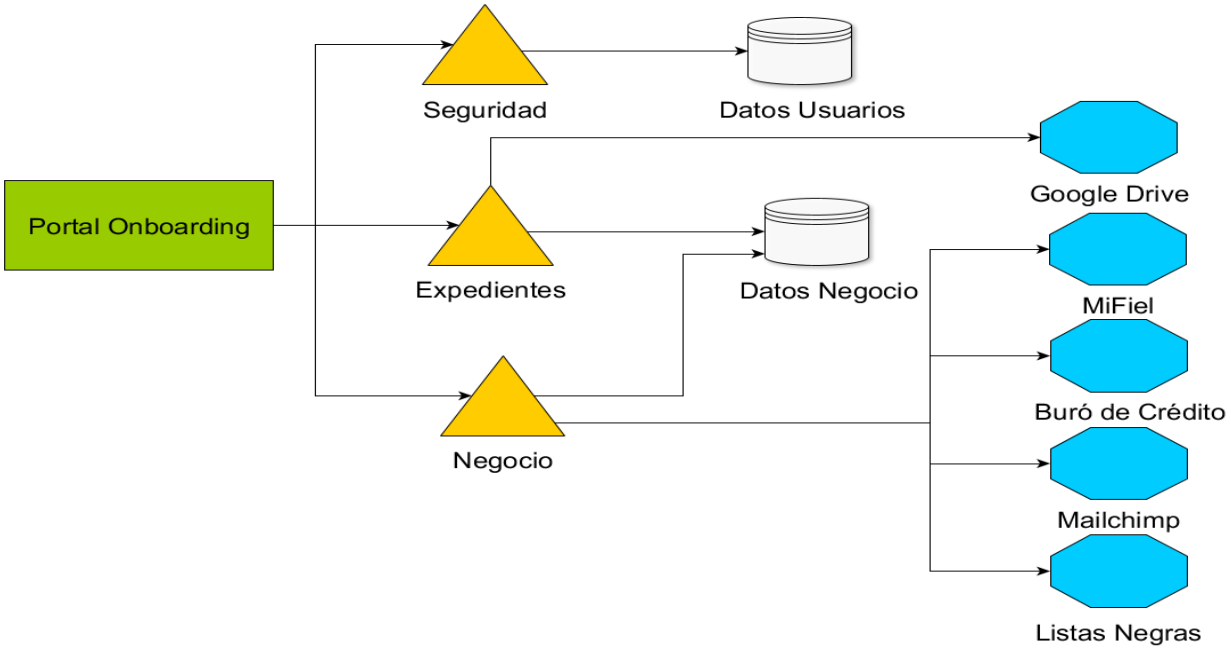


Imagen 4.11 Diagrama de componentes

Portal Onboarding: Es el sitio web al cual acceden los usuarios para dar seguimiento a los prospectos de la compañía, contiene todas las interfaces de usuario de la aplicación, solicita toda la información requerida a las API correspondientes y es el punto de interacción entre el usuario y los datos.

API Seguridad: Gestiona los usuarios, permitiendo crearlos o darlos de baja, asigna roles y permisos para cada usuario según la aplicación a la que acceda y se encarga de autenticarlos mediante un email y una contraseña. Aunque no se muestre en el diagrama hay múltiples aplicaciones de la empresa utilizando esta API para autenticarse. No requiere de servicios externos para operar y solo tiene dependencia con su propia Base de Datos.

API Negocio: Es la parte central de la aplicación ya que aquí se encuentran las reglas de negocio que describen el proceso de Onboarding para los prospectos, así como la interacción de los diferentes actores a lo largo del flujo. Es la encargada de interactuar directamente con la Base de Datos y manipular la información.

API Expedientes: Contiene la lógica correspondiente al manejo de documentación, permite a los clientes que la consumen manipular los documentos mediante las operaciones CRUD (Create, Read, Update, Delete), encapsulando el manejo de archivos y la conexión con los servicios de almacenamiento. Utiliza los servicios de Google Drive para mantener todos los archivos en la nube de forma segura.

MiFiel: Es el servicio que permite la firma electrónica de documentos a lo largo del proceso de Onboarding usando la Firma Electrónica Avanzada (FIEL) expedida por el SAT haciendo que los documentos tengan una validez oficial.

Buró de Crédito: Permite la conexión con esta entidad financiera, haciendo posible realizar consultas sobre los prospectos interesados.

Listas Negras: Es un servicio dedicado a consultar información sobre las personas físicas relacionadas a los prospectos para saber si tienen registros de actividades de lavado de dinero o financiamiento al terrorismo.

Mailchimp: Permite el envío de notificaciones mediante correo electrónico, permitiendo llevar un seguimiento más controlado.

Google Drive: Servicios proporcionados por Google para poder utilizar este medio de almacenamiento para la manipulación de archivos.

Como se puede ver la solución de software propuesta abarca diferentes componentes ya sea a nivel software o a nivel base de datos, sin embargo, no todos se explican en este documento, pues como se menciona al inicio del documento, solo se detalla en los componentes elaborados por mí y que su contenido no infringe las políticas de la compañía.

En el capítulo siguiente se detalla en la implementación de estos diseños elaborados previamente y se muestra cómo se construyeron los componentes de la solución de software listados a continuación: API Expedientes, API Negocio, API Listas Negras y Portal Onboarding.

Capítulo 5
API REST Documentación

5.1 Descripción de la API

Dentro del proceso de Onboarding trabajar con documentos es una tarea recurrente ya que a lo largo del proceso el prospecto debe subir su documentación al portal y el equipo de trabajo de la empresa debe, entre otras cosas, validarla, extraer información de los documentos y almacenarla para cumplir normas impuestas por disposición oficial.

Este acceso recurrente durante el proceso hace que en el sistema desarrollado se tenga un consumo continuo de esta funcionalidad y, por lo tanto, es fundamental que el manejo de documentos se encapsule en un componente independiente. Este componente se trata de un API que pueda gestionar los archivos y sea la responsable de subir, actualizar, eliminar y consultar los archivos de los expedientes digitales de la empresa.

Aunque no es objetivo de este documento mostrar desarrollos ajenos al Onboarding es importante resaltar que en otros proyectos de la empresa se requiere una lógica de administración de archivos similar a la que se presenta a continuación por lo cual el API diseñada y construida para el proyecto de Onboarding debe ser un componente genérico y reutilizable para diferentes proyectos.

El API desarrollado con ASP .NET implementa el patrón de diseño MVC mostrado anteriormente, solo que en este caso las vistas no serán documentos HTML que se muestran en un navegador web, serán documentos JSON que serán interpretados por los clientes del API.

5.2 Elección del servicio de almacenamiento

El componente fundamental para esta API es el servicio de almacenamiento que se utilizará para alojar todos los archivos. Para esta funcionalidad existen diferentes opciones sin embargo no todas se ajustan a las necesidades del proyecto, entre las opciones y proveedores valorados se encuentran Amazon, Azure, Google y algunos proveedores de servidores FTP / SFTP.

Un servidor FTP / SFTP permite la conexión de clientes para enviar o descargar archivos independientemente del sistema operativo que estos tengan. Aunque implementar este tipo de servicios no es complicado el problema es que los proveedores en la nube no ofrecen soluciones a la medida, por lo tanto, el espacio y el almacenamiento son muy limitados y costosos en comparación con otros productos. También existen opciones gratuitas de este tipo de servicios, pero son para infraestructuras locales y como se ha mencionado antes esto conlleva múltiples problemas, además las políticas de la compañía obligan a las soluciones de software a estar en alguna infraestructura en la nube para garantizar la disponibilidad hacia el negocio, por lo tanto, esta opción queda descartada.

Amazon cuenta con su producto *“Amazon S3”* (Amazon Simple Storage Service) y Azure de Microsoft tiene su producto “*Blob Storage”*. Ambos proveedores ofrecen excelentes servicios debido a que permiten el almacenamiento de cualquier formato de archivo que se requiera, adicionalmente ofrecen la posibilidad de escalar el espacio de almacenamiento tanto como se requiera, de forma progresiva según crezca la demanda de la aplicación. Ambos ofrecen una disponibilidad de los datos mayor al 99.99% además de ofrecer servicios de replicación y medidas de seguridad para ofrecer un servicio de calidad y con un buen rendimiento.
Desafortunadamente el costo de los proveedores se vuelve elevado debido a que la demanda de solicitudes del servicio es relativamente baja y no compensa el costo de una implementación como esta.

Por último, se tiene a Google Drive el cual ofrece una alta disponibilidad del servicio al igual que los dos anteriores, también es posible llevar un control de cambios y respaldo de archivos de una forma sencilla, así como gestionar permisos y accesos a los datos, adicionalmente toda la comunicación con este servicio se realiza mediante una biblioteca desarrollada de forma nativa en .NET proporcionada por Google y distribuida de forma gratuita.

Un punto adicional para este proveedor se debe a que la empresa ya tiene contratado un plan de Google Suite en su versión Business por lo cual ofrecen una mejora en todos los servicios ordinarios de Google para todas las cuentas de la compañía, incluyendo Google Drive, entre las ventajas que destacan es que se tiene un almacenamiento ilimitado de archivos.

Por lo tanto, se consideró a Google Drive como mejor alternativa para esta solución basado en las ventajas que ofrece y del costo de implementar la solución.

5.3 Estructura de un proyecto Web API

Para poder crear un API .net es necesario crear un proyecto de tipo “ASP.NET Web Application” en el IDE Visual Studio, como se muestra en la imagen 5.1.

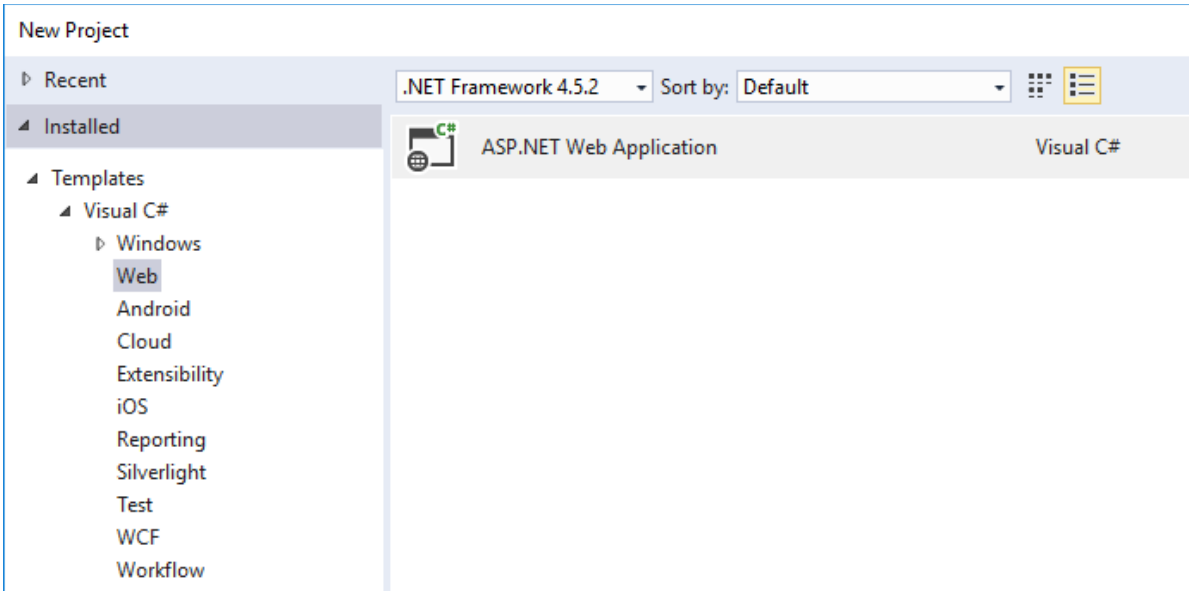


Imagen 5.1 Nuevo proyecto Visual Studio

Al momento de elegir una plantilla se selecciona MVC y Web API. Ver imagen 5.2.

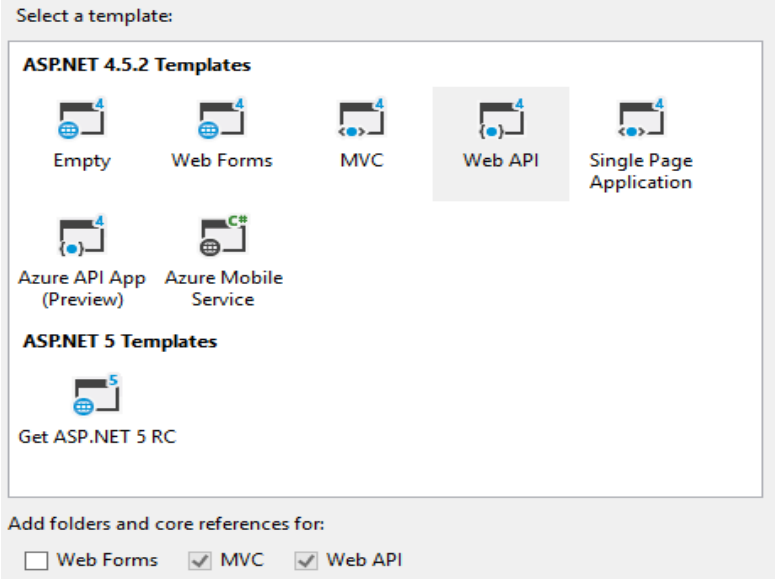


Imagen 5.2 Plantilla MVC Web API

El proyecto generado tendrá la estructura de la imagen 5.3.

