Universidad Nacional de Ingeniería

Facultad de Electrotecnia y Computación

TRABAJO MONOGRÁFICO

Implementación del Sistema web de Apoyo al Área de Ventas para la empresa Casa Cross

PRESENTADO POR:

Hubert Moisés Cross Latino

PARA OPTAR AL TÍTULO DE:

Ingeniero en Computación

TUTORA:

MSc. Lizzette Carolina Duarte Mora

Managua, Nicaragua

Octubre, 2017

DEDICATORIA

不怕慢

就怕站

No hay que temer ir despacio, sino detenerse.

FOR EMMA

Your dad finished college and so will you.

Tu papá terminó la universidad, como harás vos también.

A MI TUTORA

Quien siempre tuvo la bondad de guiarme y apoyarme, lo cual jamás olvidaré.

RESUMEN DEL TEMA

El presente estudio documenta la implementación de un sistema con el nombre de “Sistema Web de Apoyo a Ventas” o SAV. Esto se presenta para cumplir con el requisito de culminación de estudios de la carrera de Ingeniería en Computación.

El acceso a la información en tiempo y forma es un asunto crítico para cualquier organización comercial. La falta del mismo podría perjudicar la propia sobrevivencia de una empresa. Tal es el caso del área de ventas de repuestos automotrices de Casa Cross en el cual se ha sufrido un costo de oportunidad que ha impedido el alcance de su total potencial. Sus actividades se han visto afectadas en desempeño, tiempo y calidad.

Es por estas razones que se realizó este trabajo monográfico que expone la información correspondiente al análisis, diseño e implementación de la solución en base a las necesidades y requerimientos obtenidos vía estrechas interacciones con los interesados en la empresa. Entre las diversas actividades que formaron parte del análisis y diseño se encuentran el análisis de los requerimientos, la elaboración de diagramas UML y la elección de las tecnologías de software y hardware para hacer posible la eventual implementación del sistema.

El resultado final de lo antes mencionado fue un software que, por apoyar al equipo de ventas en sus actividades diarias de brindar atención a los clientes de mayor volumen y por tanto importancia, agregó y continua agregando un valor significativo y consistente a la organización. Además se estableció una base de software moderna que le pertenecerá a la empresa por siempre y sobre cual la gerencia podría bien elegir agregar aún más funcionalidad, yendo desplazando las tecnologías actualmente en uso que se encuentran en vías a la obsolescencia.

Índice de Contenido

Índice de tablas y figuras

1. Introducción

La industria de comercialización de refacciones automotrices presenta un sin número de retos gerenciales y logísticos. Entre estos se encuentra el manejo apropiado de los datos y la optimización de los procesos de negocio que dependen de ello. Darle la debida atención a este aspecto operativo puede conseguir beneficios significativos para las empresas, tales como control más efectivo, clientes más satisfechos y mayores utilidades. Es importante tomar en cuenta esta realidad a la hora de identificar y definir los procesos de negocio y los mecanismos informáticos que los regularán.

Casa Cross no es ninguna excepción a esta realidad de negocio. Por tanto, se mantiene en constante depuración el ámbito informático y su dimensión estratégica en cuanto al giro del comercio.

El área de ventas es de suma importancia ya que afecta directamente el resultado financiero de la empresa. Vendedores que trabajan fuera de las instalaciones deben poder contar con el apoyo de una herramienta web que les permita ejercer su función de atender a los clientes de manera móvil e independiente. Lamentablemente el interfaz web actual (también conocido como “Cotizador Web”) no está cumpliendo con las necesidades de negocio tales como el apoyo en consultar existencias detalladas de productos y registrar cotizaciones en la base de datos del sistema ERP (Siglas en inglés de Planificación de Recursos Empresariales) empresarial. Por estas razones se propone la implantación de un software web más sofisticado que reemplace la solución actual.[1]

[1] En Enero del 2017 el servidor en el cual residía dicho software caducó por falla de hardware y desde entonces la fuerza de ventas no quedó sin ninguna solución informática.

2. Antecedentes

Casa Cross (razón social “Maquinaria H.F. Cross”) fue fundada en 1946 por los hermanos Hubert y Wilfrid Cross Urcuyo con la misión de importar y distribuir bicicletas inglesas marca *Raleigh*. Pocos años después se empezaron a importar y distribuir vehículos ingleses de marca *Land Rover* y luego la marca japonesa *Mistubishi* en 1968. En los años 80 se sufrieron efectos adversos al giro del negocio y la empresa entró a una etapa de mantenimiento de las líneas de productos. En los 90 se dividió la empresa entre Minicar que se dedicaría a comercializar *Mistubishi* y Casa Cross con repuestos automotrices y luego vehículos *Chevrolet* y *Peugeot*. (Entrevista no estructurada con el hijo de uno de los fundadores, Hubert Cross Cooper)

Hoy la empresa tiene 4 sucursales en Managua y 3 sucursales en los departamentos. Distribuye plantas eléctricas, motores marinos, motocicletas y vehículos. También cuenta con un taller de reparación vehicular que ofrece una gama de servicios tanto al cliente particular como a flotas empresariales, más sobre todo el enfoque del negocio sigue siendo la comercialización de repuestos automotrices.

Debido al fuerte crecimiento del negocio a lo largo del tiempo, han surgido retos de seguir el paso de los avances de la tecnología y las mejores prácticas informáticas. Los negocios deben exigir flexibilidad en sus sistemas para mantener un alto grado de competitividad en el mercado. Fue con esta noción que se elaboró en 2013 el primer interfaz web en la empresa que permitiría acceder a la información y manipular datos en las bases de datos que previamente solo eran alcanzables a través del sistema *desktop* convencional SCM (“Sistema de Control Máximo”), el sistema ERP de la empresa y herramienta desarrollada en Visual Basic 6.0 antes de que los dispositivos móviles fueran tan comunes como hoy en día lo son.

Antes de que se implantara el interfaz web (también conocido como “Cotizador Web”), el procedimiento para que los vendedores operando fuera de la empresa pudieran atender a sus clientes consistía en que ellos llamaran a los facturadores de ruta para que por teléfono pudieran consultar los precios y existencias de los productos y estados de cuenta de los clientes. El facturador utilizaba el modulo de cotizaciones del sistema informático SCM para realizar las consultas y comunicárselas al vendedor.

El antecedente directo al propuesto sistema es el mencionado Cotizador Web, herramienta que por años fue la única opción para acceder a la información desde fuera de la empresa a través de internet y sin la aplicación cliente de SCM. Sin embargo, desde su puesta en marcha el Cotizador Web tuvo un desempeño insatisfactorio. El código fuente no se depuró bien y carecía totalmente de comentarios. Además no se cuenta con la versión más reciente del código fuente sino una versión anterior a la que se encuentra en producción hoy en la empresa. Esto se volvió el factor determinante en cuanto a la imposibilidad de que siguiera en uso. Además fue siempre lento, inestable, inseguro (corre bajo IIS y ASP.NET) y nunca fue completado.

El servidor en el cual residía el Cotizador Web tenía una configuración sumamente débil en cuanto a la seguridad de red. El demonio (*daemon*) que gestionaba el HTTP era Microsoft IIS (*Internet Information Services*), servidor web que se destaca por la cantidad de vulnerabilidades que ha presentado a lo largo de su historia.(Microsoft IIS version 7.0: Security vulnerabilities) A pesar de esto el equipo físico contaba tanto con un interfaz en el VLAN interno de la empresa que con otro con dirección IP publico estático, efectivamente proporcionando una potencial “puerta trasera” a la red empresarial.

Para mitigar el innegable riesgo de seguridad que esta situación presentaba, en Septiembre 2016 se creó un servicio OpenVPN (Virtual Private Network) a la vez que se deshabilitó el acceso externo directo al servidor, dejando este solamente con una dirección IP interna privada. Los usuarios tendrían que primero que nada utilizar un cliente OpenVPN desde su dispositivo para autenticarse y así

poder acceder exclusivamente al IP interno del servidor web IIS. Así se logró parchear el tema de la seguridad sin negarles acceso a los usuarios críticos, a cuestas de complicar el acceso y la gestión de las cuentas necesarias para utilizar el sistema.

En base a los estudios del código fuente disponible, la lentitud se atribuye al diseño ineficiente de la interacción entre el cliente y el servidor. Por ejemplo, en el caso específico de preparar datos para registrar una cotización, el sistema anterior envía 13 consultas distintas al DBMS (siglas inglesas de Sistema Gestor de Base de Datos) una tras otra, cada vez abriendo y cerrando una conexión. Luego se realiza un llamado a un procedimiento almacenado por cada producto presente en la cotización a través en un ciclo. Este tipo de algoritmo tiene un costo de rendimiento significativo, particularmente en dispositivos móviles que dependen de conexiones de datos lentas.

Por último la inestabilidad terminó volviéndose total cuando el equipo en cual residía el sistema dio la queda por completo en enero del 2017, lo que por casualidad sucedió durante la fase de redacción del protocolo del presente trabajo.

3. Justificación

Uno de los inconvenientes que existe con el proceso actual de ventas es que el interfaz web a menudo se encuentra en un estado no funcional. Los vendedores se quejan, manifestando que su trabajo y el proceso de ventas se ven averiados al no tener una herramienta apropiada para agilizarlo. Los facturadores también se quejan al tener que atender consultas telefónicas de existencias, precios y más con cada uno de los vendedores. (Entrevista no estructurada al gerente general, Eduardo Padilla)

Para realizar una venta, el vendedor visita al cliente y levanta una orden. Esta orden debe ser ingresada a SCM (“Sistema de Control Máximo”, el ERP interno) en la forma de una Cotización. Una vez registrada la Cotización, el facturador de ruta revisa las ubicaciones de los productos, solicitando confirmaciones de existencias entre las distintas bodegas donde se podrían encontrar los artículos que pasan a una “Cotización Con Solicitud”. El personal de centro de almacenamiento (“CA”) revisa la existencia física de los productos y confirma o niega las Solicitudes en el caso de que los artículos no se pueden confirmar.

Cuando el Cotizador Web se caía o se encontraba “inusablemente lento” (Entrevista no estructurada al vendedor Franklin Guillén), los vendedores tenían que consultar existencias y registrar precios por teléfono con el apoyo de los facturadores de ruta. Esta es una interacción en la cual ambas partes pierden tiempo productivo. Esto afecta de manera negativa la habilidad de los vendedores alcanzar su máximo potencial y por tanto se puede decir que provoca un impacto negativo tanto en las ventas como en la atención y servicio que reciben los clientes.

Un ejemplo concreto de la problemática que permite apreciar la importancia de resolverla es el caso de Julio del 2016, cuando el cotizador estuvo presentando fallas durante dos semanas de ese mes. En la tabla 1 se puede observar una disminución en el valor total de las cotizaciones de 8%, y de cotizaciones web de

23%. La discrepancia se podría deber a que las cotizaciones que se hicieron durante la caída fueron realizadas por teléfono, caso cuya desventajas ya han sido destacadas.

Tabla 1 - Efecto de la caida del Cotizador Web en Julio 2016

MES	COTIZACIONES	CAMBIO VS MES ANTERIOR	COTIZACIONES WEB	CAMBIO VS MES ANTERIOR
2016-JUN	72116182.9	-	1976066	-
2016-JUL	66285654.93	-8%	1511677	-23%
2016-AUG	74910680.77	+13%	1939090	+28%

En resumen:

El vendedor no cuenta con un mecanismo para registrar cotizaciones en la BD ("Base de Datos") del sistema empresarial: Pierde tiempo consultando por teléfono cosas que debería poder consultar de directamente por internet a través de un interfaz web apropiado.

El facturador de ruta debe atrasarse en sus funciones para atender al vendedor cuando le llama a consultar productos, saldos, etc. Esto importuna su trabajo de atender las cotizaciones conforme vienen siendo registradas.

Tomando en cuenta estos inconvenientes, se considera justificable la implementación de un sistema web que cumpla las mismas funciones que el interfaz anterior utilizando plataformas modernas para el beneficio de la empresa más otros requerimientos que se agreguen durante las fases correspondientes del proyecto.

4. Objetivos

4.1 Objetivo General

1. Implementar el Sistema Web de Apoyo a Ventas para agilizar el proceso de ventas de Casa Cross

4.2 Objetivos Específicos

1. Diagnosticar el interfaz [2] web de registro de cotizaciones actual [3] y su interacción con la base de datos del sistema informático empresarial

2. Elaborar un modelado de análisis basado en las necesidades del negocio y aquellas características que se agreguen de las peticiones de los interesados en el proceso

3. Modelar un diseño basado en los hallazgos del análisis que defina la arquitectura, el interfaz gráfico y componentes del sistema propuesto, utilizando herramientas UML (Lenguaje Unificado de Modelado)

4. Implementar incrementos funcionales a lo largo del avance del proyecto hasta la entrega final

5. Realizar pruebas de funcionalidad a la aplicación web para verificar su correcta implementación antes de cada liberación de incrementos

6. Capacitar a los usuarios finales con demostraciones presenciales y elaboración de videos manuales

7. Poner en marcha el sistema completado

[2] Cabe mencionar que se refiere al software con el apodo “Cotizador Web” como interfaz y no sistema. Para la definición de sistema véase el acápite 5.1

[3] Afortunadamente se aprovechó la oportunidad de realizar esta actividad antes de que el interfaz anterior diera la queda

5. Marco Teórico

El presente acápite plasma en forma explícita criterios teóricos y bases conceptuales en los cuales se basa el proyecto para el abordaje del problema. Se presentan los términos relacionados al área de desempeño así como los conceptos relacionados a las técnicas, estrategias y metodologías de la ingeniería del software y administración de operaciones. Estos nos permitirán diseñar un sistema con mayores niveles de calidad y aplicar buenas prácticas de ingeniería de software a lo largo del desarrollo del proyecto.

En cuanto al ordenamiento, comenzamos señalando los criterios de carácter más general, luego particularizando hacia los criterios que tienen una orientación más específica. Se hace referencia a los aspectos teóricos más importantes a utilizarse en el proyecto.(FIQ, 2010)

5.1 Sistema Informático

El sistema informático o sistema de información es un conjunto integrado de componentes para la recolección, almacenamiento y procesamiento de datos y para la entrega de información, conocimiento y productos digitales. Empresas comerciales y otras organizaciones dependen de los sistemas informáticos para ejercer y gestionar sus operaciones, interactuar con clientes y suplidores y competir en el mercado.(Zwass, 2011)

Los componentes principales de los sistemas de información son software y hardware de ordenador, telecomunicaciones, bases de datos y almacenes de datos, recursos humanos y procedimientos. El hardware, software y telecomunicaciones constituyen tecnología de la información (TI), cual está actualmente integrada en las operaciones y la gestión de las organizaciones.(Zwass, 2011)

5.1.1 Software Heredado ("*legacy*")

Los sistemas de software heredado o "legacy" fueron desarrollados hace décadas y han sido modificados en forma continua para cumplir los requerimientos de los cambios en los negocios en las plataformas de cómputo. La proliferación de dichos sistemas ha causado dolores de cabeza a las grandes organizaciones, las cuales los perciben como costosos en su mantenimiento y riesgosos en su evolución.(Dayani-Fard, 1999)

Lamentablemente, existe una característica adicional que a menudo está presente en el software legacy: la baja calidad. Algunas veces, los sistemas legacy tienen diseños imposibles de extender, código intrincado, documentación escasa o inexistente, casos de prueba y resultados que nunca fueron archivados, un historial de cambio mal manejado, etc. No obstante, estos sistemas son el soporte de "las funciones centrales de negocios y son indispensables para las empresas". ¿Qué se puede hacer?