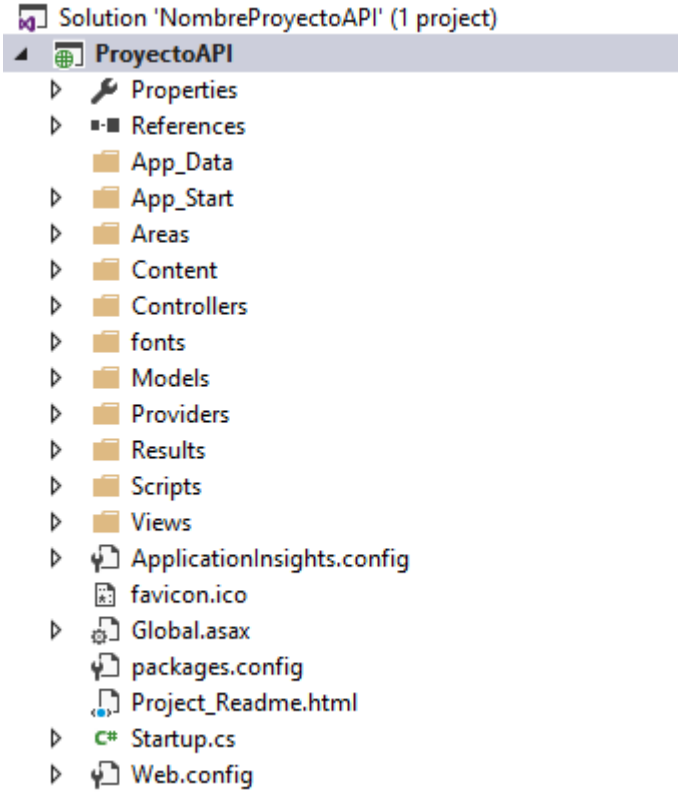


Imagen 5.3 Estructura de proyecto

Cada carpeta tiene un propósito específico dentro de la aplicación, algunas de las más destacadas son Controllers, Views y Models que almacenan respectivamente los Controladores, Vistas y Modelos de la aplicación, otra carpeta importante es Content la cual tendrá recursos propios de la aplicación como son imágenes y otros documentos estáticos y App_Start que contiene las configuraciones de arranque de la aplicación.

Otra alternativa para visualizar mejor la estructura de la aplicación puede ser utilizar el diagrama de componentes ilustrado en la imagen 5.4, este diagrama pertenece al API de Expedientes para los prospectos, en él se muestran los módulos principales que la conforman y su interacción con la Base de Datos de Negocio y con el servicio de almacenamiento de Google Drive.

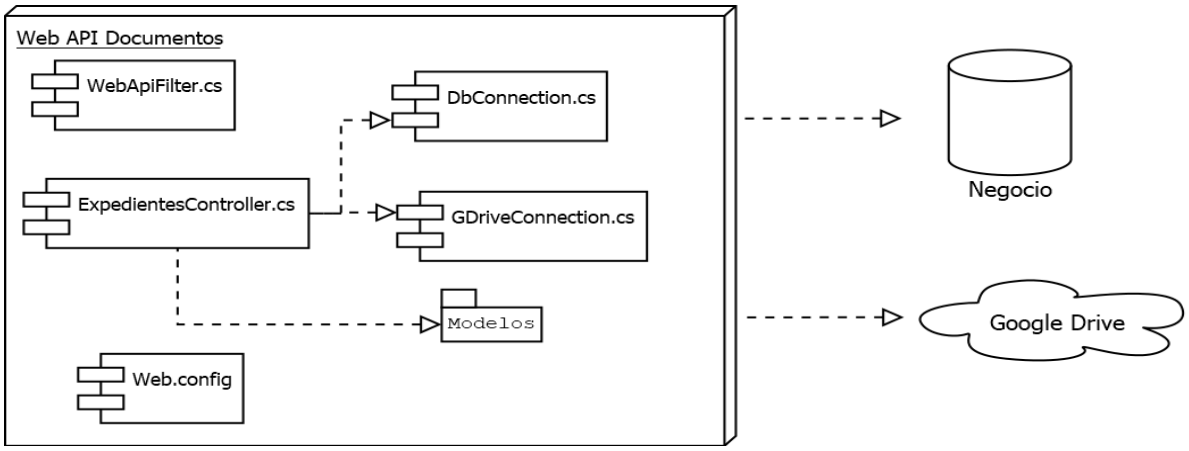


Imagen 5.4 Diagrama de clases para API

De manera interna el API está compuesta por un subconjunto de clases que definen su comportamiento, estas clases mostradas son ilustrativas y son la representación de un grupo mayor de clases que constituyen la solución. Las clases son las siguientes:

DbConnection: Interactúa con la Base de Datos de SQL Server.

GDriveConnection: Conecta el API con los servicios de Google Drive.

Modelos: Define la estructura de los datos.

Web.config: Almacena las configuraciones generales de la aplicación.

ExpedientesController: Controladores que llaman al modelo y la lógica de negocio para las solicitudes recibidas.

WebAPIFilter: Recibe las solicitudes HTTP y decide que acción ejecutar.

5.4 Peticiones HTTP

Una vez que se ha generado la estructura del proyecto se debe definir el conjunto de peticiones a las cuales responderá la Web API. Estas peticiones como se ha mencionado anteriormente son HTTP y de acuerdo con la intención de la petición debe usarse un tipo de método en particular. Es importante resaltar que el uso de HTTP o HTTPS como protocolo de transferencia es independiente del framework .NET y que solo depende del servidor en que se esté alojando la aplicación, en este caso de Azure, por lo cual y para efectos prácticos se muestra como HTTP únicamente. Los diferentes métodos HTTP que fueron utilizados para el desarrollo de las API del proyecto Onboarding se detallan a continuación.

GET: Se utiliza para obtener información de recursos solamente, no para modificarla de ninguna manera. Para cualquier solicitud exitosa se debe devolver el código de respuesta HTTP 200 (OK), junto con el cuerpo de la respuesta, en este proyecto es JSON, pero puede ser XML u otro formato.

HTTP GET http://www.appdomain.com/users
HTTP GET http://www.appdomain.com/document/123

POST: Es utilizado para crear nuevos recursos dentro de la aplicación. Idealmente, si se ha creado correctamente el recurso en el servidor la respuesta debería ser el código de respuesta HTTP 201 (Created) y contienen una entidad que describe el estatus del nuevo recurso, para este caso se devuelven los Id de los documentos y/o expedientes generados.

HTTP POST http://www.appdomain.com/users/create
HTTP POST http://www.appdomain.com/accounts/create

PUT: Se utiliza para actualizar un recurso existente, en caso de que el recurso no exista se puede o no crear el recurso de acuerdo con la naturaleza de la petición. Si se ha creado un nuevo recurso, el servidor debe responder con un código HTTP 201 (Created) y si se modifica un recurso existente, ya sea 200 (OK) o 204 (No Content).

HTTP PUT http://www.appdomain.com/users/123
HTTP PUT http://www.appdomain.com/documents/456

DELETE: Se utilizan para eliminar recursos. Una respuesta exitosa debe ser el código de respuesta HTTP 200 (OK) si la respuesta incluye una entidad que describe el estado o 204 (No Content) si la acción se realizó, pero la respuesta no incluye una entidad.

HTTP DELETE http://www.appdomain.com/users/123
HTTP DELETE http://www.appdomain.com/users/123/accounts/456

Hay una gran cantidad de códigos de respuesta para el protocolo HTTP independientes del tipo de método utilizado y según el resultado de la solicitud se regresa un código en específico, algunos de los más importantes se ilustran en la imagen 5.5.

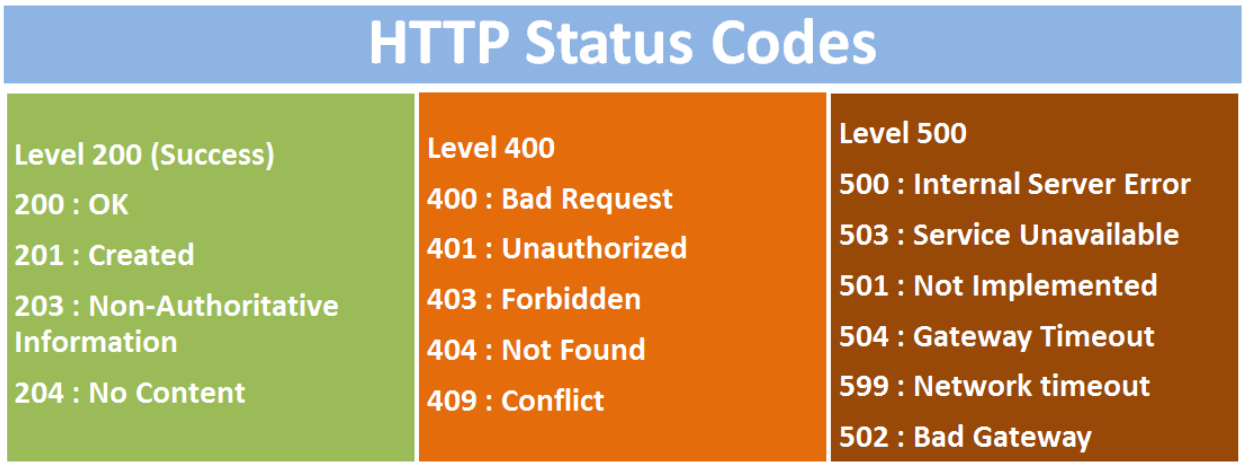


Imagen 5.5 Códigos de estatus HTTP

Dentro del proyecto creado las solicitudes HTTP que serán recibidas son definidas mediante los controladores los cuales están ubicados en la carpeta Controllers. En este nivel se encuentran los controladores del tipo MVC y API, los cuales funcionan de forma muy similar para el programador, aunque internamente poseen una funcionalidad diferente, esto se logra indicando la clase base de la cual se hereda, se usa la clase *"Controller"* para MVC y *"ApiController"* para la API

Controller MVC

```
public class HomeController : Controller
{
    // GET: Home/Index
    public ActionResult Index()
    {
        return View();
    }

    // GET: Home/Contact
    public ActionResult Contact()
    {
        return View();
    }
}
```

Controller API

```
public class ValuesController : ApiController
{
    // GET api/values
    public IEnumerable<string> Get()
    {
        return new string[] { "value1", "value2" };
    }

    // POST api/values
    public void Post([FromBody]string value)
    {
    }

    // DELETE api/values/5
    public void Delete(int id)
    {
    }
}
```

Imagen 5.6 Controladores

Como se puede ver en la imagen 5.6, se pueden devolver múltiples datos (o ninguno) como respuesta las solicitudes atendidas, de igual forma se generan de forma predeterminada algunas rutas para interactuar con los controladores, estas rutas tienen la estructura *{controller}/{action}/{id}* para MVC y *api/{controller}/{id}* para API.

Esto no siempre se adecua a las necesidades del proyecto, pero el framework de .net permite configurar las respuestas. Para las API del Onboarding se personalizaron tanto las rutas como los tipos de respuestas HTTP que se devuelven. En la imagen 5.7 se muestra un ejemplo

```
[RoutePrefix("api/documents")]
public class MiPrimerController : ApiController
{
    [HttpGet]
    [Route("details/{id}")]
    public IHttpActionResult MetodoGet(string id)
    {
        return Ok( new {id="89",file="..." } );
    }

    [HttpPost]
    [Route("create")]
    public IHttpActionResult MetodoPost()
    {
        return Created("document/123",new { id="123" });
    }

    [HttpPut]
    [Route("edit/{id}")]
    public IHttpActionResult MetodoPut(string id)
    {
        return Ok(new { id = "123" });
    }
}
```

Imagen 5.7 Métodos HTTP

El tipo de respuesta es un *IHttpActionResult* el cual es el indicado para peticiones HTTP en la versión WEB API 2 de .NET, y el código de la respuesta dependerá de la lógica del método, de igual forma se estableció la ruta para cada método y un prefijo para todo el controlador, dando como resultado rutas como las siguientes.

HTTP GET http://www.appdomain.com/documents/details/123
HTTP POST http://www.appdomain.com/documents/create
HTTP PUT http://www.appdomain.com/documents/details/123

De esta forma es posible personalizar tanto como se desee los métodos que responden a las solicitudes. En el API de Expedientes se implementó este mecanismo para generar las acciones CRUD (Crear, Leer, Actualizar y Borrar) tanto para los documentos, como para los expedientes, por otra parte, en el API de Negocio se implementaron las acciones CRUD de Prospectos, Accionistas y el resto de los modelos, así como para realizar tareas específicas del flujo del Onboarding. El uso de las funcionalidades del API de Negocio se detalla en capítulos posteriores, pero el mecanismo de implementación para la API es el mismo.

5.5 Control de peticiones

Una parte esencial del API es controlar el acceso a los recursos expuestos de la API, ya que se trata de información confidencial de los clientes y/o prospectos. Una forma de controlar este tipo de accesos al API es habilitando el estándar *Cross-Origin Resource Sharing*. Antes de mostrar su implementación es necesario conocer algunos puntos importantes.

La seguridad del navegador impide que una página web realice solicitudes AJAX a otro dominio. Esta restricción se denomina *same-origin policy* (política del mismo origen) e impide que un sitio malicioso lea datos confidenciales de otro sitio. Sin embargo, es necesario permitir el acceso a otros sitios en específico que consumen el web API.

Cross-Origin Resource Sharing (CORS) es un estándar de W3C que permite a un servidor relajar la política del mismo origen. Usando CORS, un servidor puede permitir algunas solicitudes de origen cruzado indicadas explícitamente mientras rechaza otras.

Una solicitud de origen cruzado se da cuando un cliente intenta acceder a los recursos del API. Como se muestra en la ilustración 5.8.

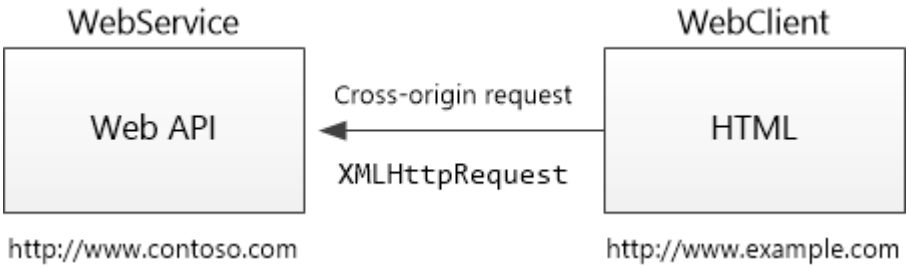


Imagen 5.8 Cross-Origin Resource Sharing

Para habilitar este tipo de accesos en el proyecto se agregó la configuración al archivo general de configuraciones y se agregó un atributo más a los controladores y/o métodos requeridos. [EnableCors(origins: "*", headers: "*", methods: "*")], donde cada parámetro indica una limitante.

Origins: delimita los dominios desde los cuales se puede acceder a los servicios del API, en este caso se colocaron los dominios de las aplicaciones de la empresa que consumen las API.

Headers: delimita las cabeceras que tienen las solicitudes que ingresan desde otros sitios, los valores que puede incluir son Origin, Content-Type, Accept, entre otros.

Methods: Indica los métodos HTTP que serán aceptados en las solicitudes, estos métodos pueden ser GET, POST, PUT, DELETE.

El atributo EnableCors puede ser agregado tanto a métodos o controladores limitando así el alcance de la regla. Si el atributo se queda configurado del modo mostrado ([EnableCors(origins: "*", headers: "*", methods: "*")]), se habilita el permiso para todos los orígenes de un controlador o método en particular.

Implementando CORS en controladores de las API es posible permitir o bloquear las peticiones que otros sitios y/o usuarios puedan realizar al API, por lo cual se mantiene un mejor control sobre los datos y se evita que alguien pueda obtener o modificar información de los expedientes de los prospectos o de su información personal.

La implementación de la solución y de medidas de control explicadas en este capítulo para el API de documentos son similares para el API de Negocio, lo que las vuelve diferentes son los valores establecidos para las URL de las solicitudes y su lógica interna que cada una maneja, la cual, por razones de confidencialidad, no puede ser mostrada. Por lo tanto, se toma este esquema presentado para entender la implementación de cualquier API utilizada en el proyecto de Onboarding.

En el siguiente capítulo se indica cómo la aplicación cliente consume las API de expedientes y de negocio para poder realizar las tareas correspondientes al flujo de un prospecto.

Capítulo 6
Portal web de usuarios

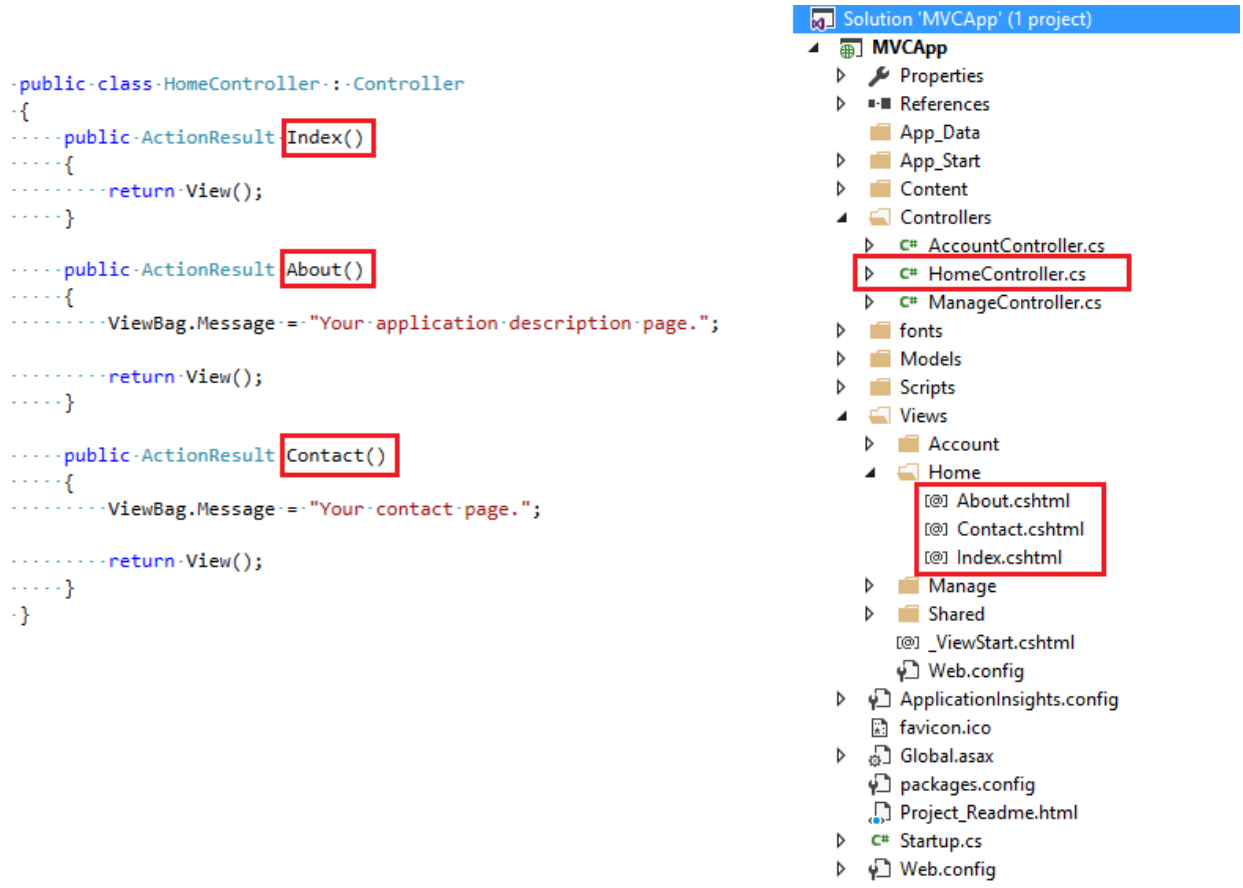
6.1 Descripción del portal

El portal Onboarding es el sitio web al cual acceden los usuarios para poder gestionar el progreso de los diferentes prospectos a lo largo de su proceso de Onboarding para la empresa. Este portal tiene el papel de FrontEnd de la solución de software, como se ha mencionado anteriormente, éste se conecta a los diferentes servicios y API´s para poder llevar toda la lógica de negocio de una forma modular y eficiente. En este capítulo se describe la implementación de algunas de las secciones más relevantes de la plataforma y su interacción dentro en el proceso de Onboarding.

El proyecto fue construido con el framework .NET MVC de Microsoft y de igual forma que una API, es posible crearlo con el IDE Visual Studio generando un proyecto de tipo “ASP.NET Web Application”.

La estructura de directorios de este proyecto es similar a la mostrada anteriormente, así como los mecanismos de programación utilizados. Entre las pequeñas diferencias están los tipos de respuesta de las peticiones, pues en la API se busca que la respuesta sea un JSON que contengan los datos, por otro lado, en un FrontEnd se busca devolver vistas en formato HTML que puedan ser presentadas en un navegador Web.

En el framework es posible elegir el tipo de dato de la respuesta de forma muy sencilla, mientras que para la API se empleaban objetos *IHttpActionResult*, ahora en el FrontEnd los métodos empleados devuelven objetos de tipo *ActionResult o* siendo más específicos objetos *ViewResult*, los cuales pertenecen a una clase que hereda de primera y nos representa la vista. La forma en que un controlador enlaza una vista con un determinado método se realiza por medio de una convención: Debe existir una carpeta con el mismo nombre del controlador en el directorio *Views* y por cada método del controlador debe existir un archivo .cshtml con el mismo nombre. Como se muestra en la imagen 6.1.



```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }

    public ActionResult About()
    {
        ViewBag.Message = "Your application description page.";

        return View();
    }

    public ActionResult Contact()
    {
        ViewBag.Message = "Your contact page.";

        return View();
    }
}
```


Imagen 6.1 Asociación de métodos HTTP y vistas

Los archivos .cshtml contienen la estructura html de la página, la cual es manipulada con los datos del Modelo (previamente obtenidos por el controlador) para mostrar el comportamiento deseado en la Vista. El resultado de este proceso es el documento HTML que será leído por el navegador del usuario.

Recordando el mecanismo de asignar rutas del framework y manteniendo el ejemplo anterior, se obtienen los siguientes enlaces para que el usuario navegue en los módulos del portal (Peticiones GET).

http://www.appdomain.com/Home/Index
http://www.appdomain.com/Home/About
http://www.appdomain.com/Home/Contact

De esta forma se fueron agregando todos los controladores y vistas necesarios para extender la funcionalidad de la aplicación y cubrir las funcionalidades del Onboarding.

6.2 Inicio de sesión en el sistema

6.2.1 Acceso de colaboradores

Como lo indican los requerimientos una parte indispensable es el control de acceso sistema, en este caso por medio de un correo y contraseña. En la pantalla de inicio mostrada en la imagen 6.2, se puede ver un formulario para la captura de estos datos.

Imagen 6.2 Vista de la pantalla de inicio de sesión

Una vez que el usuario intenta iniciar sesión, el portal debe validar su identidad a través de una API de seguridad. En caso de que el usuario exista la API regresará los datos correspondientes a la autenticación del usuario, como lo son el nombre, los roles dentro de la aplicación, etc. , este flujo se representa en la imagen 6.3.

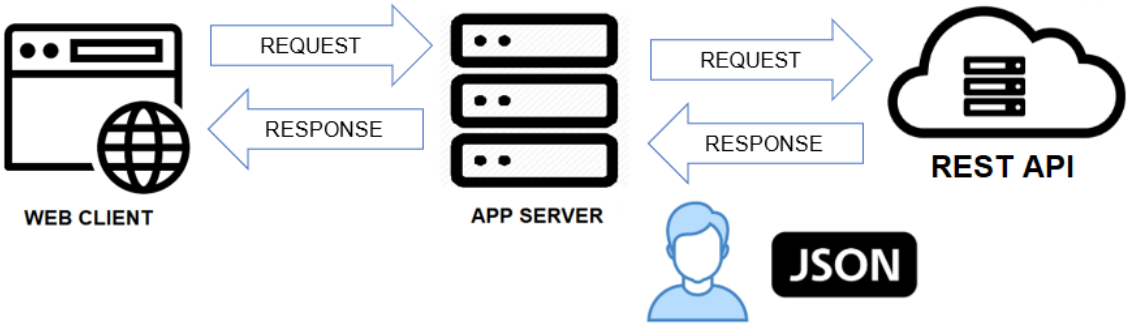


Imagen 6.3 Flujo de peticiones para ingreso de usuarios

El portal gestiona estos datos para configurar el usuario que ha iniciado sesión y de acuerdo con su rol dentro de la aplicación se le puede otorgar el acceso a determinadas funcionalidades.

La configuración de roles y permisos se puede hacer a nivel de controlador o a nivel de método, indicando de igual forma a quien se les permite el acceso, se ilustra en la imagen 6.4.

```
[Authorize]
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }

    [Authorize(Roles = "Comercial,Riesgos")]
    public ActionResult About()
    {
        ViewBag.Message = "Your application description page.";

        return View();
    }
}
```

Imagen 6.4 Autorización de usuarios

Existen diferentes roles en la aplicación, uno por cada área de la empresa, los más relevantes en el proceso de Onboarding son Riesgos, Legal y Comercial. Adicionalmente se crea un perfil administrador de sistema y otro para el prospecto. Los niveles de acceso a los módulos del proceso de Onboarding se muestran en la imagen 6.5.

Sección o Módulo	Comercial	Riesgos	Legal	Prospecto	Admin
Dashboard	✓	✓	✓	X	✓
Información General	✓	✓	✓	X	✓
Expedientes: Carga / Descarga	✓	✓	X	✓	✓
Expedientes: Validación	X	✓	X	X	✓
Captura de Información	X	✓	X	✓	✓
Proceso PLD	X	X	✓	X	✓
Contratos	X	✓	✓	X	✓
Estadísticas CIEC	✓	✓	✓	X	✓
Reportes	✓	✓	✓	X	✓

Imagen 6.5 Roles y permisos

Una vez que el colaborador de la empresa ha ingresado correctamente su usuario y contraseña puede visualizar el Dashboard de la imagen 6.6 que le muestra un panorama general de lo que ha realizado dentro del sistema.