Muchas veces la respuesta más razonable es no hacer nada. Mientras el sistema legacy satisfaga las necesidades de los usuarios y funcione de manera confiable, el sistema no está roto y no requiere arreglos. Sin embargo, conforme pasa el tiempo, los sistemas legacy evolucionan por una o más de las razones siguientes:

- El software debe adaptarse para satisfacer las necesidades de los nuevos ambientes o las nuevas tecnológicas de cómputo
- El software debe mejorarse para implementar los nuevos requerimientos de los negocios
- El software debe extenderse para hacerlo operable con sistemas y bases de datos más modernos
- El software debe rediseñarse para hacerlo viable dentro de un amiente de red(Pressman, Ingeniería del Software, 6ta ed, 2005)

Cuando suceden estas formas de evolución en un software legacy, éste debe someterse a una reingeniería de modo que conserve su viabilidad en el futuro. La meta de la ingeniería de software moderna es “imaginar metodologías que se basen en la noción de la evolución”; esto es, la noción de que “los sistemas de software cambian de manera continua, los nuevos sistemas de software se construyen a partir de los viejos, y… todos deben interactuar y cooperar con los demás”.(Dayani-Fard, 1999)

5.1.2 Base de Datos

Una base de datos es una colección de datos o información organizada específicamente para acceso y búsqueda rápida. Las bases de datos son estructuradas para facilitar el almacenamiento, acceso, modificación y eliminación de datos en conjunto con varias operaciones de procesamiento de datos. El sistema gestor de base de datos (DBMS en inglés) extrae información de la base de datos en respuesta a consultas.(Encyclopædia Britannica, 2016)

SQL – Structured Query Language. Lenguaje de Consulta Estructurada. SQL es un lenguaje de programación de propósito especial para gestionar datos registrados en un Sistema Gestor de Base de Datos Relacional (RDBMS – Relational Database Management System). SQL consiste en tres sub-lenguajes: DDL (Data Definition Language), DML (Data Manipulation Language) y DCL (Data Control Language)

La óptica de SQL incluye la inserción, consulta, actualización e eliminación de datos. (Insert, Query, Update, Delete). Además incluye la creación y modificación de esquemas y el control de acceso a datos. SQL es definido por el estándar ANSI/ISO SQL que ha venido evolucionando desde el año 1986. El estándar actual es ISO/IEC 9075-1:2011.

Modelo Relacional (Relational Database Model)

El modelo relacional (RM o Relational Model) fue inventado en 1969 por Edgar F. Codd en el Laboratorio de Investigación San José, California de IBM (San José Research Laboratory). Codd concluyó que los modelos de uso común (graph, network models) presentaban problemas e inconvenientes que se podían superar con otro modelo. Según Codd, el MR provee “una base firme para tratar la derivabilidad, redundancia y consistencia de las relaciones”.(Codd, 1970, pág. 377)

La base de datos relacional registra los datos en tablas separadas en lugar de almacenarlos todos en una sola unidad lógica. Las estructuras de las BD son organizadas en archivos físicos optimizados para mayor velocidad. El modelo lógico, que cuenta con objetos tales como bases de datos, tablas, vistas, filas y columnas ofrece un entorno de programación flexible. Se establecen reglas que gobiernan las relaciones entre campos de datos distintos tales como uno a uno, uno a mucho, unicidad, mandatorio u opcional y punteros (“pointers”) entre tablas distintas. La base de datos se encarga de ver que estas reglas se cumplan de tal manera que con una base de datos bien diseñada, la aplicación nunca presenta datos inconsistentes, duplicados, huérfanos, desactualizados o faltantes. (Oracle Corporation, 2016, pág. 5)

RDBMS – un RDBMS es un DBMS que utiliza el modelo relacional. En el caso de este proyecto el RDBMS del sistema heredado es Microsoft SQL Server. La estructura general de un RDBMS se ilustra en la Figura 1.

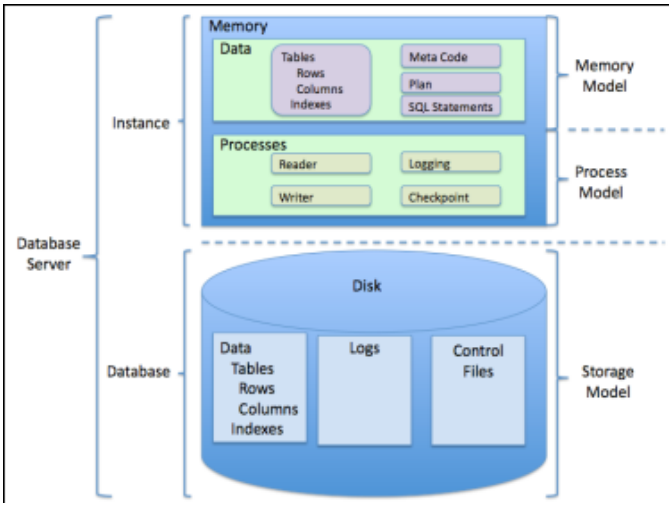


Figura 1 - Modelo RDBMS

5.2 Modelo cliente-servidor

Un servidor es una computadora que lleva a cabo un servicio que normalmente requiere mucha potencia de procesamiento. Un cliente es una computadora que solicita los servicios que proporciona uno o más servidores y que también lleva a cabo algún tipo de procesamiento por sí mismo.

La computación [sic] cliente/servidor es un intento de equilibrar el proceso a una red hasta que se comparta la potencia de procesamiento entre computadoras que llevan a cabo servicios especializados tales como acceder a bases de datos (servidores), y aquellos que llevan a cabo tareas tales como la visualización IGU (interfaz gráfico usuario) que es más adecuado para el punto final dentro de la red. Por ejemplo, permite que las computadoras se ajusten a tareas especializadas tales como el procesamiento de bases de datos en donde se utilizan hardware y software de propósito especial para proporcionar un procesamiento rápido de la base de datos comparado con el hardware que se encuentra en las *mainframes* que tienen que enfrentarse con una gran gama de aplicaciones.(Pressman, Ingeniería del Software; Un Enfoque Práctico V ed, 2002, pág. 492)

5.3 Open Source

Open Source o Código Abierto es un término que se aplica al software distribuido bajo una licencia que le permite al usuario acceso al código fuente del software, y además le permite estudiar y modificarlo con toda libertad, sin restricciones en el uso del mismo; y además le permita redistribuirlo, siempre y cuando sea de acuerdo con las condiciones de la licencia bajo la cual el software original fue adquirido.(Isocron, 2016, pág. 1)

5.4 Aplicaciones y la Ingeniería Web

Una aplicación web puede considerarse un sistema web (servidor web, red, HTTP y navegador) en cual la entrada del usuario (navegación e ingreso de datos) afecta el estado del negocio (Conallen, 1999, pp. 63-70). A pesar de la aparente facilidad con cual páginas HTML son creadas, el desarrollo exitoso de aplicaciones web de grande escala es una actividad compleja que requiere métodos y herramientas apropiadas(A. Ginige, 2001, pp. 22-25). Debido a que el desarrollo de estas aplicaciones es una tarea compleja, el soporte de modelación es esencial para proporcionar una vista abstracta de la aplicación. La modelación puede apoyar a los diseñadores durante las fases de diseño a través de la definición formal de los requerimientos, el brindar de detalles de múltiples niveles tanto como brindar soporte para el avance de pruebas antes de la implementación(M. Winckler, 2003, pp. 61-76).

Los sistemas y aplicaciones basados en la Web (WebApps) ofrecen un complejo arreglo de contenido y funcionalidad a una amplia población de usuarios finales. La ingeniería web (IWeb) es el proceso con el que se crean WebApps de alta calidad. La IWeb no es un clon perfecto de la ingeniería del software, pero toma prestados muchos conceptos y principios fundamentales de ella. Además, el proceso IWeb acentúa actividades técnicas y administrativas similares. Existen sutiles diferencias en la manera como se dirigen dichas actividades, pero el

método primordial dicta un enfoque disciplinado para el desarrollo de un sistema basado en la computadora. (Pressman, Ingeniería del Software, 6ta ed, 2005, pág. 502)

5.4.1 Atributos de sistemas y aplicaciones basados en Web

Lo que siguen son descripciones de los atributos de los sistemas Web de Roger Pressman y la contextualización a este proyecto. Pressman nos indica que "estos atributos generales se aplican a todas las WebApps, pero con diferentes grados de influencia".

Tabla 2 - Atributos de los sistemas web y sus relevancias al proyecto

Atributo	Influencia en nuestro proyecto
1. Intensidad de red. Una WebApp reside en una red y debe satisfacer las necesidades de una variada comunidad de clientes. Una WebApp puede residir en el Internet (y, en consecuencia, permitir una comunicación abierta). Alternativamente, una aplicación puede colocarse en una Intranet (lo que implementa la comunicación en una organización).	Nuestro sistema residirá en internet, pero solamente será accesible por un conjunto limitado de personas.
Concurrencia. Un gran número de usuarios puede tener acceso a la WebApp al mismo tiempo. En muchos casos los patrones de uso entre los usuarios finales varían enormemente.	Se exige concurrencia en las transacciones con la DBMS.
Carga impredecible. El número de usuarios de la WebApp puede variar en órdenes de magnitud de día a día. El lunes pueden mostrarse 100 usuarios; el martes pueden usar el sistema 10,000.	El número máximo de usuarios concurrente no será de más de 40.
Desempeño. Si un usuario de WebApp debe esperar demasiado (para ingresar, para procesamiento en el lado del servidor, para formateo y despliegue en el lado del cliente) puede decidir irse a cualquier otra parte.	Nuestro sistema no será competidor entre otros. Será un servicio único dentro de la empresa hecho "a la medida".
Disponibilidad. Aunque la expectativa de una disponibilidad del total es poco razonable, los usuarios de las WebApps	Existe toda la intención de que el servicio sea disponible las 24 horas. Sin embargo, en la práctica, el uso normal será

populares con frecuencia demandan acceso sobre una base de "24/7/365". Los usuarios en Australia o Asia pueden demandar acceso durante momentos cuando las tradicionales aplicaciones de software doméstico en Norteamérica pueden estar fuera de línea por mantenimiento.	solamente durante horas laborales en Nicaragua.
Gobernada por los datos. La función primordial de muchas WebApps es usar hipermedia para presentar contenido de texto, gráficos, audio y video al usuario final. Además, por lo general, las WebApps se utilizan para tener acceso a información que existe en bases de datos que originalmente no eran parte integral del ambiente basado en Web (por ejemplo, comercio electrónico o aplicaciones financieras).	Este atributo es altamente relevante para este proyecto. La esencia del sistema es proporcionar un mecanismo accesible desde fuera de la empresa para consultar y registrar datos en la BD.
Evolución continúa. A diferencia del software de aplicación convencional, que evoluciona a lo largo de una serie de planeadas liberaciones espaciadas cronológicamente, las aplicaciones Web evolucionan de manera continua. No es raro que algunas WebApps (específicamente, su contenido) se actualicen sobre una agenda minuto a minuto, o que el contenido sea calculado de manera independiente para cada solicitud.	Se pretende entregar un producto totalmente funcional que cumpla con todos los requisitos que sean establecidos al inicio del proyecto. Sin embargo se entregará el código fuente sumamente documentado con comentarios para que el personal del área de sistemas pueda aumentarlo conforme vayan evolucionando los procesos de la empresa.
Inmediatez. Aunque la inmediatez—la apremiante necesidad de poner software en el mercado rápidamente—es una característica de muchos dominios de aplicación, las WebApps con frecuencia muestran un tiempo para comercializar que puede ser cuestión de unos cuantos días o semanas. Los ingenieros Web deben aplicar métodos de planeación, análisis, diseño, implementación y puesta a prueba que han sido adaptados a los apretados tiempos requeridos para el desarrollo de WebApps.	La inmediatez que se vive con este proyecto se debe sobre todo a la necesidad de agilizar las operaciones del equipo de ventas. El enfoque no es comercializar el software.
Seguridad. Puesto que las WebApps están disponibles mediantes el acceso a la red, es difícil, si no imposible, limitar la	Muy importante. En la versión final del sistema se utilizará HTTPS para mitigar el riesgo de filtración de información a

población de usuarios finales que pueden tener acceso a la aplicación. Con la finalidad de proteger el contenido confidencial y ofrecer modos seguros de transmisión de datos, se deben implementar fuertes medidas de seguridad a lo largo de la infraestructura que sustenta una WebApp dentro de la aplicación misma.	intrusos.
Estética. Una parte innegable de la apariencia de una WebApp es su presentación y la disposición de sus elementos. Cuando una aplicación se diseña para comercializar o vender productos o ideas, la estética puede tener tanto que ver con el éxito con el diseño técnico.	La estética tiene una relevancia de baja a intermedia en nuestro caso. El interfaz debe ser amigable y presentable, pero se toma en cuenta que es un servicio interno y por tanto no se requiere una estética al nivel de un proyecto de mercadeo tal como un sitio web de comercio electrónico.

5.5 Metodología de desarrollo

5.5.1 Ciclo de vida de desarrollo de sistemas

El ciclo de vida de desarrollo de software (SDLC – *Software Development Life Cycle*) es un proceso de creación o alteración de sistemas y los modelados y metodologías que los profesionales utilizan para desarrollar sistemas en los campos de ingeniería de sistemas, sistemas informáticos y software.(WikiBooks) El proceso se presenta de manera gráfica en la Figura 2.

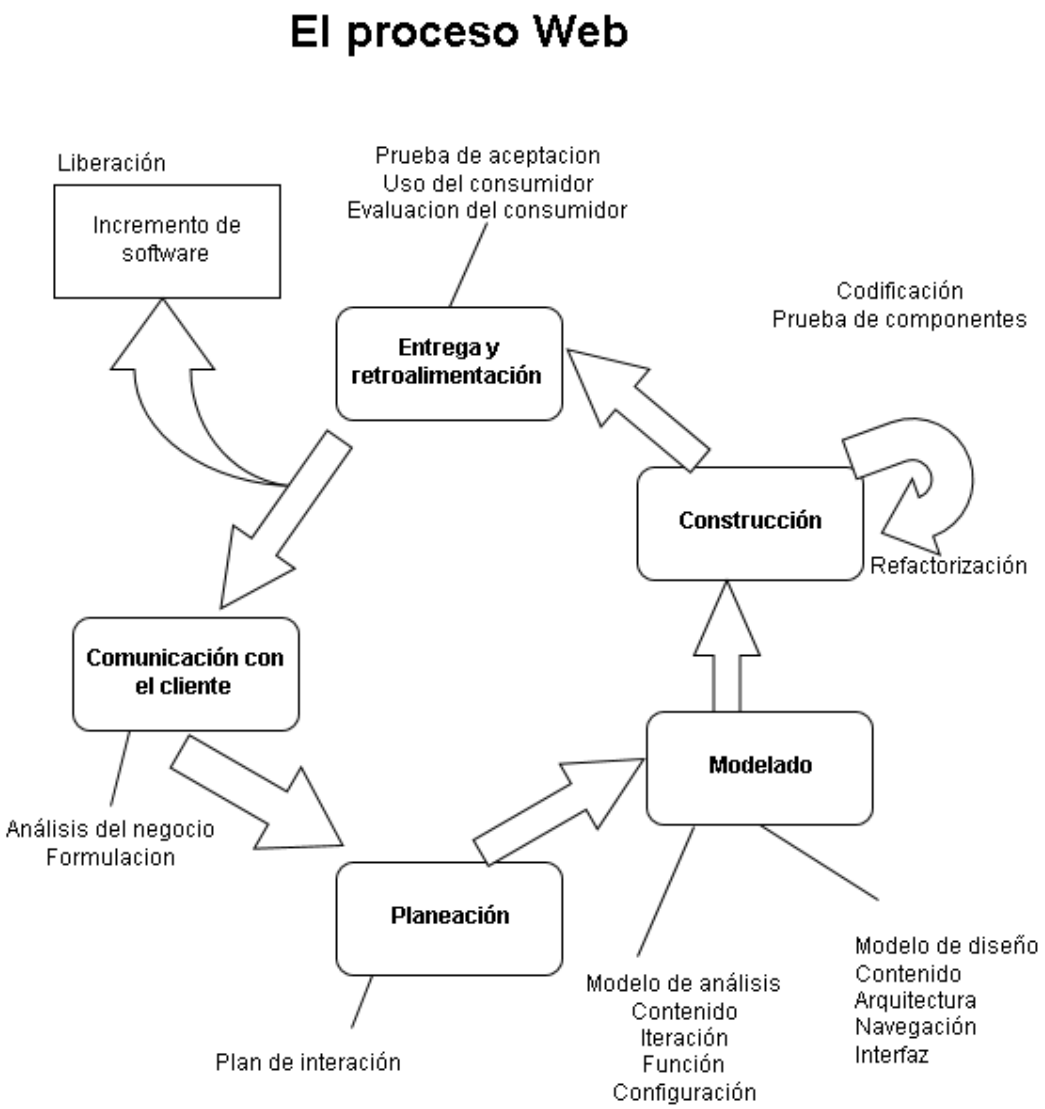


Figura 2 - El proceso de ingeniería web

5.5.2 Modelado de Análisis

El propósito del modelado es reducir la complejidad a través de la construcción de una representación simplificada de la realidad que ignora detalles irrelevantes. Las preguntas cuyas respuestas se buscarán en el modelo determinan la relevancia de los detalles.(Bruegge, 2009)

Los sitios web, por lo general, son enormemente dinámicos. Requieren fases de desarrollo cortas con la finalidad de tener listo el producto y ejecutarlo rápidamente. Con frecuencia, los desarrolladores van directo hacia la fase de codificación sin comprender qué están tratando de construir o cómo quieren construirlo. La codificación respecto del servidor con frecuencia se hace *ad hoc*, las tablas de bases de datos se agregan conforme se necesitan y la arquitectura evoluciona en una forma a veces no intencional. Pero alguna ingeniería de software modelada y disciplinada logrará que el proceso de desarrollo de software sea mucho más suave y asegurará que el sistema web sea más sustentable en lo futuro.(Pressman, Ingeniería del Software, 6ta ed, 2005, pág. 544)

Entregables del modelado de análisis:

- Requisitos No Funcionales (RNF)
- Caso de uso
- Diagrama de secuencia

Entregables del modelado funcional:

- Requisitos Funcionales (RF)
- Diagrama de actividad

Requisitos Funcionales y No Funcionales y su Clasificación.

En la rama de ingeniería de requerimientos, los RF describen lo que un sistema **hace** conforme su diseño. Son enunciados de servicios que el sistema debe proveer, como el sistema debe reaccionar a entradas (*inputs*) particulares y como el sistema debe comportarse in situaciones particulares. Los RNF contrastan con los RF en que en lugar de describir lo que hace un sistema, describen **como** es el sistema conforme su arquitectura.(Chen, Ali Babar, & Nuseibeh, 2013)

Los Requerimientos Funcionales de Usuario (RFU) son aserciones expresadas en lenguaje natural de los servicios que el sistema debe proveer y sus restricciones operativas. Son escritos para el consumidor del sistema. Los Requerimientos Funcionales de Sistema (RFS) conforman documento o acápite estructurado que plasma una descripción detallada de los servicios del sistema. Los RFS son escritos como un acuerdo entre el consumidor y el proveedor del sistema. Los RFS son generales y corresponden a uno o más RFU específicos. Ambos forman parte de la especificación del software ("*software spec*" en inglés) que se define como una descripción detallada de software que puede servir de base para el diseño o la implementación del mismo.

Diagramas UML - Diagramas Caso de Uso

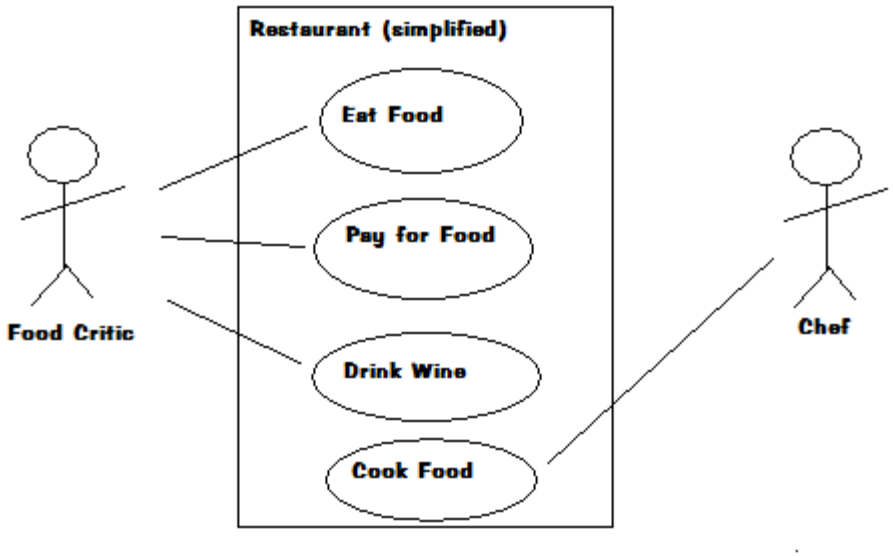


Figura 3 - Ejemplo de diagrama caso de uso

Como se puede observar en la Figura 3, los diagramas de casos de uso sirven para representar, en la manera más sencilla posible, la interacción del usuario con el sistema y las distintas situaciones en cuales el usuario está involucrado. Proporcionan una vista de alto nivel del sistema y comunican los requerimientos de forma fácil de comprender para todos los interesados. En otras palabras, el diagrama caso de uso captura las interacciones que ocurren entre los productores y consumidores de información y del sistema en sí mismo.

Los diagramas de caso de uso representan:

- Se muestran **acciones** que agregan valor al usuario, representándose con elipsis horizontales
- **Actores** tales como una persona, organización o un sistema exterior que asumen algún rol de interacción con el sistema. Se representan con figuras de palitos
- **Asociaciones** entre actores y acciones se indican con líneas sólidas conectando estos. Existe una asociación en cualquier caso que un actor esté involucrado en una interacción descrita en un caso de uso
- Los **límites del sistema** son representados por rectángulos. Son útiles para indicar la entrega de casos de uso por publicación de versiones nuevas(Ambler, 2014)

Diagrama de secuencia (DS)

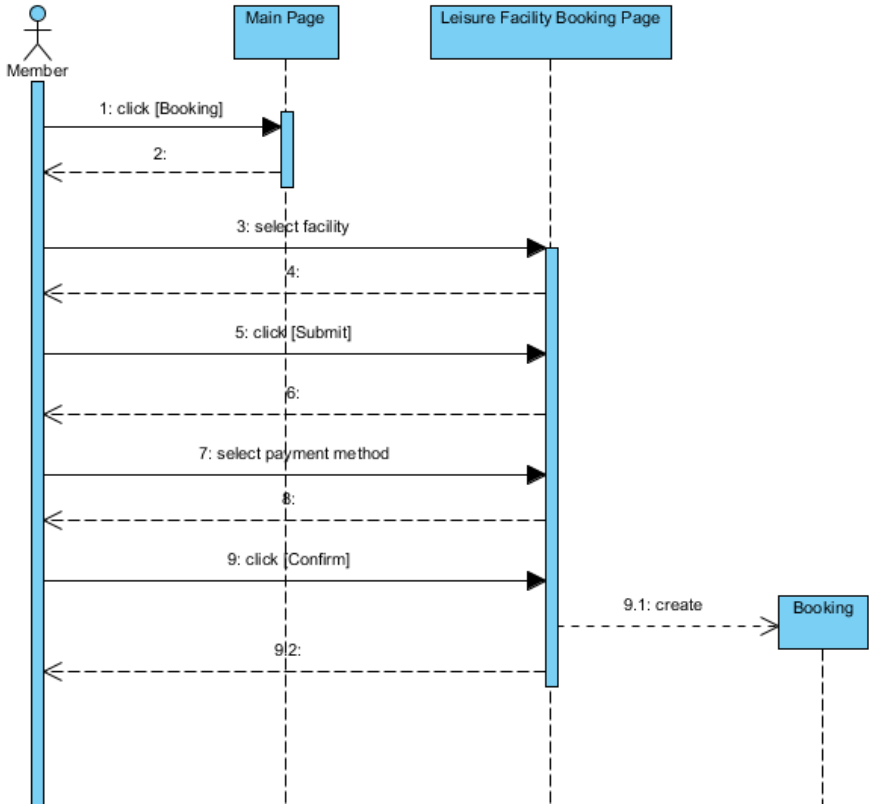


Figura 4 - Diagrama de secuencia ejemplo

Un DS es un diagrama de interacción que muestra cómo operan los objetos unos con los otros y en qué orden lo hacen. Como se puede observar en la Figura 4, el diagrama de secuencia es una representación de comportamiento que indica cómo los eventos causan transiciones de objeto a objeto. Una vez que se han identificado los eventos al examinar un caso de uso, el modelador crear un diagrama de secuencia: una representación de cómo los eventos causan un flujo de un evento a otro como una función del tiempo. En esencia es una versión abrevada del caso de uso. Representa clases clave y eventos que causan que el comportamiento fluya de clase a clase.(Pressman, Ingeniería del Software, 6ta ed, 2005, pág. 238)

Diagrama de actividad

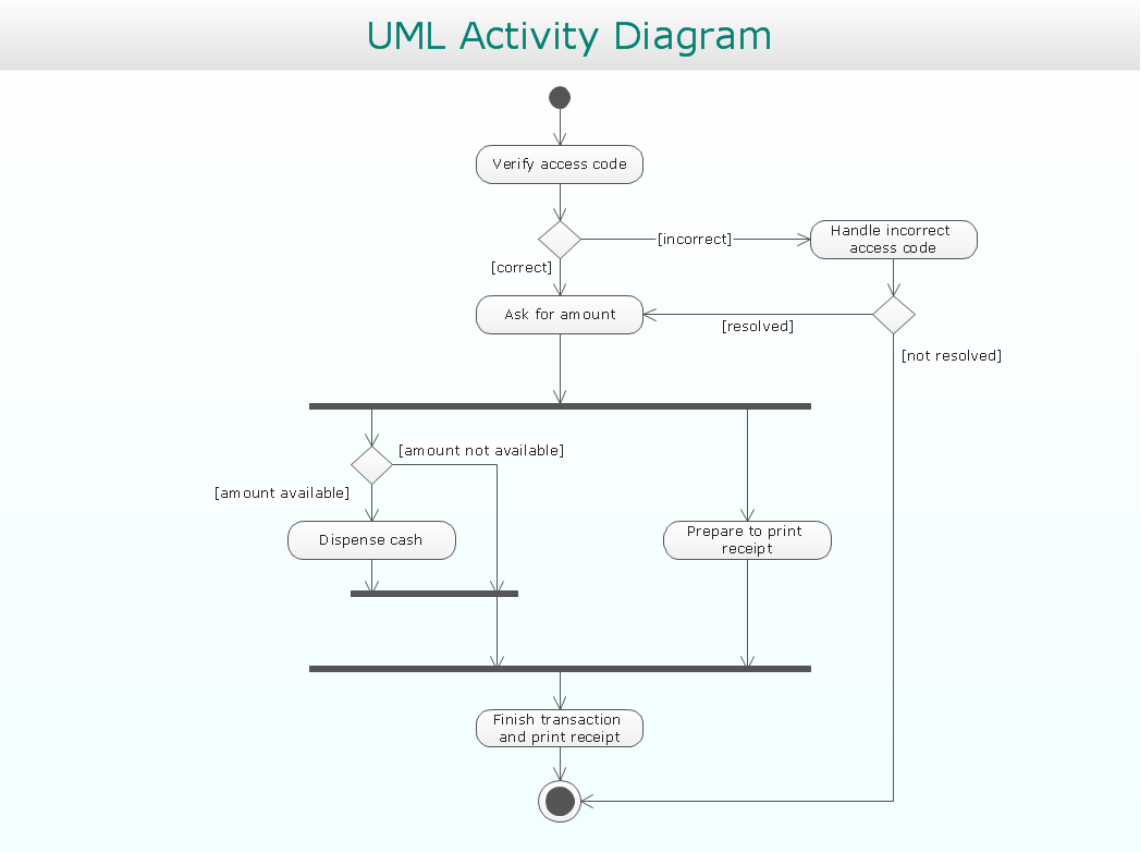


Figura 5 - Diagrama de actividad ejemplo

El diagrama de actividad (véase el ejemplo en Figura 5) sirve para representar los aspectos procedimentales del software. Esta notación UML es similar a aquella del flujograma y se utiliza para representar lo que sucede conforme el sistema realiza sus funciones. Los rectángulos redondeados representan funciones específicas del sistema; las flechas indican el flujo a través del sistema; el diamante representa una decisión bifurcada (cada flecha saliente del diamante se etiqueta); líneas sólidas representan que las actividades ocurren en paralelo. (Pressman, Ingeniería del Software, 6ta ed, 2005)

5.5.3 Modelado de Diseño

El diseño de WebApps abarca actividades técnicas y otras que no lo son. La visión y el sentido del contenido se desarrollan como parte del diseño gráfico; la plantilla estética de la interfaz de usuario se crea como parte del diseño de la interfaz; y la estructura técnica de la WebApp se modela como parte del diseño arquitectónico y de navegación. En toda instancia se debe crear un modelo de diseño antes de que se comience la construcción, pero un buen ingeniero Web reconoce que el diseño evolucionará mientras más se conozcan de los requisitos de los participantes conforme se construya la WebApp.(Pressman, Ingeniería del Software, 6ta ed, 2005, pág. 566) El entregable del proceso de modelado de diseño es el diseño de navegación, un ejemplo del cual se presenta en la Figura 6.