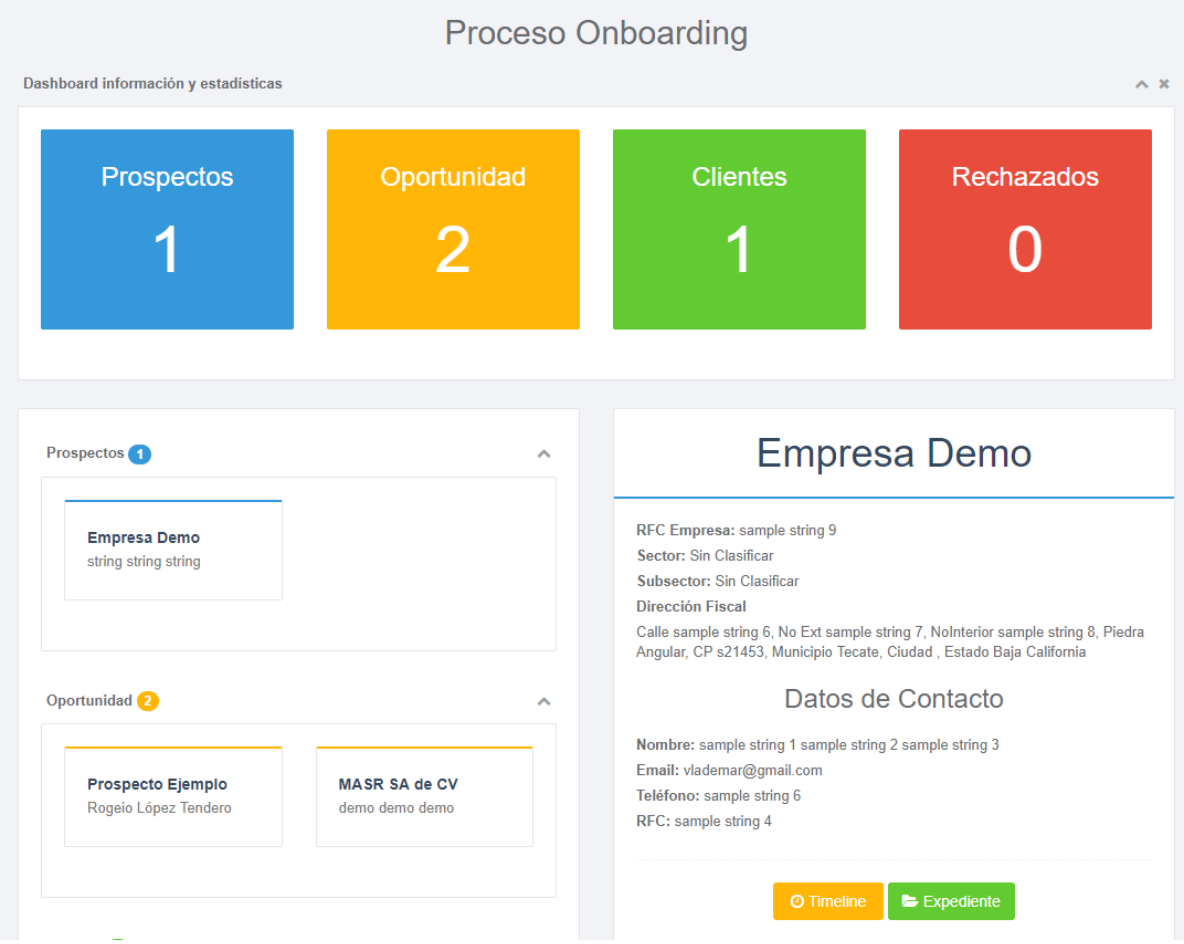


Imagen 6.6 Vista principal para colaboradores

De manera interna el prospecto puede pasar por diferentes fases y éstas son las que se representan en el dashboard principal, cada fase va acompañada de un número que indica cuántos usuarios se encuentran en esa fase.

Prospecto: Se considera prospecto desde el momento en que el usuario completa su registro inicial y proporciona algunos datos de contacto.

Oportunidad: Cuando un prospecto ha subido su documentación oficial y ha proporcionado información adicional existen más probabilidades de que pueda completar su proceso y se convierte en Oportunidad

Cliente: Una vez que el prospecto ha completado todo su proceso de Onboarding, adicionalmente ha proporcionado toda la información requerida y se han terminado las validaciones internas de negocio se procede a la firma de su contrato. Cuando el prospecto firma el contrato se vuelve cliente.

Rechazado: En algunas ocasiones los prospectos no cumplen con los procesos internos de la compañía y/o presenta problemas ante instituciones como el Buró de Crédito, o incluso sólo desisten de la búsqueda de los servicios de la compañía. En estos casos el prospecto pasa a estatus rechazado y finaliza su proceso.

En diagrama de la imagen 6.7 muestra las diferentes fases del ciclo de vida de un prospecto a lo largo del proceso de Onboarding.

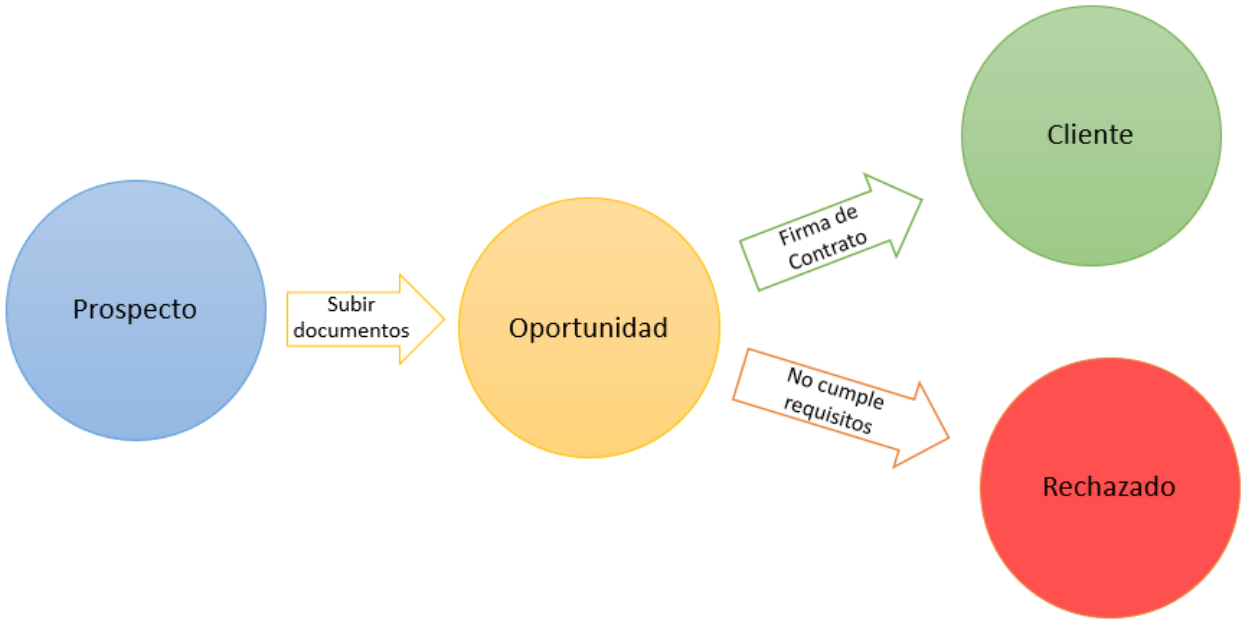


Imagen 6.7 Fases del prospecto

La forma en que el prospecto cambia de fase y los pasos para cambiar entre cada una se muestra en los siguientes puntos de este capítulo.

6.2.2 Acceso de prospectos

El prospecto no cuenta con accesos para poder ingresar a la plataforma, adicionalmente la vista que tiene es diferente de la de negocio y cuenta con las funcionalidades mínimas del sistema. Su objetivo es únicamente proporcionar información requerida por la empresa.

El primer paso del proceso es el registro, para ello es necesario completar los campos de un formulario como el que se muestra en la imagen 6.8 y 6.9, en el cual se capturan datos de la empresa, el representante legal entre otros, de acuerdo con las entidades presentadas en el diseño de Base de Datos.

Imagen 6.8 Registro de prospectos

Imagen 6.9 Registro de contactos

Cuando el usuario completa el formulario se envían notificaciones a los miembros de la compañía para que estén informados del nuevo usuario y puedan darle seguimiento a la brevedad.

El usuario prospecto puede acceder al sistema una vez haya completado su registro, pero su pantalla de inicio no es la misma que la de los usuarios de negocio. Su alcance dentro de la plataforma se limita a conocer en cual punto del proceso se encuentra y cuál es el siguiente paso por cumplir.

El portal funciona como punto de entrada para todos los documentos y toda la información adicional, además desencadena las alertas correspondientes y ayudar al personal a saber cuándo intervenir, volviendo el proceso más eficiente y controlado.

Imagen 6.10 Vista principal de prospectos

Como se puede ver en la imagen 6.10 se tienen diferentes colores que ayudan al usuario a saber el estatus de cada parte del proceso y cuál es el siguiente paso. De forma similar a este bloque de procesos se cuentan con bloques para cada proceso del flujo de Onboarding, desde la carga de documentos hasta la firma de contrato.

Algunos bloques pueden contener subtareas, las cuales tienen el mismo mecanismo de activación de acuerdo con la fase del proceso y con el mismo sistema de colores haciendo intuitiva la navegación para el usuario. El ejemplo más claro de subtareas son los bloques de documentación los cuales se explican a continuación

6.3 Módulo expedientes

Como ya se ha mencionado anteriormente el intercambio de documentos por parte del prospecto hacia la empresa es una tarea fundamental en el proceso de Onboarding, en el capítulo anterior se mostró el detalle de la API encargada de todo este manejo y procesamiento de archivos. En esta sección se muestra como la API es consumida por el cliente web.

Existen dos formas de interactuar con los expedientes, desde la vista del prospecto y desde la vista de Negocio. Ambas se muestran a continuación.

En la ventana *“Documentación Inicial”* el prospecto sube su documentación en el bloque correspondiente. Al subir la documentación sus bloques cambian de color indicando que el archivo fue enviado y está pendiente de revisión, como se ve en la imagen 6.11.

Imagen 6.11 Vista de prospecto para documentación

Internamente se tiene el control para tomar el archivo del equipo del usuario y enviarlo al servidor y, posteriormente, gestionarlo con la API de expedientes, ilustrado en la imagen 6.12. Así como enviar notificaciones a los usuarios encargados de la revisión.

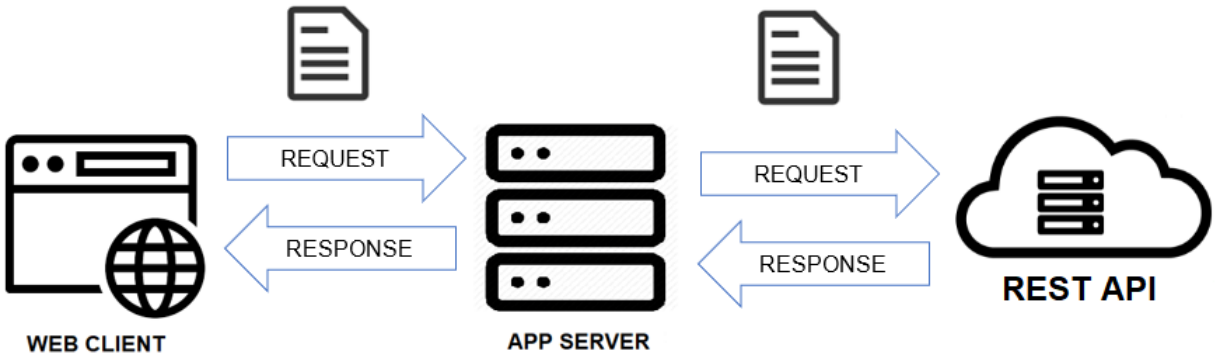


Imagen 6.12 Flujo de solicitudes de documentación

Para los usuarios de la empresa la vista presentada es diferente y se ilustra en la imagen 6.13.

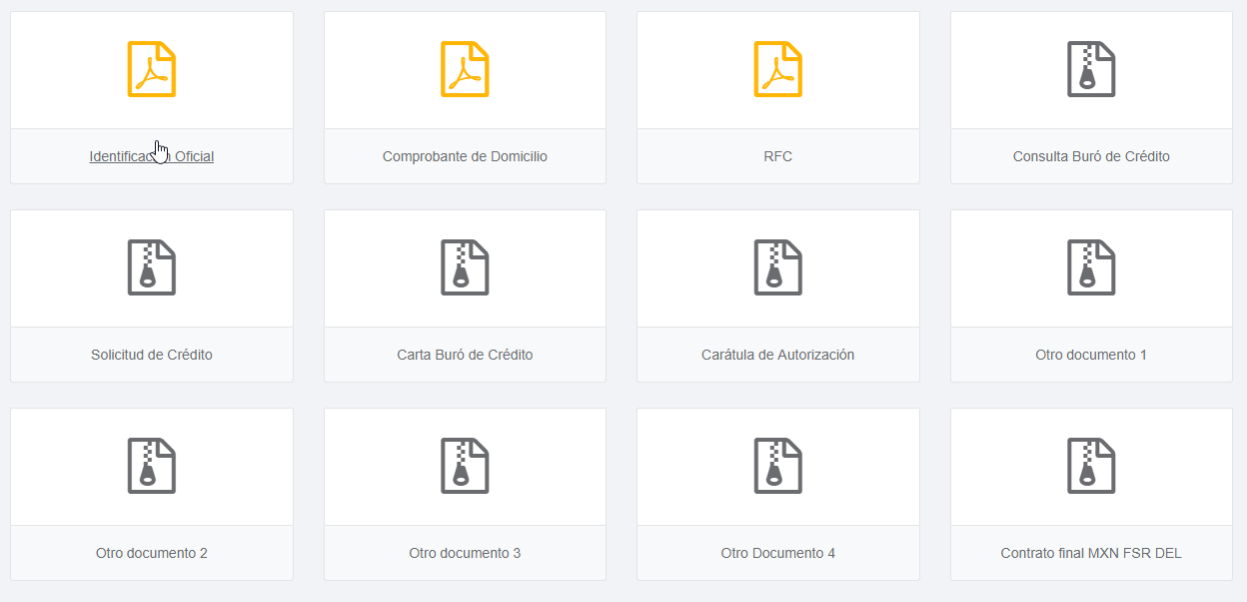


Imagen 6.13 Vista de colaboradores para documentación

En el cual se pueden ver todos los documentos posibles, así como el estatus en el que se encuentra cada uno. En esta misma sección el personal interno puede crear, actualizar, rechazar y leer los documentos, funciones proporcionadas por la API de Expedientes.