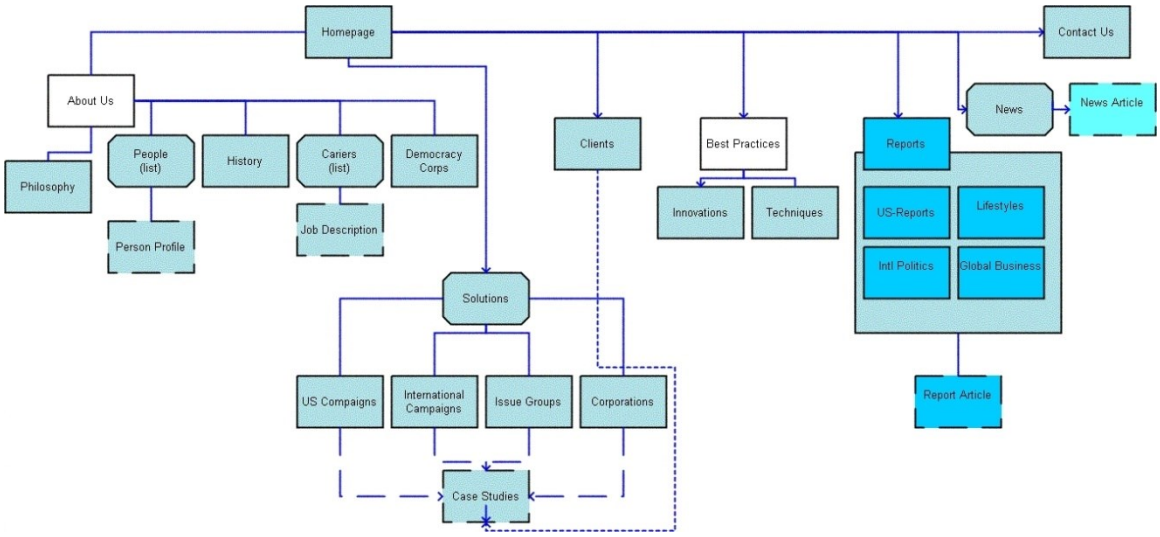


Figura 6 - Arquitectura de navegación (*site map*)

5.5.4 Arquitectura Modelo-Vista-Controlador

La arquitectura MVC es uno de varios modelos de infraestructura WebApp sugeridos para desacoplar la interfaz de usuario de la funcionalidad y el contenido de información de la WebApp. El modelo contiene todo el contenido específico de la aplicación y la lógica de procesamiento, e incluye todos los objetos de contenido, el acceso a fuentes de datos externas y toda la funcionalidad de procesamiento que son específicos de la aplicación. La *vista* contiene todas las funciones especificas de la interfaz y habilitar la presentación del contenido y la lógica de procesamiento, e incluye todos los objetos de contenido, acceso a fuentes de datos/información externas y a toda la funcionalidad de procesamiento requerida por el usuario final. El *controlador* gestiona el acceso al *modelo* y a la *vista* y coordina el flujo de datos entre ellos. En una WebApp, "la vista actualiza el controlador con datos provenientes del modelo con base en la entrada del usuario". (Pressman, Ingeniería del Software, 6ta ed, 2005, pág. 690) Véase la Figura 7 para una representación grafica de la arquitectura MVC.

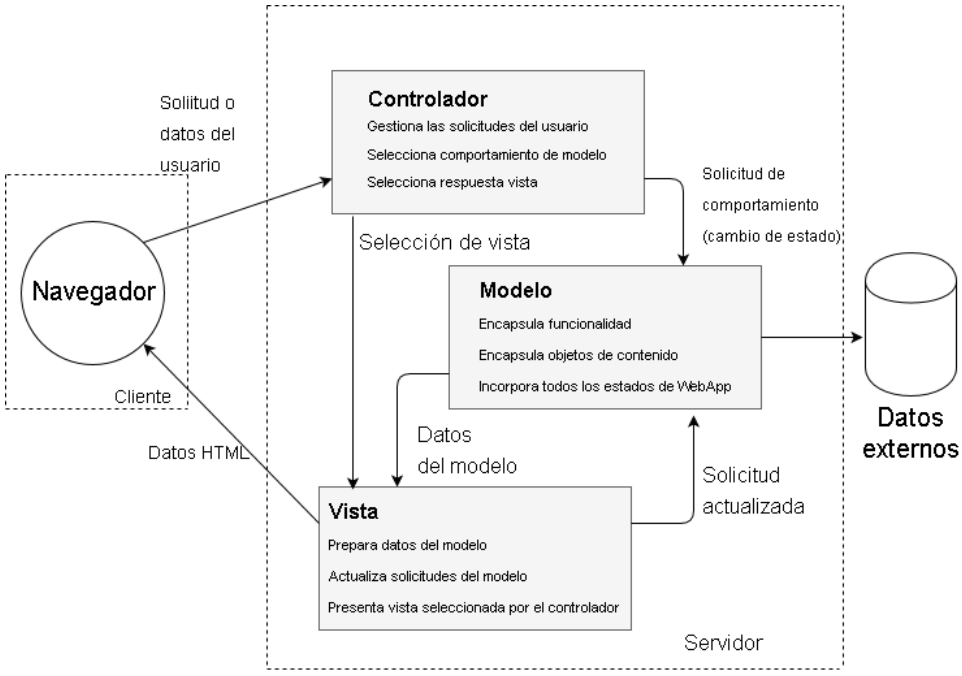


Figura 7 - Arquitectura modelo-vista-controlador

5.5.5 Metodología

El equipo de este proyecto tendrá un solo miembro. Por tanto, metodologías de diseño convencionales que presuponen equipos de gerentes, arquitectos de solución, desarrolladores, probadores etc. deben ser modificadas para el contexto de una persona trabajando de manera individual. También elementos de la programación extrema se pueden aplicar en este caso.

El proyecto se planifica basado en historias de usuario (*User Stories*), utilizando el plan de publicaciones (*Release Plan*) para describir el plan general. El trabajo se planifica a través de iteraciones periódicas (3 semanas por ejemplo) utilizando un plan de iteración (*Iteration Plan*) para establecer lo que se va a hacer. El código se construye utilizando programación en parejas (*Pair Programming*) normalmente, uno adelantando las pruebas unitarias mientras el otro produce código para pasar dichas pruebas. Por último las pruebas de aceptación (*Acceptance Test*) nos indican si hemos completado. (Jeffries, 2005)

Este proceso de desarrollo iterativo nos proporciona los siguientes beneficios:

- Publicación incremental. Para resolver necesidades inmediatas de negocio se liberan versiones con menos características mientras se avanza hacia la versión final del software
- Elasticidad. El enfoque tradicional hacia la programación funciona de manera óptima cuando los requerimientos permanecen estáticos. En la vida real los requerimientos viven cambiando debido a la emergencia de nuevas oportunidades de negocio o sencillamente a que la fase inicial de recolección de requerimientos fue incompleta. La programación extrema acomoda semejantes cambios a través de las historias de usuario al inicio de las iteraciones y la retroalimentación durante las iteraciones.
- Productos robustos. El software se trata como un rompecabezas cuyas piezas van siendo construidas por desarrolladores en iteraciones. La combinación de tales iteraciones al final da un producto completo. El resultado es software funcional con pocos defectos. Esta característica

también permite al cliente tanto como el proveedor suspender el desarrollo en cualquier momento y aún así contar con un código altamente funcional hasta donde alcanzó el desarrollo.(Nayab, 2010)

Sin embargo, la programación extrema tiene sus críticas en el mundo de ingeniería de software. Afortunadamente, el proceso fue diseñado para poderse modificar hacia mejor cumplimiento de las necesidades especificas al proyecto. Entre las reportadas las dos principales son:

- La programación extrema supone y depende de interacción constante con el cliente. Su éxito depende de gran manera en la colección de datos en muchos puntos del proceso de desarrollo. Algunos clientes pueden ser menos disponibles que otros. Cabe mencionar que en nuestro contexto dentro de la organización, este problema no es relevante para nosotros.
- La programación extrema es un enfoque centralizado en vez de orientado al diseño y por tanto la falta de documentación detallada puede crear problemas de gestión y herencia en productos muy grandes cuando miembros del equipo de proyecto parten o nuevos miembros se integran después.(Nayab, 2010)

5.6 Entorno de Desarrollo

A continuación se plasman los elementos que se emplearán en desarrollo del sistema. Cabe mencionar que esto es apenas una lista preliminar de los componentes que estarán presentes en la versión final del software. En el caso de node.js se cuenta con un gestor de paquetes (Node Package Manager) que permite agregar módulos al proyecto existente para la reutilización de código. (Node.js Foundation) Para el mismo propósito existe una gama de *plugins* disponibles para jQuery. (jQuery Foundation)

Tabla 3 - Los componentes software que incluirá el sistema

Software	Descripción
Node.js 	Siendo un *runtime* de Javascript asíncrono orientado a eventos, Node está diseñado para la construcción de aplicaciones de red escalables. Muchas conexiones pueden ser gestionadas de manera simultánea. Con cada conexión se dispara un *callback*, pero si no hay tareas por realizar Node hiberna. (Node.js Foundation) Dentro de MVC, node.js se encarga del componente modelo.
express	Express es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles. Con miles de métodos de programa de utilidad HTTP y middleware a su disposición, la creación de una API sólida es rápida y sencilla. Express proporciona una delgada capa de características de aplicación web básicas, que no ocultan las características de Node.js que tanto ama y conoce.(Node.js Foundation) Dentro de MVC, Express se encarga del componente controlador.

Pug/Jade 	Pug (antes conocido como Jade) es un motor de plantilla de alto rendimiento influenciado por Haml e implementado con Javascript para Node.js y navegadores. (Pug Project, 2017) Dentro del patrón de diseño de software MVC, Pug se encarga del componente vista.
jQuery write less, do more.	jQuery es una librería de Javascript rápida y liviana con muchas características. jQuery facilita y simplifica la manipulación de documentos HTML, gestión de eventos, animación y AJAX a través de un API amigable que funciona en una multitud de navegadores. Con su combinación de flexibilidad y extensibilidad, jQuery ha transformado la manera en que millones de personas codifican en Javascript. (jQuery Foundation)
SQLite	SQLite es un motor de base datos SQL embebido, de alto rendimiento y dominio público. Según el sitio web del proyecto es el motor BD más utilizado en el mundo.(SQLite)
debian	Debian es un sistema operativo (SO) gratuito. Un sistema operativo es un conjunto básico de programas y utilidades que hacen que un ordenador funcione. Debian usa el núcleo Linux (el núcleo es el corazón de un SO), pero la mayoría de las herramientas básicas del SO provienen del proyecto GNU. Por lo tanto, nos referimos a Debian como el sistema operativo «Debian GNU/Linux», dando crédito a todas las fuentes. Debian GNU/Linux proporciona mucho más que un simple SO: incluye un amplio rango de programas. Concretamente ofrece más de 37.500 paquetes precompilados distribuidos en un formato que hace más sencilla la instalación en un ordenador.(Alonso, 2017)
git	Git es un sistema de control de versiones distribuido (SCVD) escrito en C. Un sistema de control de versiones permite la creación de una historia para una colección de archivos e incluye la funcionalidad para revertir la colección de archivos a otro estado. Otro estado puede significar a otra colección diferente de archivos o contenido diferente de los archivos.

Un completo entorno de desarrollo integrado (IDE por sus siglas en inglés) para crear aplicaciones para Windows, Android e iOS, además de aplicaciones web. Cabe mencionar que se utiliza la versión 2010 del software para el escrutinio el código fuente del sistema anterior debido a que el VBP (Visual Basic Project) del sistema actual fue trabajado originalmente con esa publicación.

6. Capítulo I: Estudio de Factibilidad

Antes de proceder con cualquier proyecto de implementación de sistemas es necesario establecer que dicho sistema sea viable. Los resultados de esta actividad se presentan como el estudio de factibilidad que consiste en datos claves y las conclusiones que se han extraído de ellas. El estudio de factibilidad está compuesto de 4 dimensiones investigativas: técnica, operativa, económica y legal. En cada uno de estos aspectos se pretende dar repuesta a la pregunta fundamental: "¿Se debería siquiera realizar el proyecto?"

6.1 Factibilidad Técnica

El Sistema de Apoyo a Ventas se ha considerado técnicamente factible por tres motivos principales. Primero, la empresa ya contaba con la infraestructura tecnológica necesaria para hospedar el software y el equipo informático capaz de gestionar dicha estructura para garantizar la funcionalidad continua de los sistemas. Segundo, la tecnología necesaria para obtener la funcionalidad deseada ya existía y estaba disponible de manera libre. Por último, le gerencia había reconocido la necesidad de fortalecer su enfoque tecnológico a los problemas de negocio, particularmente en el punto clave de las ventas.

En cuanto a la infraestructura informática y personal necesario para mantenerlo, la empresa goza de una conexión internet simétrica de fibra óptica de 50Megabits con varios IPs púbicos y control sobre su *firewall* para regular las entradas y salidas de trafico de red. Además en su data center tiene un arreglo de *VMWare ESXi hypervisors* en cual una maquina virtual de Linux podría hospedar la aplicación de Node.Js con poco impacto en los recursos. También se contaba con un equipo de 2 técnicos de soporte, 1 administrador de redes y 2 programadores encargados del mantenimiento y ampliación del sistema ERP (Planificación de Recursos Empresariales).

El ecosistema de proyectos de código abierto permite hacer grandes avances en cuanto a la ingeniería de los sistemas. Esto resulta en que sea cada vez menor el esfuerzo necesario para lograr la implementación del software. Trabajando con los módulos existentes en *Node Package Manager* (NPM) por ejemplo, el ingeniero en software puede orquestar distintos elementos de código para que los datos fluyan entre ellos y se cumplan procesos que agreguen valor a la organización.

Además de comprobar la capacidad técnica para realizar el proyecto, es necesario también mostrar también la necesidad técnica del mismo. La implementación del SAV fue necesaria porque significó un gran salto adelante en cuanto a la modernización de la cultura tecnológica de una empresa veterana que se ha visto obligada a evolucionar para seguir siendo competitiva durante sus 70+ años de existencia. Los puntos anteriores constituyen los medios, la oportunidad y el motivo para proceder con el proyecto propuesto.

Aspecto Técnico	Detalle
Infraestructura	• Data center privado con capacidad de hospedar aplicaciones 24/7 • Conectividad de internet fibra óptica con múltiples IPs estáticos
Capacidad Técnica	• 2 técnicos informáticos • 1 administrador de red • 2 programadores dedicados al sistema informático central SCM
Tecnología al alcance	• 350,000 librerías de software gratuitas para node en npm.org(Brown, 2017) • Librerías de encriptación fuerte • Ubicuidad de los dispositivos móviles

Figura 8 - Aspectos de la factibilidad técnica

6.1.1 Plataforma de Ejecución

Node.JS puede ejecutar en Windows, Mac, BSD y prácticamente cualquier distribución de Linux. Se ha elegido trabajar con Debian GNU/Linux en maquinas virtuales de VMWare ESXi debido a la facilidad de respaldar, duplicar o trasladar el servidor de un equipo físico a otro en caso de fallas de hardware.

	Plataforma de SAV	Recomendaciones de smartphone
Sistema operativo	Debian GNU/Linux	Android 3 o superior con Google Chrome
Procesamiento	Intel Xeon CPU E5-2680 @ 2.70GHz (1 core)	1.0Ghz
Memoria	1GB	1.0GB RAM
Almacenamiento	16GB	4GB

Figura 9 - Detalles técnicos de la plataforma en cual ejecuta SAV

6.2 Factibilidad Económica

La factibilidad económica fue demostrada de las siguientes formas. Primero se plasmó el costo de realizar el proyecto de la implementación del SAV. Luego se hizo una proyección de los beneficios económicos directos que resultarían del la adopción de dicha implementación. Los costos de inversión y los ahorros del uso del sistema por último se combinan para mostrar en cuanto tiempo el proyecto no solamente cubriría sus costos pero también cuando empezaría a generar ganancias.