Una vez que el área responsable ha validado o rechazado la documentación del prospecto se envía una notificación al usuario para que continúe con el progreso o en caso de ser necesario envíe de nuevo la información correcta, completando de esa forma el flujo de envío de documentación.

6.4 Módulo prevención de lavado de dinero

La prevención del lavado de dinero (PLD) y el financiamiento al terrorismo (FT) se ha convertido en un tema serio y de alta prioridad a nivel internacional. Este conlleva graves implicaciones económicas y sociales ya que permite al crimen organizado financiar y mantener sus organizaciones delictivas, lo que además de generar inseguridad pública y desestabilización social, puede dañar severamente la reputación de ciertos sectores o entidades financieras y del país.

Como entidad financiera es necesario implementar mecanismos de PLD dentro del proceso de Onboarding para ello se desarrolló un módulo especial que permita realizar la consulta de accionistas, representantes, contactos y de cualquier persona relacionada con la empresa mediante un proveedor de listas negras.

La búsqueda en listas negras se realiza por medio de peticiones HTTP a un API del proveedor en la cual se envían las personas que se desean consultar y se obtienen los resultados encontrados. Esto se representa en la imagen 6.14.

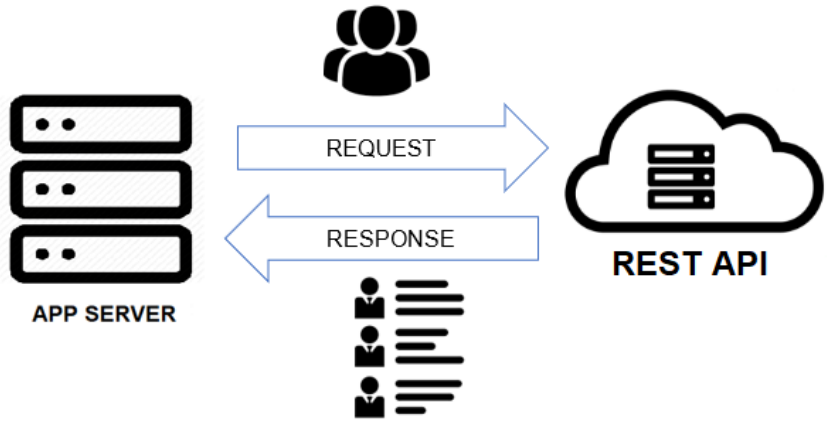


Imagen 6.14 Flujo de solicitudes para búsqueda en listas negras

Las búsquedas se realizan en catálogos de PEP (Personas Expuestas Políticamente) y en listas proporcionadas por el SAT y otras instituciones gubernamentales. Una Persona Expuesta Políticamente es aquella que es (o ha sido) asignado a una función pública por ejemplo políticos, funcionarios gubernamentales, judiciales o militares de alta jerarquía, así como sus cónyuges, sus parientes hasta el segundo grado de consanguinidad. De igual forma en estas listas se puede encontrar a personas relacionadas al narcotráfico y personas con problemas de evasión fiscal .

Una vez que se tienen los resultados de la búsqueda para las personas relacionadas al prospecto se pueden presentar al usuario, el cual podrá decidir qué acción tomar “Aprobar”, “Rechazar” o “Enviar a Revisión”. La pantalla se puede ver en la imagen 6.15.

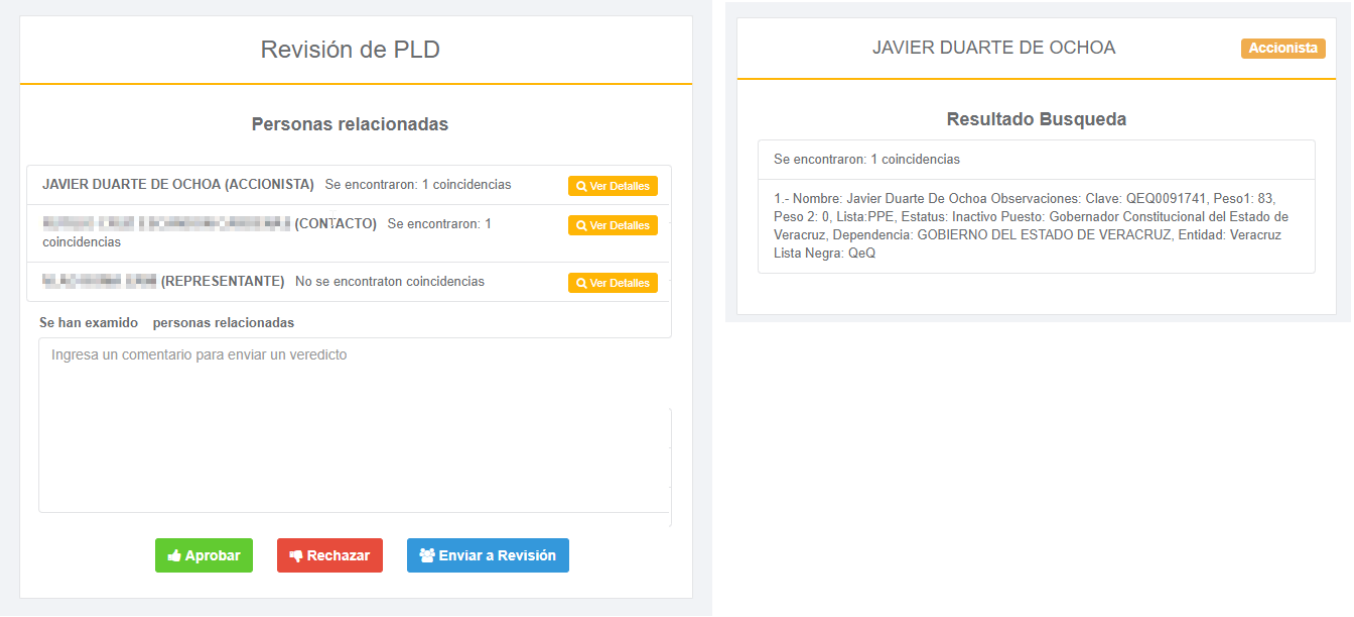


Imagen 6.15 Vista aprobación de PLD

Aprobar: Permite al prospecto continuar con su proceso de forma normal

Rechazar: Se bloquea al prospecto para no permitirle seguir el proceso de Onboarding y se le niega cualquier servicio de la compañía.

Enviar a Revisión: Se notifica al director general y al Oficial de cumplimiento de la empresa, los cuales en conjunto deberán realizar una investigación más exhaustiva y Aprobar o Rechazar al prospecto. Este escenario se puede presentar, por ejemplo, con algún homónimo, es decir, una persona que tiene el mismo nombre que otra que sí está en la lista, pero se trata de personas diferentes.

6.5 Módulo de contratos

El proceso final del Onboarding es el contrato, una vez que éste se completa el prospecto se convierte en cliente. El contrato es un documento institucional redactado por la financiera y rellenado por la compañía con datos específicos del prospecto, al igual que este documento se tienen otros con la misma naturaleza por ejemplo la solicitud de crédito, la carta de consulta a Buró de Crédito, entre otros. Para todos estos fue necesario implementar un mecanismo que permita optimizar esta labor que está sujeta a múltiples errores manuales momento de la captura de datos, en este apartado se muestra el contrato, pero el mecanismo es igual para los demás.

Todos los datos necesarios para el contrato pueden ser capturados mediante un formulario dentro del portal, esto es muy útil debido a que se puede implementar cualquier tipo de validación en los campos para garantizar que se están llenando de forma correcta, además de que pueden ser almacenados en la base de datos.

Una vez capturados los datos se usan para generar el documento, esta tarea se desarrolló con una biblioteca de JavaScript llamada *Pdfmake* que permite tomar un objeto JSON con el contenido del documento (en el formato específico de la biblioteca) y convertirlo en un archivo pdf.

Estos archivos en formato pdf son almacenados en el expediente y son presentados al usuario para su aprobación, como se puede ver en la imagen 6.16 y 6.17.

Imagen 6.16 Creación de contratos y documentos

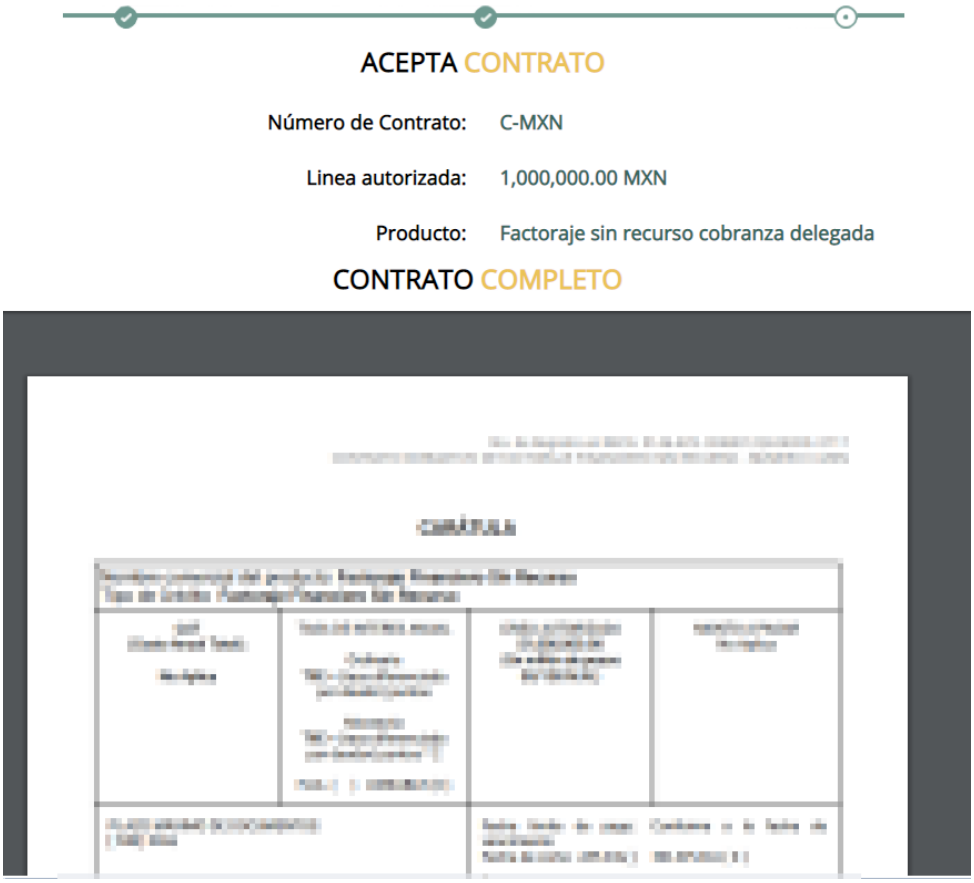


Imagen 6.17 Presentación de contratos

Una vez que se tiene el visto bueno del prospecto es necesario firmar documentos y que estos contengan una validez oficial, para ello se utiliza un servicio externo llamado MiFiel que implementa este mecanismo usando la firma electrónica (e.firma). De acuerdo con el SAT la e.Firma se define de la siguiente forma:
"La e.firma es un archivo digital que te identifica al realizar trámites por internet en el SAT e incluso en otras dependencias del Gobierno de la República. La e.firma es única, es un archivo seguro y cifrado, que tiene la validez de una firma autógrafa."

Las características antes mencionadas hacen que la e.firma sea ideal para firmar documentos, pues contienen la validez oficial que se requiere ante cualquier asunto legal que se pueda presentar, y al estar usando las credenciales personales del prospecto que obtuvo en el SAT únicas e intransferibles se obtiene un mecanismo de no repudio, es decir, no hay forma de que el firmante niegue la validez del documento. Es importante mencionar que nunca se almacenan las firmas del prospecto o la empresa y el único que tiene el conocimiento de estas claves es el usuario.

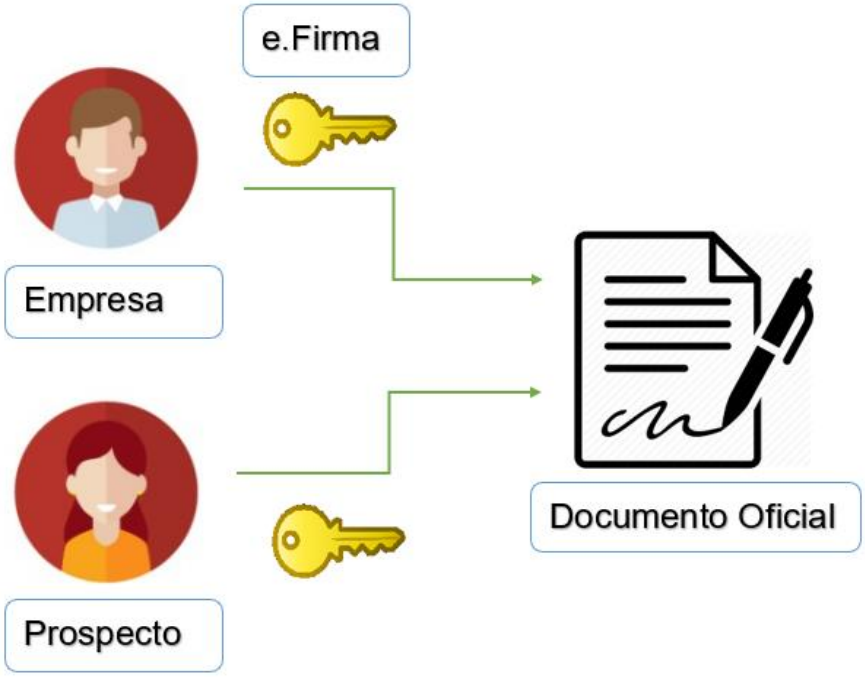


Imagen 6.18 Mecanismo de firma digital

Este mecanismo de firmas en el proceso de Onboarding se ilustra en la imagen 6.18 y es utilizado incluso en documentos para operaciones de la empresa con los clientes resultando ser bastante eficaz, ya que de no usarse sería necesario la firma autógrafa de los representantes legales ya sea para la fase de prospecto o de cliente, lo que implica entre otras cosas el traslado de los representantes, gastos de envío de documentos en servicios de mensajería y sobre todo tiempo.

Cuando el documento es firmado, el usuario deja de ser un prospecto y se convierte en un nuevo cliente, con lo cual puede acceder a los servicios de factoraje de la empresa, completando de forma satisfactoria su proceso de Onboarding.

Capítulo 7
Control y calidad

7.1 Control de versiones del código

Una parte fundamental del desarrollo de software es mantener el control sobre el código que se está generando, más aún si se está trabajando en un proyecto grande, es por ello por lo que se debe tener un sistema de control de versiones que ayude con esta labor. Hoy en día se cuentan con múltiples herramientas para llevar a cabo esta tarea y en este caso se eligió *Git debido* a su sencillez y robustez. Al tener el código del proyecto en un repositorio es mucho más sencillo implementar un control de versiones y con múltiples ventajas a no hacerlo, algunas de las más significativas se listan a continuación.

Trabajo en equipo: Es posible que múltiples desarrolladores se encuentren trabajando sobre el mismo proyecto, incluso sobre el mismo archivo de manera simultánea, compartiendo su código e integrando los cambios de los compañeros de forma sencilla y automatizada utilizando el repositorio de código.

Versionar el código: Se puede guardar en cualquier momento los cambios del proyecto y crear un punto de acceso para regresar a este más adelante en el tiempo. Esto resulta bastante útil cuando se necesita conocer el estatus en un punto específico del tiempo, por ejemplo, en una liberación, una mejora o simplemente el punto donde se comenzó a desarrollar. Con esto es innecesario crear copias constantes del proyecto para guardar cambios.

Visibilidad del proyecto: Debido a que todos los cambios de los desarrolladores están en un solo lugar es posible saber quién es responsable de cada movimiento y saber cuál es el aporte (en código) al desarrollo.

Desde la perspectiva del desarrollador debe existir un repositorio remoto en el cual se concentran todos los cambios del equipo, usualmente está alojado con algún proveedor de este servicio como GitHub o GitLab, pero nada impide que se encuentre en un equipo de la empresa, en este caso se usó GitLab. Cada desarrollador puede crear una copia del proyecto, realizar sus cambios y cuando estén listos puede compartirlos con el equipo a través del repositorio remoto, como se ve en la imagen 7.1.

Imagen 7.1 Trabajo con repositorios

Trabajar de forma paralela: En algunos casos es necesario que se realicen ciertas actividades de forma simultánea en el proyecto por ejemplo el desarrollo de alguna mejora y una corrección de un bug detectado, en estos casos es posible tomar diferentes caminos *“Braches”* para las distintas actividades, sin necesidad de deshacer el progreso, estas ramificaciones se ilustran en la imagen 7.2.

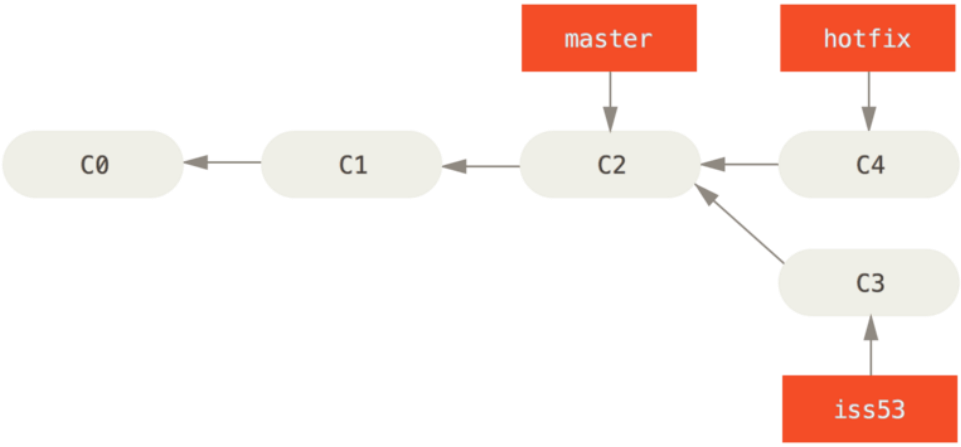


Imagen 7.2 Branches en Git

Por lo tanto, se puede concluir que implementar este tipo de herramientas es fundamental para poder llevar a cabo las actividades cotidianas del desarrollo de software. Algunos proveedores de servicio ofrecen adicional a git otro conjunto de herramientas para un mejor seguimiento de actividades, algunas se evalúan más adelante en este capítulo.

7.2 Proceso de pruebas

Una parte fundamental del proceso de software es el proceso de pruebas mediante el cual se garantiza el correcto funcionamiento del sistema, ya sea de forma total o parcial, se valida que cumpla con los requerimientos de negocio para los cuales fue diseñado y se verifica que cumpla con los estándares internos de calidad de la compañía.

El objetivo principal del proceso de pruebas es obtener información que permita medir la calidad del producto. Esta información es utilizada por las personas involucradas en el proyecto para corregir los defectos o bugs en caso de ser encontrados, de igual forma es utilizada como evidencia para aumentar la confianza del cliente al saber que recibe un producto de calidad.

Este proceso de pruebas debe realizarse en un ambiente completamente diferente al de desarrollo o producción para que sea efectivo, es importante destacar que entre mejor sea el proceso de pruebas se reduce la posible aparición de defectos en ambientes productivos. En el siguiente punto de este capítulo se profundiza en los diferentes ambientes de trabajo utilizados en este proyecto.

De acuerdo con la naturaleza de la prueba esta puede clasificarse en un cierto tipo en específico, en este documento solo se abordan algunos tipos como son las pruebas unitarias, pruebas de integración y pruebas basadas en requerimientos.

Pruebas Unitarias
Las pruebas unitarias comprueban componentes y métodos de software de forma individual. Prueban el código que controla el desarrollador y no deberían usarse para validar la infraestructura, como es la conexión a bases de datos o acceso a recursos externos. Debido a que prueban una unidad del código, sin dependencias externas, se deben ejecutar muy rápidamente.

En .NET existe la posibilidad de agregar pruebas unitarias de forma muy sencilla. Para ello es necesario crear un nuevo proyecto en Visual Studio del tipo *“Unit Test Project”* y agregar la referencia al proyecto que se desea probar, como se ve en la imagen 7.3.

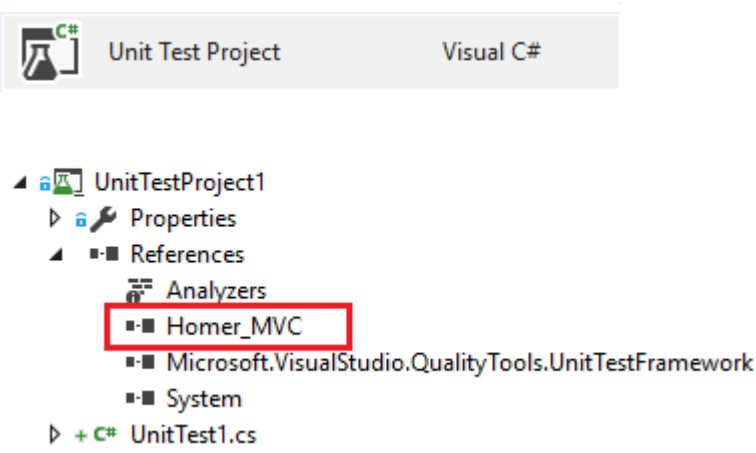


Imagen 7.3 Estructura de proyecto Test Unit

Los proyectos de pruebas unitarias son mucho más simples que los MVC o WebAPI, estos proyectos solo contienen clases como la que se muestra en la imagen 7.4.

```
[TestClass]
public class UnitTest1
{
    [TestMethod]
    public void TestCadenaTexto()
    {
        //... Obtener datos ...
        string x = "Rogelio";
        //Comparar con resultado esperado
        Assert.AreEqual("Rogelio",x);
    }

    [TestMethod]
    public void TestCalculo()
    {
        //... Obtener datos ...
        long calculoComplejo = 3 + 2;
        //Comparar con resultado esperado
        Assert.AreEqual(8, calculoComplejo);
    }
}
```

Imagen 7.4 Controlador de pruebas unitarias

Gracias a los atributos TestClass y TestMethod se indica que son parte del proceso de pruebas unitarias, como se mencionó antes estos deben ser métodos pequeños con validaciones simples por ejemplo la obtención de un texto, algún cálculo entre otras cosas. Cada método lleva una validación con la clase Assert que indicará el resultado de la prueba.

Una vez que se han establecido todas las pruebas requeridas pueden ejecutarse las veces que se desee. En cada ejecución de pruebas se muestra un informe indicando el estatus de cada prueba y el tiempo de ejecución. Este informe se muestra en la imagen 7.5.

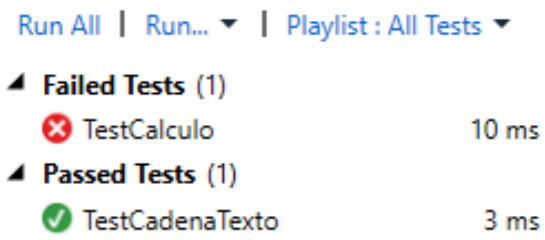


Imagen 7.5 Resultado de pruebas unitarias

De esta forma se puede validar que el código desarrollado cumple o sigue cumpliendo la funcionalidad para la cual se diseñó, lo mejor es que se realiza de forma automatizada con lo cual puede ejecutarse por el desarrollador en todo momento para garantizar que todo está funcionando correctamente a un nivel Unitario.

Pruebas de Integración

Las pruebas de integración se encargan de validar el código que interactúa con la infraestructura como bases de datos, sistemas de archivos y conexión con recursos externos como las API, adicionalmente comprueba que los componentes unitarios se comporten según lo esperado cuando interactúan unos con otros.

Estas pruebas son más lentas en cuanto al tiempo de ejecución y difíciles de configurar, por lo cual se recomienda que cualquier tarea que pueda hacerse con una prueba unitaria se haga de esa forma y no se una con la lógica de la prueba de integración. Usualmente estas pruebas no son ejecutadas por los desarrolladores

Pruebas de Aceptación

Estas pruebas son una aproximación sistemática al diseño de casos de prueba donde el usuario considera cada requerimiento y deriva un conjunto de pruebas para cada uno de ellos. Estas pruebas se centran más en la validación en lugar de la búsqueda de defectos, con lo cual se demuestran que el requerimiento ha sido plasmado en el software de forma correcta.

En la imagen 7.6 se muestra un ejemplo para la carga de documentos, tomando como referencia el siguiente diagrama, producto de los requerimientos del usuario, con lo cual es posible diseñar un conjunto de pruebas.