En seguida se expone una tabla de tiempos de recuperación en dependencia del costo del sistema y de la cantidad de facturadores que se obviarían al integrar dicho sistema al flujo de trabajo de la empresa. El costo de implementación de proyectos similares realizado consultores externos oscila entre 4 mil y 6 mil dólares.(Freelancer.com, 2017) Vale mencionar que en esta proyección se utilizaron cifras conservadoras.

Tabla 4 - Proyección de tiempos de recuperación de inversión en SAV

NUMERO DE FACTURADORES	1	2	3	4	5	6
COSTO MENSUAL CORDOBAS	9,620.00	19,240.00	28,860.00	38,480.00	48,100.00	57,720.00
COSTO MENSUAL DOLARES	320.67	641.33	962.00	1,282.67	1,603.33	1,924.00
COSTO ANUAL CORDOBAS	115,440.00	230,880.00	346,320.00	461,760.00	577,200.00	692,640.00
COSTO ANUAL DOLARES	3,848.00	7,696.00	11,544.00	15,392.00	19,240.00	23,088.00
TIEMPO DE RECUPERACION DE INVERSION BASADO EN COSTO DE 6,000USD (EN MESES)	19	9	6	5	4	3

En Tabla 4, se puede observar que con solo la reducción de un puesto en facturación, el sistema cubriría sus costos en 19 meses. Según entrevista no estructurada con la Gerencia de Repuestos, la reducción total será de al menos 4 personas. Eso significaría un tiempo de 5 meses de recuperación de la inversión en el sistema.

6.3 Factibilidad Operativa

Antes de proceder con cualquier proyecto de implementación de sistemas es esencial establecer que dicho sistema sea necesario para la organización, que operará y que exista recurso humano para operarlo. Además hay que explicar cómo se atendieron los asuntos relacionados a la seguridad, la gestión del cambio y la capacitación. El conjunto de respuestas a estas preguntas constituye la factibilidad operativa.

El sistema se considera necesario debido a la naturaleza del trabajo de ventas remotas. Los aproximadamente 40 vendedores necesitan una herramienta que les permita consultar inventarios de manera rápida e independiente. La capacitación a los usuarios se ha combinado con la reunión semanal del equipo de ventas los sábados por las mañanas y la elaboración de video manuales para que los usuarios puedan repasar los procesos del sistema de manera independiente. De esta manera se ha mitigado el riesgo del uso incorrecto del sistema.

La empresa cuenta con el personal necesario para operar el sistema pues existe un departamento de informática capaz de mantener los equipos de red y hardware en buen estado para que el sistema siga al aire. La disposición a utilizar el sistema es total pues el mismo es un reemplazo de un producto similar que en su momento agregaba valor y ahorraba trabajo manual. Por tanto el personal de la empresa tiene mucho entusiasmo para el SAV.

Aparte de que la inversión para la empresa se recuperaría a mediano plazo, también se debe tomar en cuenta que, al automatizar el proceso de registrar cotizaciones, el nivel de personal existente se podría reducir o el personal se podría dedicar a actividades más rentables en otras aéreas de la empresa. Esto es compatible con la fuerte cultura de rotación interna presente en la empresa y fomentaría la diversificación de la experiencia laboral del personal.

La seguridad operativa del sistema se obtiene a través del uso de encriptación. Las contraseñas de usuario son *hashed* antes de ser registradas en la base datos. Todos los datos transmitidos entre el navegador y el servidor web son encriptados antes de entrar a transito. Se adquirió un certificado SSL de *LetsEncrypt* para poder utilizar el protocolo seguro HTTPS en cumplimiento con las mejores prácticas de seguridad.

A continuación se presentan dos versiones del mismo proceso de negocio a nivel macro. El primero es el proceso de ventas sin ningún sistema web. El segundo es el mismo proceso, pero simplificado ya que el vendedor puede contar con un mecanismo para efectuar su cotización de manera independiente.

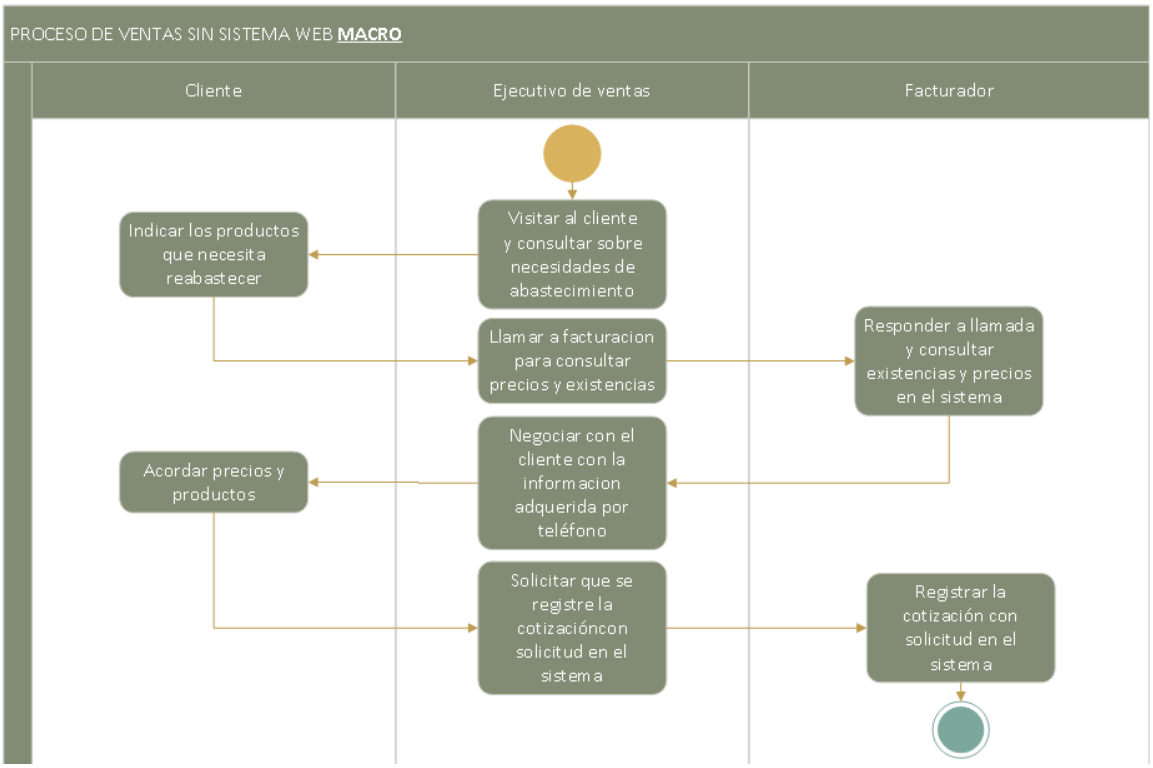


Figura 10 - Proceso de negocio de ventas sin sistema web

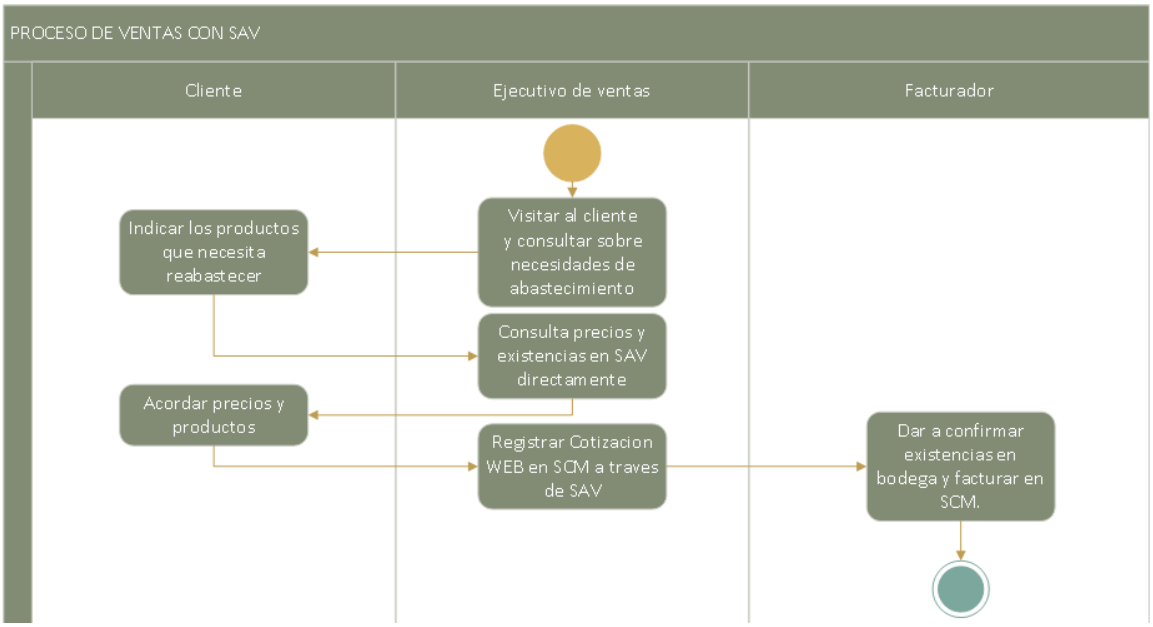


Figura 11 - Proceso de negocio de ventas con sistema web

6.4 Factibilidad Legal

El tema legal tiene varios aspectos, entre ellos el licenciamiento de las herramientas de software, los acuerdos entre cliente y proveedor y la propiedad intelectual. El sistema se está implementando con herramientas gratuitas de código abierto en las instalaciones y la propiedad de la empresa. Existe un acuerdo de no confidencialidad entre la empresa y el proveedor (el monografista en este caso) en cual se estipula que ninguna información que el proveedor pudo haber conocido durante el transcurso del proyecto debe ser transmitida a un tercero. También existe un acuerdo confidencial de nivel de servicio que plasma los términos y condiciones bajo cual el proveedor está obligado a brindar soporte y respuesta por el sistema. Por último existe un acuerdo explícito que enuncia que la propiedad intelectual (el código fuente) pertenece exclusivamente a la persona jurídica de la empresa.

6.5 Beneficios de automatizar

Se han identificado beneficios de la automatización tanto tangibles como intangibles.

Tangibles

- El ahorro de esfuerzo humano debido al incremento de la eficiencia
- La reducción de errores
- Reducción de costos de papelería y desgaste en equipos de impresión
- Facilidad de trabajo desde cualquier sitio
- Mayor trazabilidad de información y documentos

Intangibles

- La mejora de la atención al cliente
- Mayor satisfacción del cliente
- Moralizar el usuario al facilitar su trabajo
- Le mejora del imagen de la empresa al utilizar una herramienta moderna
- Incremento de capacidad de procesamiento

7. Capítulo II: Análisis del Sistema

7.1 Requerimientos del Sistema

Requerimientos Funcionales

Tabla 5 - Requerimiento funcional RF1 y sub-requerimientos

Código	Tipo	Descripción del Requerimiento	Iteración
RF1	**RFS**	**Permitir la consulta de clientes de la empresa a la base datos del sistema informático SCM en base a ciertos parámetros**	
-	-	**Consultar clientes filtrando por las siguientes propiedades**	
RF1A	RFU	Consultar clientes por nombre	MARK1
RF1B	RFU	Consultar clientes por código	MARK1
RF1C	RFU	Consultar clientes por RUC	MARK2
RF1D	RFU	Filtrar resultados por tipo (crédito o contado)	MARK1

Tabla 6 - Requerimiento funcional RF2 y sub-requerimientos

Código	Tipo	Descripción del Requerimiento	Iteración
RF2	**RFS**	**Permitir la consulta de productos de la empresa en la base datos del sistema informático SCM en base a ciertos parámetros**	
RF2A	RFU	Consultar productos por código alterno (identificador único)	MARK1
RF2B	RFU	Consultar productos por código original (identificador único del fabricante)	MARK2
RF2C	RFU	Consultar productos por descripción	MARK2
RF2D	RFU	Consultar productos por aplicación	MARK2
RF2E	RFU	Consultar productos por línea	MARK3
RF2F	RFU	Permitir al usuario elegir el tipo de precio que se va a presentar en los resultados de la búsqueda según sus acceso a los tipos de precio	MARK4
RF2G	RFU	Filtrar resultados por existencia (si hay o no hay)	MARK1

Tabla 7 - Requerimiento funcional RF3 y sub-requerimientos

Código	Tipo	Descripción del Requerimiento	Iteración
RF3	**RFS**	**Permitir el registro de cotizaciones en la base datos del sistema informático SCM de tal manera que estas se puedan procesar hasta llegar ser facturadas**	
RF3A	RFU	Permitir la especificación de precios de los productos en la cotización	MARK1
RF3B	RFU	Permitir la especificación de porcentajes de descuento	MARK3
RF3C	RFU	Permitir la especificación de cantidades de productos en la cotización	MARK1
RF3D	RFU	Agregar productos a la cotización antes de registrar la cotización	MARK1
RF3E	RFU	Eliminar productos de la cotización antes de registrar la cotización	MARK1
RF3F	RFU	En el caso de trámites de ruta, permitir la especificación del nombre del cliente	MARK3

Tabla 8 - Requerimiento funcional RF4 y subrequerimientos

	Tipo	Descripción del Requerimiento	Iteracion
RF4	**RFS**	**Permitir la gestión de usuarios**	
-	-	**Permitir la creación de usuarios, registrando las propiedades:**	
RF4.1	RFU	Nombre	MARK1
RF4.2	RFU	Apellido	MARK1
RF4.3	RFU	Username	MARK1
RF4.4	RFU	Password	MARK1
RF4.5	RFU	Código de vendedor SCM	MARK1
RF4.6	RFU	Serie asignada	MARK4
RF4.7	RFU	Bodega asignada	MARK4
RF4.8	RFU	Tipo de cuenta (véase Jerarquía de Usuario)	MARK3
RF4.9	RFU	Estado (Habilitado o Deshabilitado)	MARK3
-	-	**Permitir la modificación de cuentas de usuario existentes, registrando las propiedades:**	
RF4.10	RFU	Nombre	MARK1
RF4.11	RFU	Apellido	MARK1
RF4.12	RFU	Password	MARK1
RF4.13	RFU	Código de vendedor SCM	MARK1
RF4.14	RFU	Serie asignada	MARK1
RF4.15	RFU	Bodega asignada	MARK4
RF4.16	RFU	Tipo de cuenta (véase Jerarquía de Usuario)	MARK4
RF4.17	RFU	Estado (Habilitado o Deshabilitado)	MARK3

Requerimientos No Funcionales

(Restricciones a los servicios o funciones ofrecidos por el sistema tales como restricciones de tiempos (*timing*), restricciones al proceso de desarrollo, estándares, etc.)