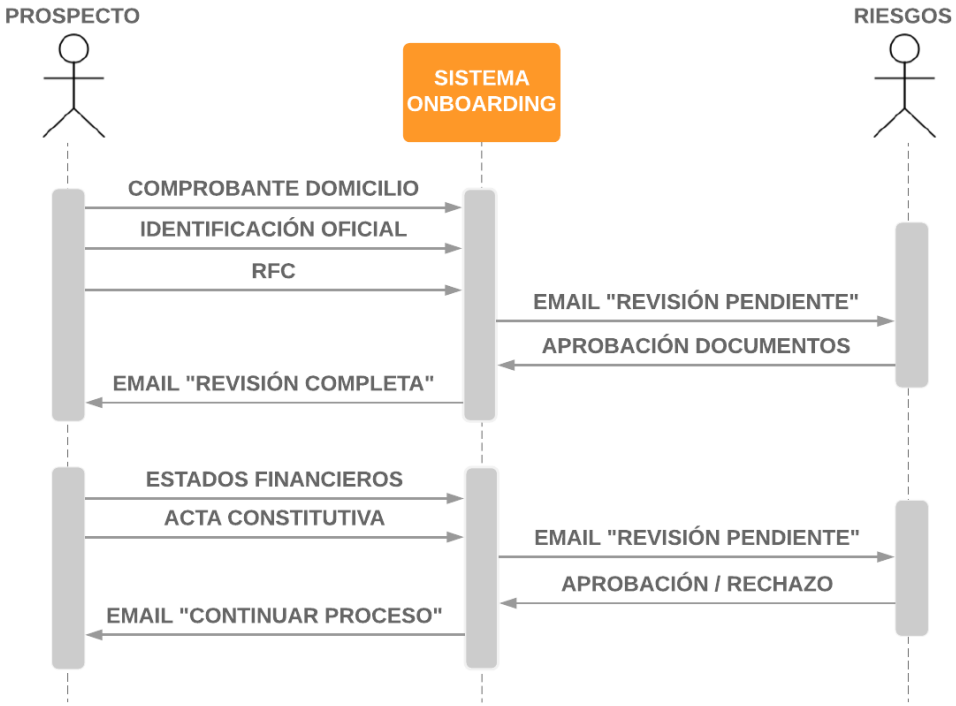


Imagen 7.6 Carga y validación de documentos

Escenario: Carga y validación de documentación Inicial

Requisitos Iniciales: Se debe tener un usuario con perfil Riesgos y un prospecto que haya confirmado su cuenta

Script de Prueba

Paso 1: Ingresar a la cuenta del prospecto con su correo y contraseña.
Paso 2: Ingresar a la sección Documentación Inicial
Paso 3: Arrastrar un archivo pdf, png o jpg hasta la casilla Comprobante Domicilio, Identificación Oficial y RFC
Paso 4: Validar recepción de email en cuenta de usuario Riesgos
Paso 5: Ingresar al portal como usuario Riesgos
Paso 6: Ingresar a la sección “Mi expediente” del prospecto
Paso 7: Marcar documentos Comprobante Domicilio, Identificación Oficial y RFC con estatus Aprobado
Paso 8: Validar recepción de email en cuenta de usuario Prospecto

En este caso se muestra un caso en particular de un requerimiento, de igual forma se tiene un script alterno para el caso de documentos erróneos, documentos rechazados y para el resto de los requerimientos y sus variantes. El éxito de la prueba dependerá de si fue posible realizar el script de forma consecutiva y se completó correctamente cada uno de los pasos.

Como se puede observar cada una de las pruebas tiene su propósito en específico, así como su momento de ser ejecutada y su responsable por lo cual no deben mezclarse sus propósitos. El tener múltiples pruebas documentadas del producto que se está desarrollando, cada una con su propio objetivo, brinda la seguridad a todos los involucrados en el proyecto de su calidad y confiabilidad. Una vez que las pruebas han sido exitosas es necesario pasar a la siguiente fase del desarrollo de software: la liberación.

7.3 Ambientes de trabajo y proceso de liberación

Como ya se ha mencionado anteriormente una forma de garantizar la calidad del desarrollo de software es tener diferentes ambientes de trabajo, independientes uno del otro. En este desarrollo se utilizaron 3 diferentes: Desarrollo, QA y Producción. En otras empresas y dependiendo de los recursos de los que se dispongan se pueden tener más o menos ambientes. En múltiples casos se tienen solo dos ambientes: Desarrollo y Producción, sin embargo, tener un solo ambiente es una muy mala práctica y es susceptible a muchos errores.

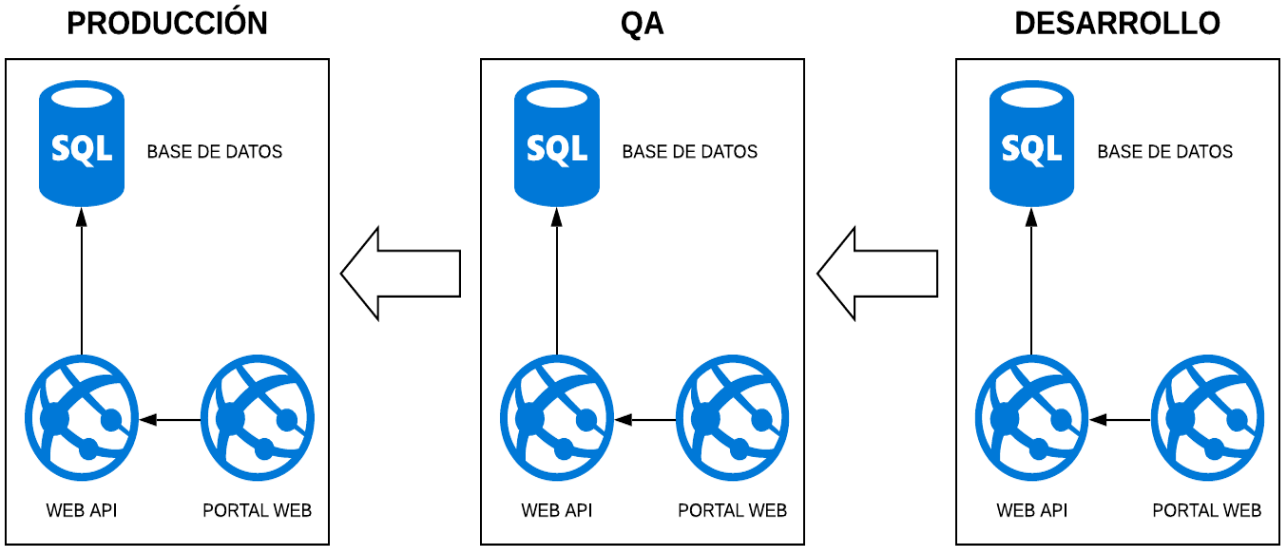


Imagen 7.7 Ambientes de trabajo

Como se puede ver en la imagen 7.7 un ambiente está constituido por una Base de Datos, el portal de Onboarding y el conjunto de API´s que se emplean. Cada vez que un desarrollo se ha completado comienza un proceso de liberación de cambios, desde el ambiente de desarrollo al ambiente de QA, posteriormente cuando las pruebas han sido concluidas y el software aprobado comienza la liberación de cambios del ambiente de QA al ambiente de producción. cómo se puede ver cada ambiente tiene un propósito en específico.

Desarrollo: Ambiente sobre el cual trabajan los programadores, la mayor parte del tiempo no tiene una versión completa ya que es aquí donde están los cambios de las mejoras y nuevos requerimientos en desarrollo.

QA: Una vez que se tiene una versión estable en desarrollo, se hace la migración de cambios a este ambiente. Aquí se realizan las pruebas de la solución completa, validando que cumple con los requerimientos del usuario y que lo haga de forma correcta, esta labor se realiza por parte de la gerencia de QA.

Producción: Una vez que la gerencia de QA ha determinado que el producto es correcto se libera la nueva versión en este ambiente, el cual tiene la operación de los usuarios de la empresa.

El tener tres ambientes de trabajo diferentes ofrece múltiples ventajas, siendo la más importante la calidad del producto, ya que se garantiza que el software es de calidad pues ha pasado por todas las validaciones necesarias tanto por desarrolladores, testers y líderes de proyecto.

Así como se tienen ventajas también hay un par de desventajas de tener 3 ambientes de trabajo como son los costos de implementación y mantenimiento, así como el control de contenido que hay en cada uno. Afortunadamente estos pueden ser mitigados gracias a la infraestructura de Azure, por ejemplo, se pueden tener Bases de datos de menor capacidad de almacenamiento y procesamiento para los ambientes de QA y Desarrollo ya que el volumen de información es menor, de esta forma son menos costosos que la Base de Datos destinada para ambiente de Producción.

En el caso de los recursos de tipo WebApp, como son el portal web y las API, es posible crear diferentes *“slots”* o ranuras de implementación usando el mismo recurso de la infraestructura Azure y sin elevar los costos, el objetivo es crear una ranura para cada ambiente (desarrollo, QA y producción).

Aunque un grupo de ranuras esté dentro del mismo recurso, cada una se puede ver como un recurso completamente diferente ya que cada ranura tiene asociado su propio archivo de configuraciones (Ajustes de aplicación y Cadenas de conexión a Base de Datos), su propia URL de acceso y sus propias credenciales FTP para hacer un despliegue del sitio.

La utilidad de crear slots en el recurso WebApp es que permiten tener los 3 ambientes de trabajo en un solo lugar, adicionalmente es posible intercambiarlos entre sí de forma muy rápida y sencilla, por lo tanto, cuando se ha aprobado una versión y hay que liberar los cambios en el siguiente ambiente solo hay que intercambiar las ranuras correspondientes, sin la necesidad de hacer el despliegue de un nuevo compilado y sin afectar la actividad de los usuarios, como se ve en la imagen 7.8.

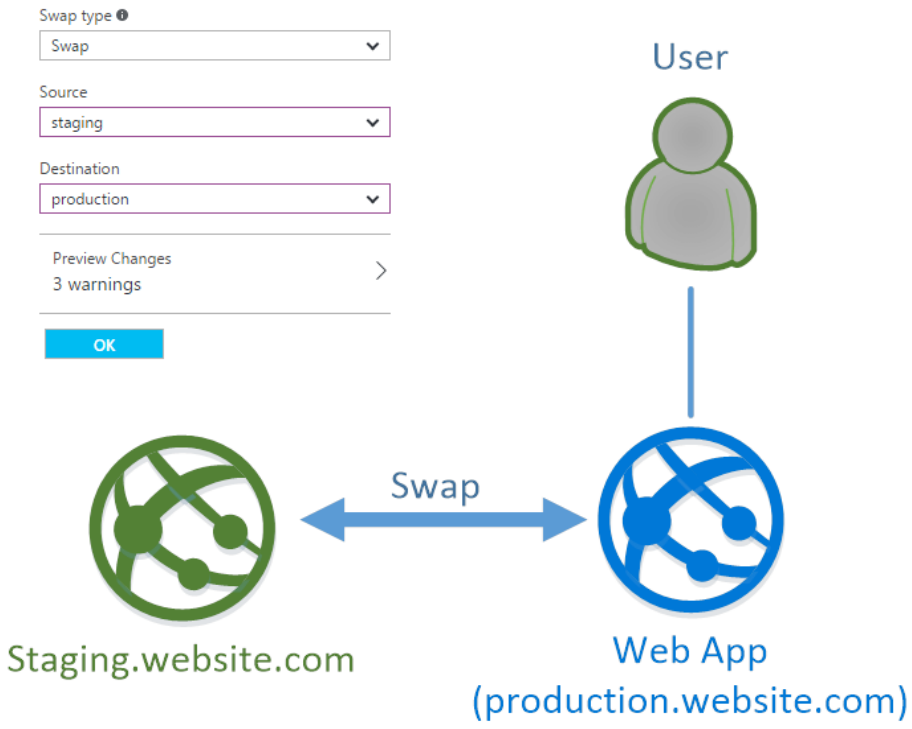


Imagen 7.8 Despliegue de cambios de Staging a Production

Esta es una característica extremadamente útil ya que facilita mucho las labores de liberación de cambios, evita los errores de configuración al momento del despliegue, ayuda en la gestión de recursos en la infraestructura de Azure, ya no que es necesario crear un recurso por cada ambiente de trabajo y asegura que las versiones liberadas sean exactamente las mismas que han sido probadas previamente.

Para el caso de las Bases de Datos por desgracia no se tiene una característica como esta, sin embargo, es posible realizar comparaciones y aplicación de cambios de una BD origen a una BD destino utilizando el IDE Visual Studio, haciendo que la labor de liberación de cambios de BD sea mucho más sencilla y automatizada. Adicionalmente la infraestructura de Azure realiza respaldos continuos de la BD con lo que se permite hacer rollbacks completo en caso de que los cambios aplicados no fueran los correctos.

De esta forma es posible realizar liberaciones continuas de cambios entre ambientes de forma automatizada, sin afectar la operación de los usuarios y sobre todo con la certeza de que se están liberando cambios que han sido probados y validados previamente.

7.4 Soporte y atención a usuarios

Una vez que el software ha sido entregado al cliente, con todas las validaciones necesarias tanto de negocio como del mismo equipo de tecnología, el software comienza su fase de maduración en la cual ya es utilizado por los usuarios finales y comienza a operar de forma productiva, cumpliendo así su propósito para el cual fue concebido. Sin embargo, todo producto de software es susceptible a mejoras ya sean nuevas funcionalidades, mejoras en las ya existentes o de igual forma se reciben peticiones para la capacitación de los usuarios.

Sea cual sea la necesidad del usuario es importante dar un seguimiento correcto del producto ya que esto garantiza la satisfacción del cliente con el producto entregado, para cumplir con esta importante tarea se utilizó en este proyecto *“Jira Software”* el cual es una herramienta en línea para la administración de tareas de proyectos, el seguimiento de errores e incidencias y para la gestión operativa de proyectos. El cual se muestra en la imagen 7.9.