Tabla 9 - Requerimientos no funcionales (RNF)

	Descripción del Requerimiento
RNF1	Web - El sistema debe poder ser utilizada desde cualquier dispositivo que cuenta con conexión a internet y navegador web, por ejemplo, celulares inteligentes (*Smartphone*)
RNF2	Eficiente (Dado que las conexiones móviles tienen ancho de banda y consumo de datos limitado, el manejo de datos debe llevarse de tal manera que se minimiza el trafico necesario entre el navegador del usuario el servidor para maximizar el provecho de los recursos de la organización)
RNF3	*Responsive* (*Tablets*, *Smartphones*, PC)
RNF4	Seguro – Debería utilizar encriptación en la transmisión de datos
RNF5	Jerárquico – Debería distinguir entre cuentas de vendedores, supervisores y administradores y los accesos a interfaces que se otorgan a cada uno

7.2 Casos de Uso

El caso de uso capta las interacciones que ocurren entre los productores y consumidores de información y el propio sistema. Antes de entrar a los casos de uso es necesario plasmar la jerarquía de usuarios del sistema como se hace a continuación. Luego se exponen los casos de uso vía un diagrama de casos de uso. Por último se profundiza los casos de uso a través de tablas que explican cada uno detalladamente. De esta manera se resume lo que se hace en el sistema y quienes lo hacen.

Tabla 10 - Descripción de niveles de usuario

Actor	Descripción
Admin	El *superusuario* con permisos de crear y modificar cuentas de usuario
Supervisor	El usuario que corresponde al supervisor de equipo de ventas. Puede hacer todo lo que hace el vendedor más visualizar cotizaciones de todo su equipo
Vendedor	Miembro de equipo de ventas que tiene un supervisor. Puede consultas productos, clientes, registrar cotizaciones y visualizar sus propias cotizaciones que ya han sido registradas.

Figura 12 - Modelo de jerarquía de usuarios

7.2.1 Especificación de los casos de uso

Tabla 11 - Caso de uso CU1 - Login

ID	CU1
Nombre	Login
Actores	Vendedor, Supervisor, Admin
Prioridad	Alta
Frecuencia	Alta
Descripción: ✓ Permite al usuario ingresar al sistema e iniciar sesión.	
Precondiciones: ✓ El actor habrá navegado a la página de entrada de SAV**.**	
Flujo normal: 1) El actor ingresa su nombre de usuario y contraseña 2) El sistema valida los credenciales del actor 3) El sistema dirige el navegador del actor a la página de bienvenida	
Resultados: ✓ El actor ha ingresado al sistema y puede realizar operaciones.	

Tabla 12 - Caso de uso CU2 - Búsqueda de cliente

ID	CU2
Nombre	Búsqueda de cliente
Actores	Vendedor, Supervisor
Prioridad	Alta
Frecuencia	Alta
Descripción: Permite consultar los clientes de la empresa	
Precondiciones: ✓ Estar autenticado en el sistema	
Flujo normal: 1) El actor ingresa parámetros en el formulario de búsqueda de cliente. 2) SAV consulta la base datos de SCM y devuelve los resultados.	
Resultados: ✓ Se presenta la información al actor en su dispositivo.	

Tabla 13 - Caso de uso CU3 - Búsqueda de producto

ID	CU3
Nombre	Búsqueda de producto
Actores	Vendedor, Supervisor
Prioridad	Alta
Frecuencia	Alta
Descripción: Permite consultar los productos de la empresa y sus existencias	
Precondiciones: ✓ **Estar autenticado en el sistema**	
Flujo normal: 1) El actor ingresa parámetros en el formulario de búsqueda de producto. 2) SAV consulta la base datos de SCM y devuelve los resultados.	
Resultados: ✓ Se presenta la información al actor en su dispositivo.	

Tabla 14 - Caso de uso CU4 - Registrar cotización

ID	CU4
Nombre	Registrar cotización
Actores	Vendedor, Supervisor
Prioridad	Alta
Frecuencia	Alta
Descripción: Permite registrar la cotización con el cliente y productos (cantidades, precios, descuentos) seleccionados tal que se pueden visualizar en la aplicación de escritorio SCM en las oficinas centrales de Facturación	
Precondiciones: ✓ El actor debe estar autenticado en el sistema ✓ El actor debe haber elegido un cliente y al menos un producto	
Flujo normal: 1) El actor navega a la página de guardar cotización actual 2) El actor da *click* en el botón de guardar 3) Se presenta un mensaje confirmando que la cotización fue registrada correctamente	
Resultados: ✓ La cotización está registrada en el sistema SCM	

Tabla 15 - Caso de uso CU5 - Visualización de cotizaciones propias

ID	**CU5**
Nombre	Visualización de cotizaciones propias
Actores	Vendedor
Prioridad	Alta
Frecuencia	Mediana
Descripción: Permite al actor ver un listado de las cotizaciones que ha registrado en SCM a través del sistema SAV	
Precondiciones: ✓ El actor debe estar autenticado ✓ El actor debe haber registrado alguna cotización en el rango de fechas que especifique	
Flujo normal: 1) El actor navega al panel de reportes a vía el menú 2) El actor elige un rango de fechas en el modal de parámetros 3) Se ejecuta la consulta pertinente en el sistema	
Resultados: ✓ Se presentan los resultados al actor en su dispositivo	

Tabla 16 - Caso de uso CU6 - Reporte de cotizaciones de equipo

ID	**CU6**
Nombre	Reporte de cotizaciones de equipo
Actores	Supervisor
Prioridad	Mediana
Frecuencia	Mediana
Descripción: Permite al supervisor visualizar listado y detalle de cotizaciones de cualquier miembro de su equipo	
Precondiciones: ✓ El actor debe ser de tipo Supervisor y esta autenticado	
Flujo normal: 1) El actor navega al panel de reportes a vía el menú 2) El actor elige un vendedor y rango de fechas en el modal de parámetros 3) Se ejecuta la consulta pertinente en el sistema	
Resultados: ✓ Se presentan los resultados al actor en su dispositivo	

Tabla 17 - Caso de uso CU7 - Crear usuario

ID	**CU7**
Nombre	Crear usuario
Actores	Admin
Prioridad	Alta
Frecuencia	Baja
Descripción: Permite al actor agregar una nueva cuenta de usuario al sistema	
Precondiciones: ✓ El actor debe estar autenticado con el nivel de autoridad administrativa ✓ No se deben duplicar nombres de usuario (*username*)	
Flujo normal: El actor navega al panel administrativo El actor elige “crear usuario” en el panel administrativo El actor debe completar el formulario con la información pertinente al usuario	
Resultados: ✓ La cuenta de usuario es creada y el usuario nuevo puede ingresar al sistema	

Tabla 18 - Caso de uso CU8 - Modificar usuario

ID	**CU8**
Nombre	Modificar usuario
Actores	Admin
Prioridad	Alta
Frecuencia	Baja
Descripción: Permite al actor modificar una cuenta de usuario existente	
Precondiciones: ✓ El actor debe estar autenticado con el nivel de autoridad administrativa ✓ La cuenta de usuario debe existir	
Flujo normal: El actor navega al panel administrativo El actor elige “modificar usuario” en el panel administrativo El actor debe completar el formulario con la información pertinente al usuario	
Resultados: ✓ La cuenta de usuario es modificada y se reflejará en el próximo ingreso del usuario	

7.2.2 Diagramas de caso de uso

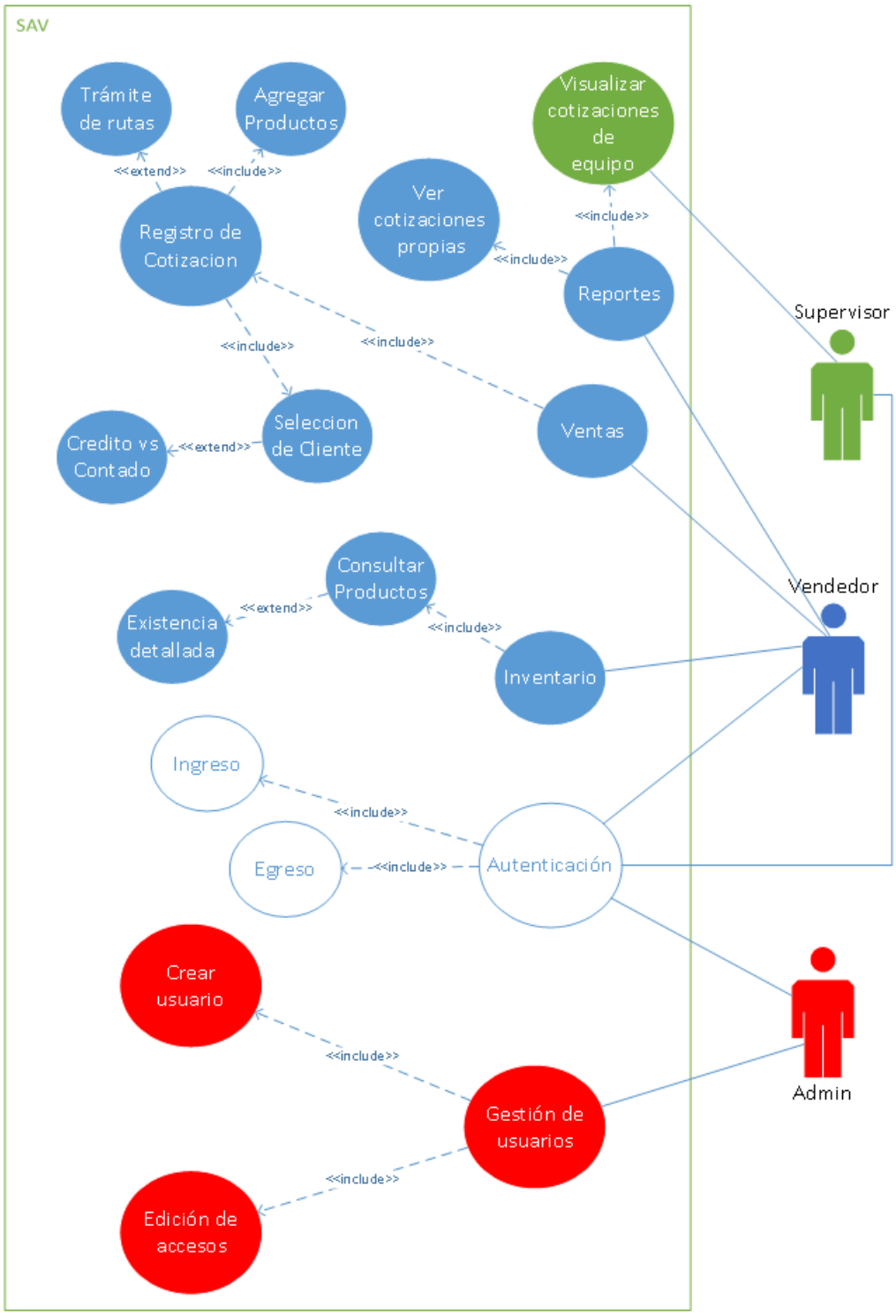


Figura 13 - Diagrama de casos de uso

7.3 Modelo relacional

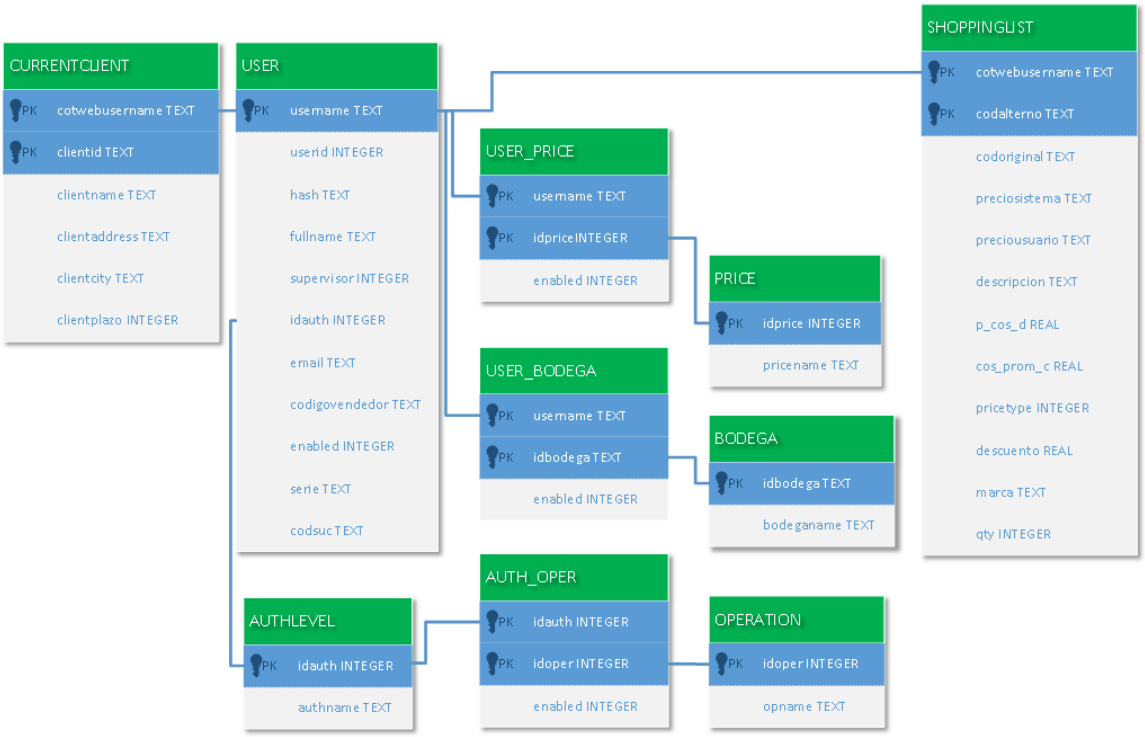


Figura 14 - Modelo conceptual en SQLITE

Figura 15 - Modelo conceptual de tablas originales en SCM

8. Capítulo III: Diseño del Sistema

El diseño del sistema se formó a partir del análisis de las necesidades y la elaboración de los requerimientos. El resultado es un conjunto de planos conceptuales que sirven para realizar una implementación ordenada y eficiente.

8.1 Diagrama de actividades

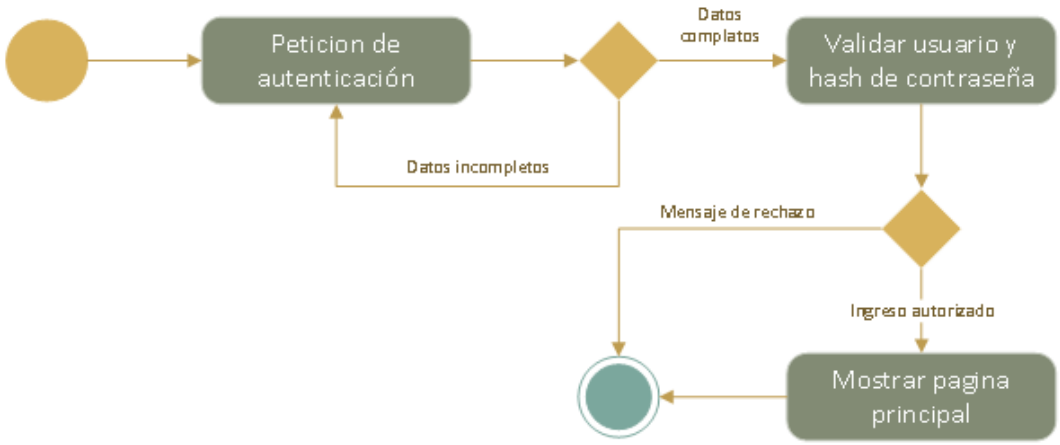


Figura 16 - Diagrama de actividad para ingreso de usuario (*Login*)

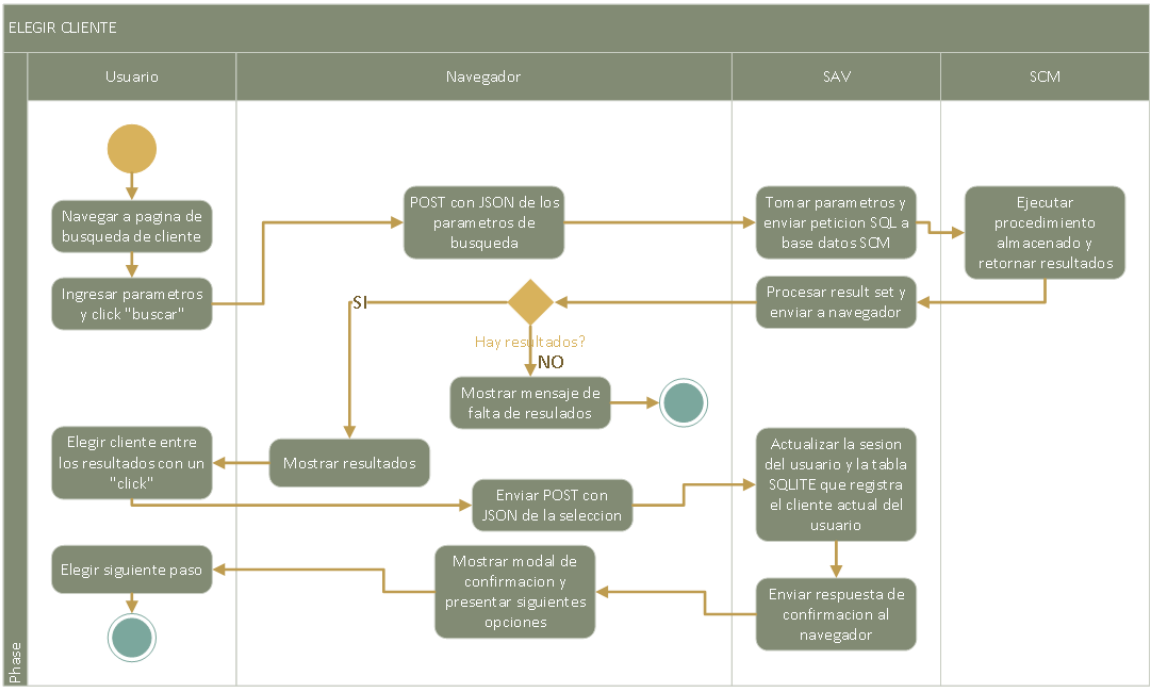


Figura 17 - Diagrama de actividad del proceso de elección de cliente

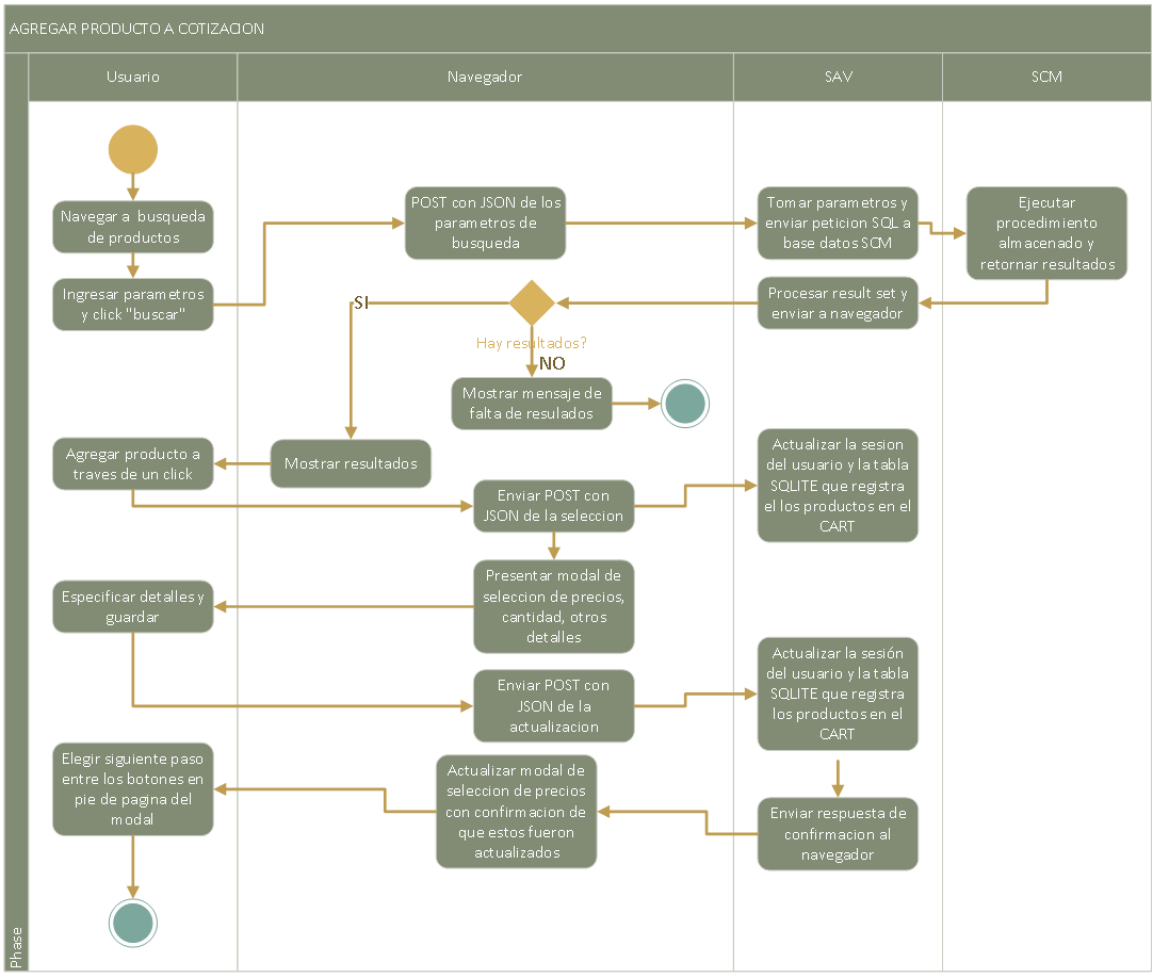


Figura 18 - Diagrama de actividad del proceso de agregar productos a la cotización

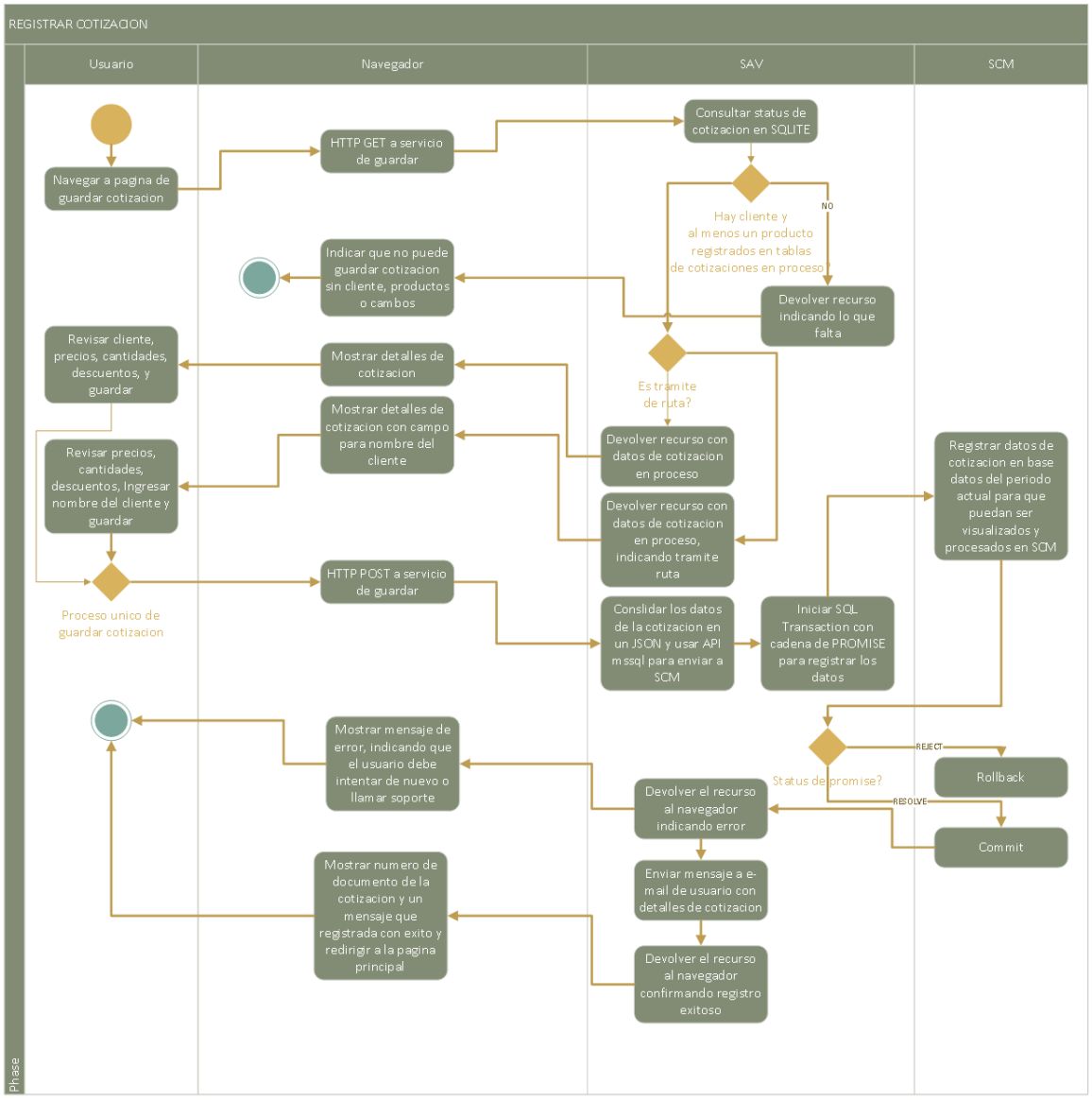


Figura 19 - Diagrama de actividad para el registro de la cotización

Como se puede observar en los diagramas de actividad, el único rol que juega el sistema SCM es de responder a consultas de datos de clientes y productos y el registro de datos de las cotizaciones para que estas puedan ser visualizadas y procesadas en el cliente de escritorio de SCM.

8.2 Diagramas de secuencia

Los procesos principales del sistema se detallan a nivel de los componentes técnicos abajo.