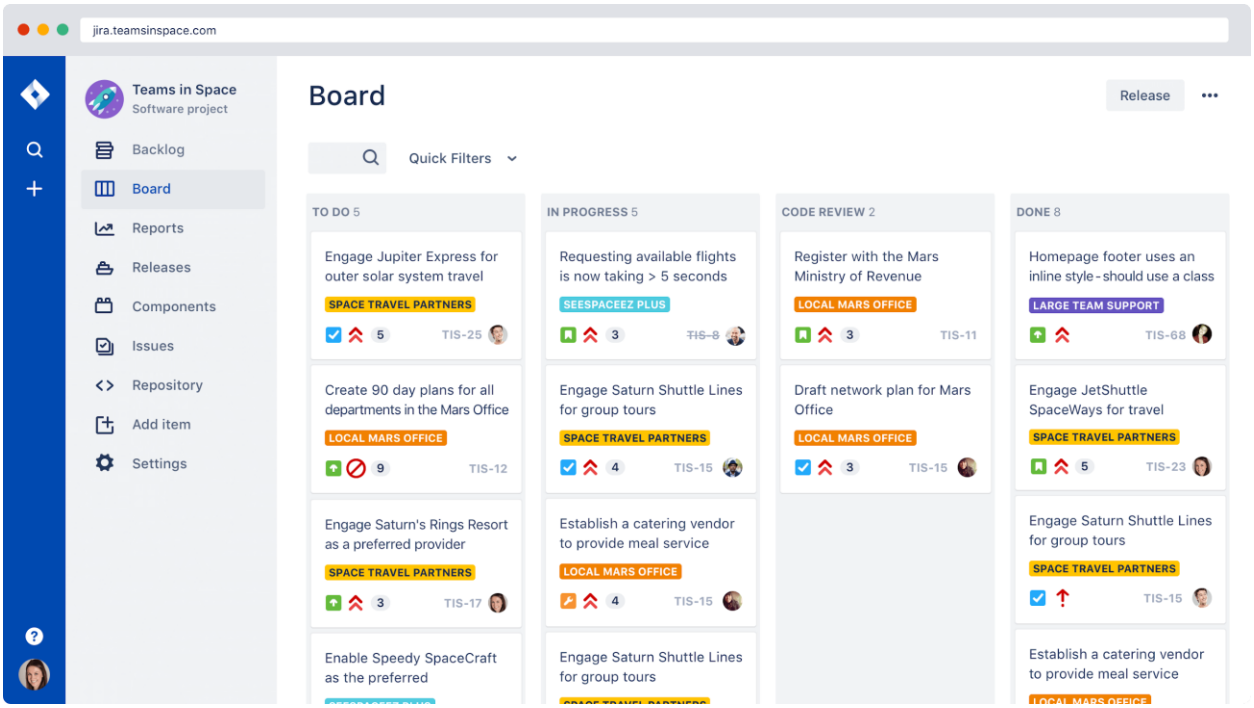


Imagen 7.9 Tablero de tareas de Jira

Gracias a este tipo de herramientas es muy fácil tener control de las solicitudes del usuario, clasificarlas de acuerdo con su nivel de prioridad y seguir un flujo de trabajo para resolverlas a la brevedad. Adicionalmente permite llevar registro del tiempo de respuesta de la solicitud y del tiempo que tardó en cada una de las fases.

El flujo normal de las solicitudes comienza cuando el usuario levanta su ticket y finaliza con la aceptación de este por parte del usuario una vez que fue resuelto. Sin embargo, en ocasiones la solicitud no es tan sencilla y requiere un desarrollo adicional para poder cumplir la demanda del usuario, para estos casos se debe de iniciar un nuevo desarrollo, este proceso es similar al descrito a lo largo de este documento sólo que mucho más simplificado pues solo se trata de una funcionalidad específica de un módulo en particular. Gracias a la herramienta Jira se puede gestionar estos pequeños desarrollos de una forma más ágil, de esa forma la solicitud pasa por los estados de la imagen 7.10.

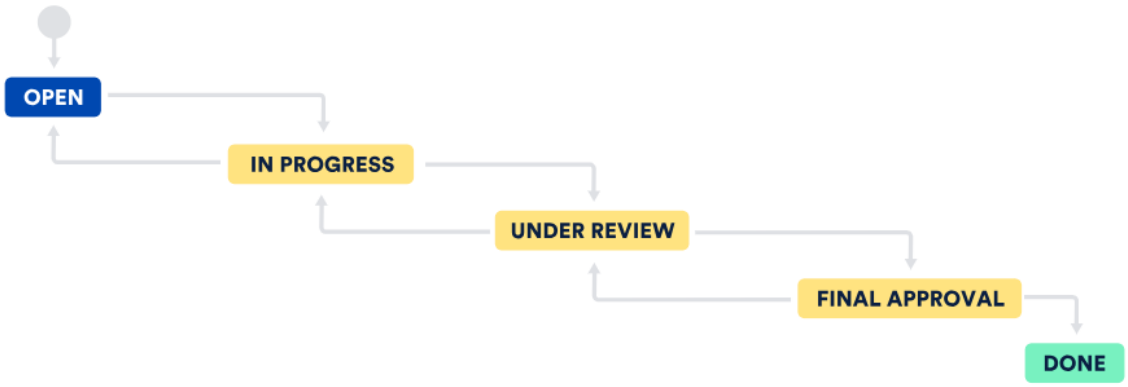


Imagen 7.10 Flujo de estados de un ticket Jira

Open: El usuario realiza el requerimiento que necesita en un nuevo ticket de soporte.

En progreso: Involucra el análisis de impacto, diseño de solución y desarrollo de la actividad, estas tareas son realizadas por el desarrollador.

En revisión: La gerencia de QA realiza las validaciones pertinentes para asegurar la correcta implementación del requerimiento.

Aprobación: Se presenta el resultado al usuario y se obtiene su validación de que la funcionalidad requerida se ha cubierto.

Done: Se realiza la liberación de cambios al ambiente productivo de la forma descrita anteriormente y finaliza la solicitud.

De esta forma se tiene una integración continua de cambios y mejoras al producto sin perder la calidad y robustez de éste con las cuales fue construido. Adicionalmente se brinda un producto que ha sido probado por diferentes profesionales haciendo que se pueda ofrecer un producto de calidad y a la altura de las necesidades del negocio.

En este punto se cierra el ciclo de desarrollo de software para la empresa (o al menos las fases en las que he tenido y sigo teniendo participación directa en el proyecto) con lo cual se da por terminado el presente informe de actividades profesionales.

Conclusiones

Haber tenido la oportunidad de estudiar la carrera de Ingeniería en Computación en la Facultad de Ingeniería ha sido fundamental para mi crecimiento personal y profesional, ya que he logrado adquirir conocimientos sólidos que me permiten afrontar proyectos laborales cada vez más grandes y ofrecer soluciones tecnológicas completas.

Es importante destacar que la mayoría de las tecnologías que se utilizaron en este proyecto no son enseñadas dentro de la facultad en alguna asignatura en específico, pero gracias a los conocimientos de la facultad y siendo autodidacta es posible adquirir todo el conocimiento que se desee.

La realización del proyecto presentado en este documento me ha permitido tener un panorama completo de lo que implica realizar un desarrollo de software de inicio a fin. He podido entender que la programación es solamente una actividad más dentro del desarrollo de software, así como el diseño, las pruebas, el análisis de requerimientos y la administración de tiempos y costos. Entendí que todas estas habilidades son igual de importantes y necesarias para poder ofrecer un software de calidad que se apegue a las necesidades de los clientes.

Durante la formación académica suele darse mayor peso a las habilidades técnicas, sin embargo, al realizar este proyecto he podido entender la importancia de una correcta planeación, el manejo de los recursos y sobre todo de una buena comunicación con los involucrados en un proyecto profesional, ya que esto ayuda a entregar soluciones que se ajusten a las necesidades del cliente.

En este informe he descrito la aplicación de los conocimientos adquiridos en mi formación académica en la facultad y he podido medir el progreso que he tenido como profesionista y como desarrollador de software desde que concluí mis estudios. He adquirido nuevas habilidades y he mejorado mucho las existentes, fruto del esfuerzo, compromiso, dedicación y sobre todo de la pasión por todas las actividades que abarca la Carrera de Ingeniería en Computación.

Bibliografía

Salvador Alonso Sánchez, Miguel Sicilia Urban, & Daniel Rodríguez Gárcia. (2012). Ingeniería del software - un enfoque desde la guía swebok. México: Alfaomega. Recuperado en 2018

Erich Gamma. Design Patterns Elements of Reusable Object-Oriented Software (37 edición). Recuperado en 2018

Ian Sommerville. (2005). Ingeniería del software. España:Pearson Educación. Recuperado en 2018

Philippe Kruchten. (1995). Architectural Blueprints – The “4+1” View Model of Software Architecture. IEEE Software. Recuperado en 2018

Microsoft Azure: plataforma y servicios de informática en la nube. Recuperado en 2018 de https://azure.microsoft.com/es-es/

ASP.NET MVC 5. (2019). Recuperado en 2019 de https://docs.microsoft.com/en-us/aspnet/mvc/mvc5

Matos, A., Verma, R., & Masne, S. (2019). What is REST – Learn to create timeless RESTful APIs. Recuperado en 2019 de https://restfulapi.net/

Flux Financiera, Factoraje. Recuperado en 2018 de from http://fluxfinanciera.com/factoraje/

Alto Nivel. (2017). Qué es el factoraje y por qué es fuente de financiamiento para PyMEs. Recuperado en 2018 de https://www.altonivel.com.mx/empresas/que-es-el-factoraje-y-por-que-es-fuente-de-financiamiento-para-pymes/

Tomás Álvarez Meli. (2017). Flux Financiera: Convirtiéndose en el líder de factoraje a base de transformación digital. MiFiel. Recuperado en 2018 de http://blog.mifiel.com/factoraje-transformacion-digital/

Alto Nivel. (2017). Flux Financiera: los Rolling Stones del factoraje en México. Recuperado en 2018 de https://www.altonivel.com.mx/empresas/flux-financiera-los-rolling-stones-del-factoraje-en-mexico/

RedHat, El concepto de cloud computing. (2018). Recuperado en 2018 de https://www.redhat.com/es/topics/cloud

Glosario Financiero. (2019). Disponible en https://graficos.elfinanciero.com.mx/2014/glosario-financiero/

Comisión Nacional Bancaria y de Valores, Glosario. (2019). Disponible en https://portafolioinfo.cnbv.gob.mx/Paginas/Glosario.aspx

Glosario

API: Una interfaz de programa de aplicación (API) es un código que permite que dos programas de software se comuniquen entre sí. Lo hacen exponiendo al resto de aplicaciones el conjunto de servicios disponibles en cada una y cómo se deben acceder.

Buró de Crédito: El buró de crédito es una empresa privada, independiente de las instituciones financieras, de las comerciales y de las gubernamentales, que tiene como fin concentrar y proporcionar a sus empresas afiliadas, la información referente al comportamiento que han tenido las personas físicas y morales con respecto a sus créditos.

CIEC: La Clave de Identificación Electrónica Confidencial, o CIEC es un mecanismo de acceso, formado por el número de RFC y una contraseña elegida por el contribuyente, que se utiliza para el ingreso a diversas aplicaciones y servicios del SAT a través de su Portal de Internet.

CSS: Hojas de estilo en cascadas (Cascading Style Sheet). Un término usado para describir un método para separar el diseño web. También como un tipo de archivo utilizado para el diseño del estilo de la página web.

Deadline: Un deadline hace referencia a la fecha de entrega total o parcial de un proyecto o servicio. Esta fecha suele fijarse tras una negociación entre dos partes: la persona (o el grupo) que realiza el proyecto y el receptor de ese proyecto o cliente

Factura: Una factura es un documento de tipo mercantil que sirve para recopilar toda la información relacionada con la compra y la venta de un producto o servicio.

Framework: Es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que pueden servir de base para la organización y desarrollo de software.

HTML: Es un lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet. Se trata de la sigla que corresponde a HyperText Markup Language. HTML se encarga de desarrollar una descripción sobre los contenidos que aparecen como textos y sobre su estructura, complementando dicho texto con diversos objetos.

Requerimientos Funcionales: Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer.

Requerimientos No Funcionales: Son restricciones de los servicios o funciones ofrecidos por el sistema. Hacen referencia a las propiedades emergentes del sistema como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema.

SAT: El Servicio de Administración Tributaria (SAT) es un órgano desconcentrado de la Secretaría de Hacienda y Crédito Público, que tiene la responsabilidad de aplicar la legislación fiscal y aduanera, con el fin de que las personas físicas y morales contribuyan proporcional y equitativamente al gasto público; de fiscalizar a los contribuyentes para que cumplan con las disposiciones tributarias y aduaneras; de facilitar e incentivar el cumplimiento voluntario, y de generar y proporcionar la información necesaria para el diseño y la evaluación de la política tributaria.

Servicio Web: Sistema de software designado para dar soporte a la interacción de máquina a máquina interoperativa a través de una red. Un servicio web realiza una tarea específica o un conjunto de tareas y se describe mediante una descripción de servicio en una notación XML estándar llamada WSDL (Web Services Description Language). La descripción de servicio proporciona todos los detalles necesarios para interactuar con el servicio, incluidos los formatos de mensaje (que detallan las operaciones), los protocolos de transporte y la ubicación.

SQL: Structured Query Language es un lenguaje de programación estándar e interactivo para la obtención de información desde una base de datos y para actualizarla.

W3C: El Consorcio World Wide Web (W3C) es un consorcio internacional donde las organizaciones miembros, personal a tiempo completo y el público en general, trabajan conjuntamente para desarrollar estándares Web. La misión del W3C es guiar la Web hacia su máximo potencial a través del desarrollo de protocolos y pautas que aseguren el crecimiento futuro de la Web.