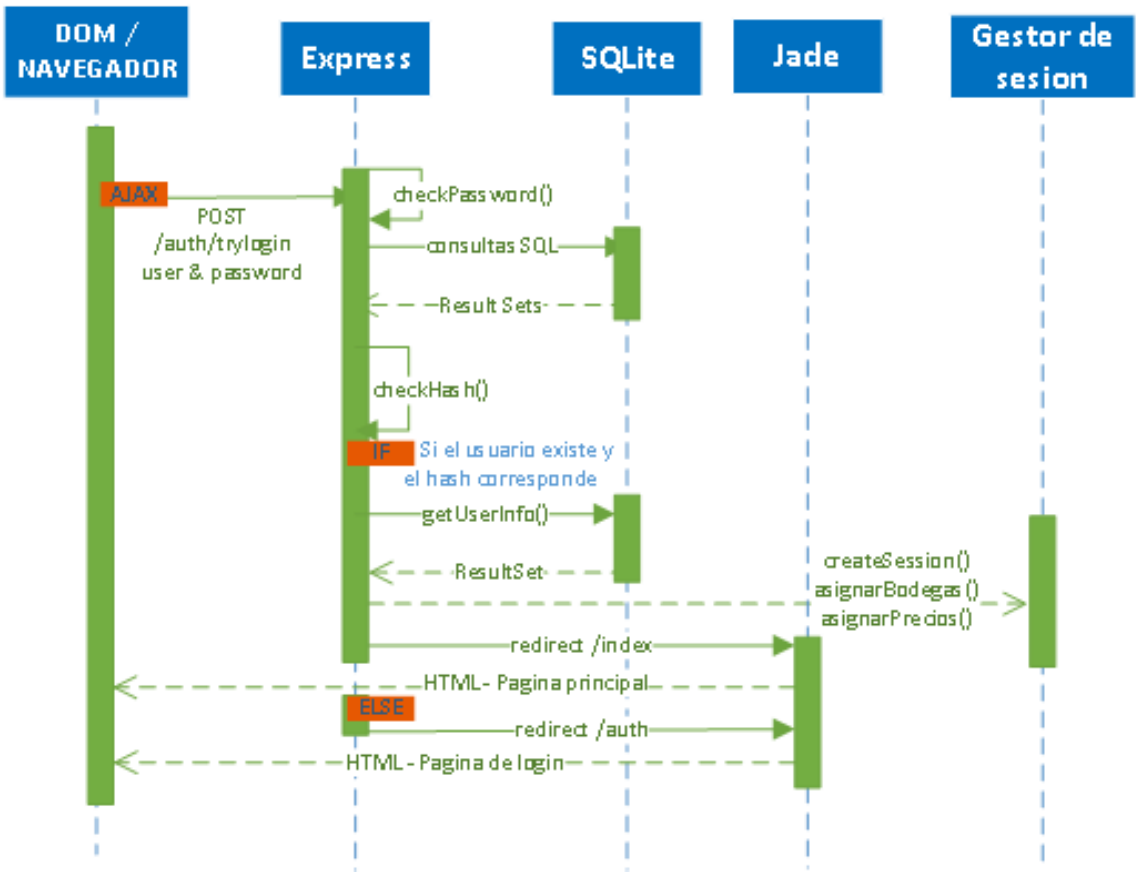


Figura 20 - Diagrama de secuencia de ingreso/login de usuario

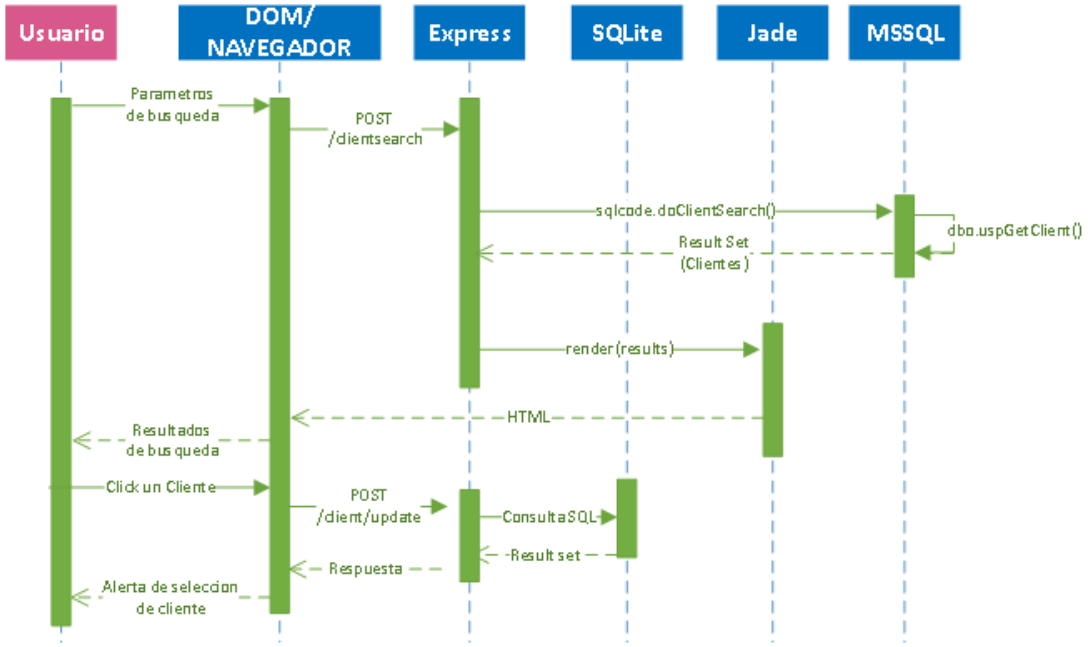


Figura 21 - Diagrama de secuencia de selección de cliente

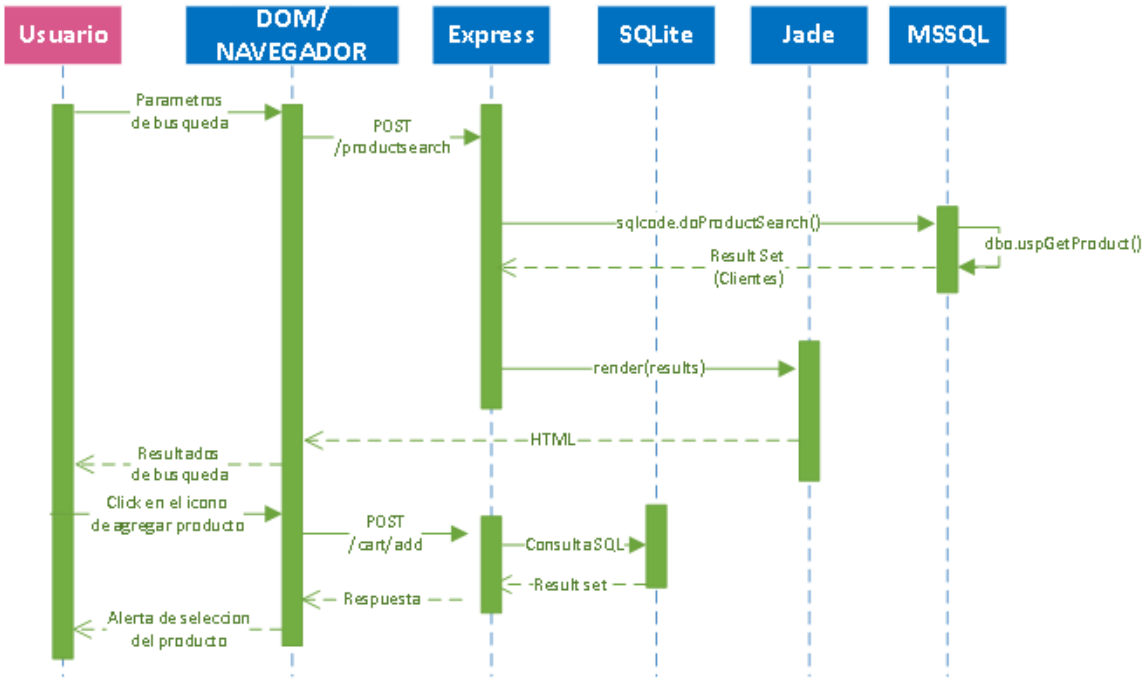


Figura 22 - Diagrama de secuencia de selección de producto

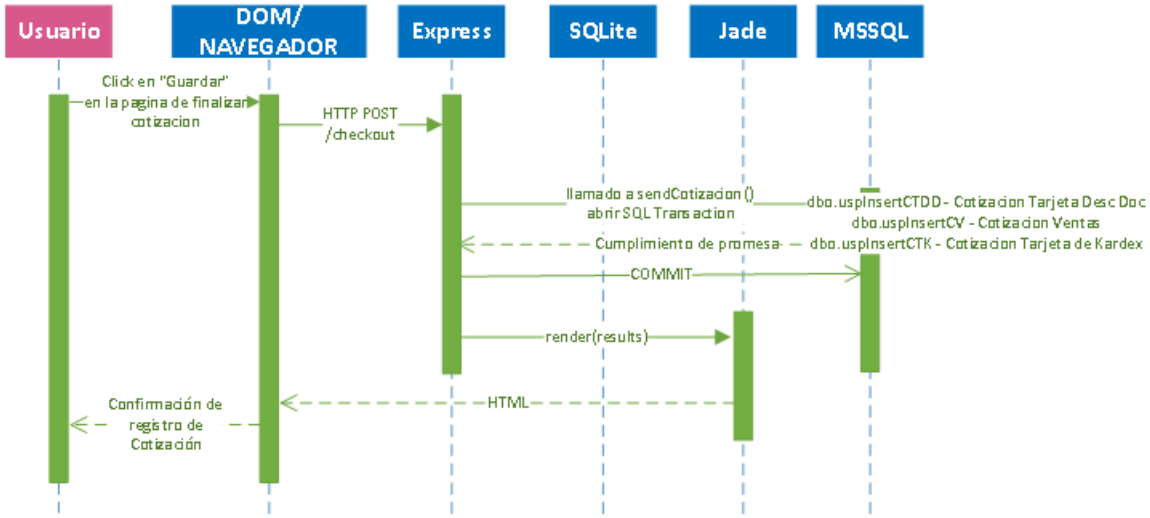


Figura 23 - Diagrama de secuencia de guardar cotización

8.3 Modelos de navegación

A continuación se presentan los modelos de navegación para cada tipo de usuario.

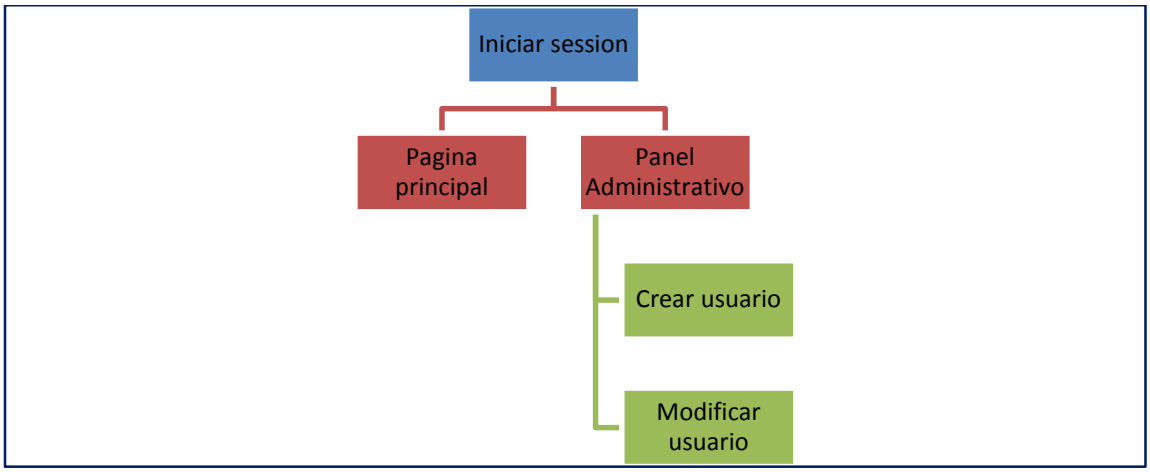


Figura 24 - Modelo de navegación de usuario administrador

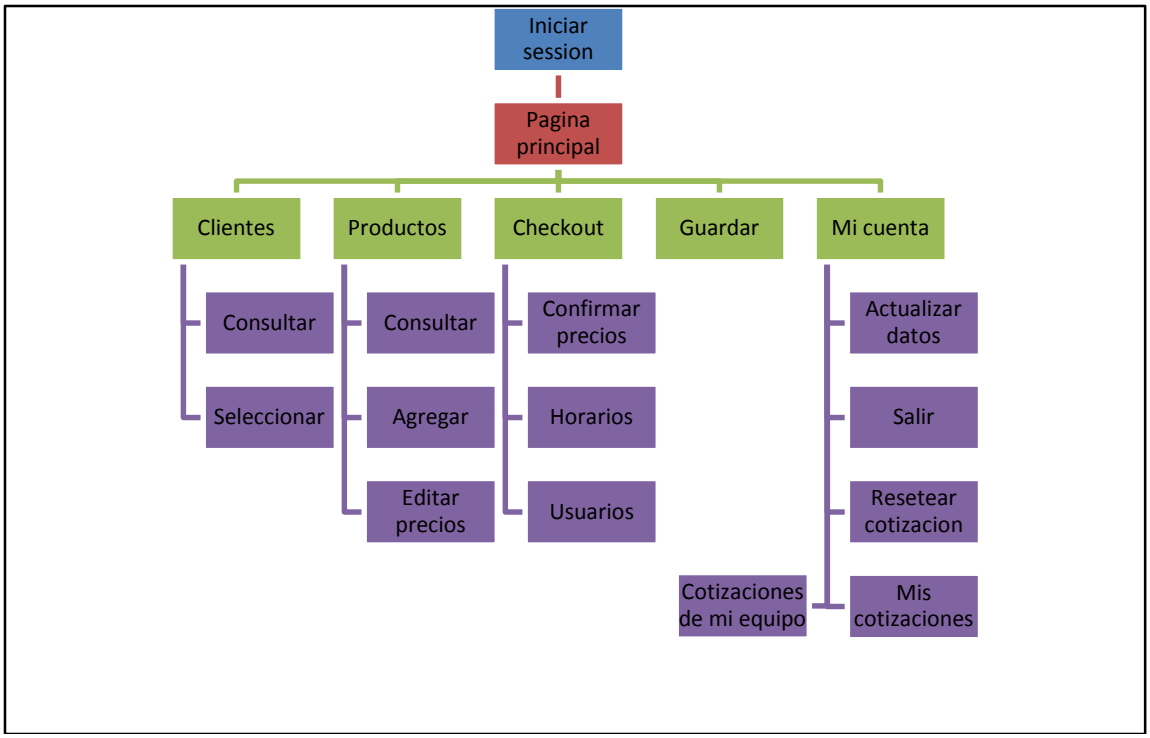


Figura 25 - Modelo de navegación de usuario supervisor

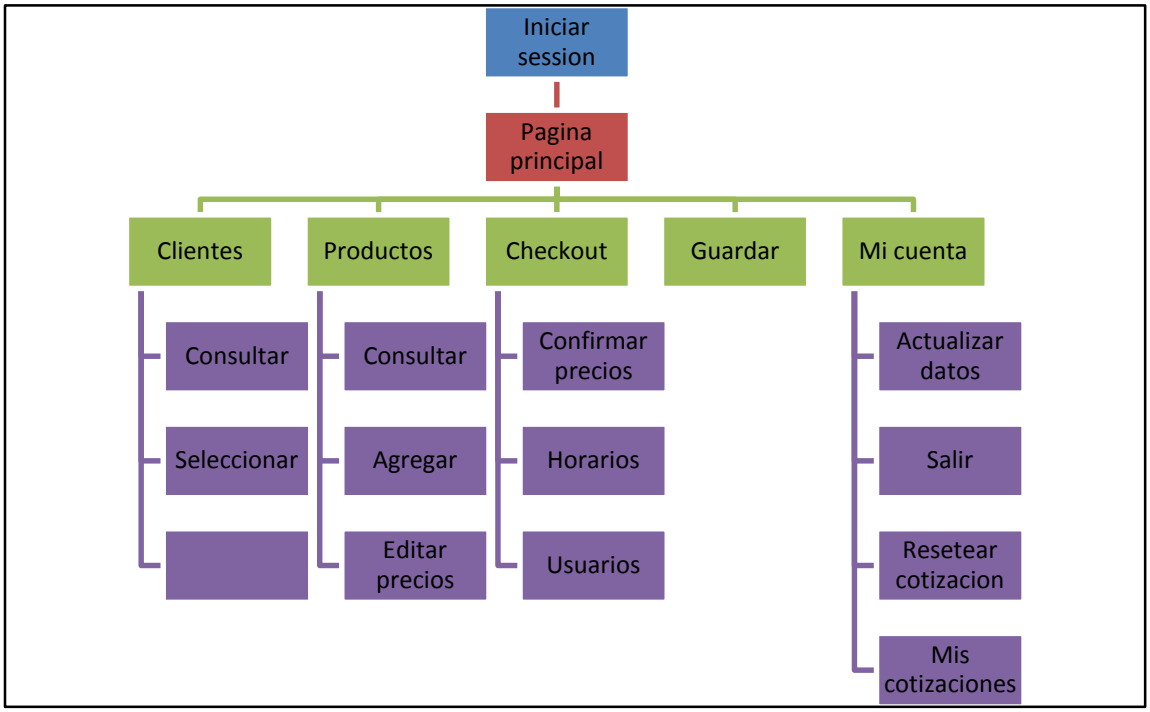


Figura 26 - Modelo de navegación del vendedor

8.4 Diagrama de componentes

En el siguiente diagrama se representan los componentes del sistema y las relaciones dependientes entre ello.

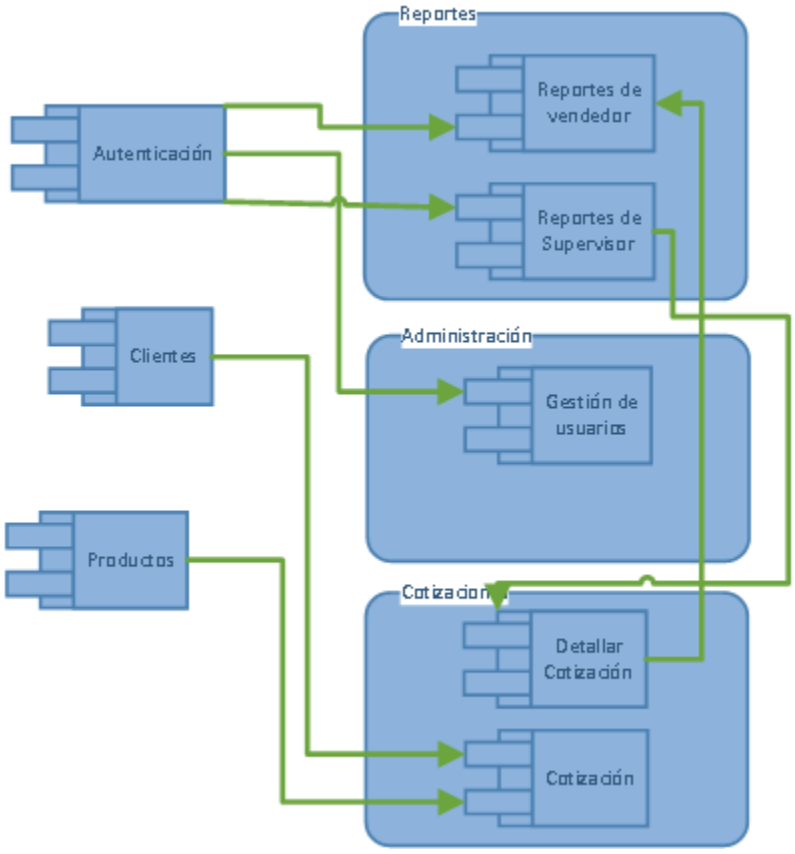


Figura 27 - Diagrama de componentes

8.5 Diagrama de despliegue

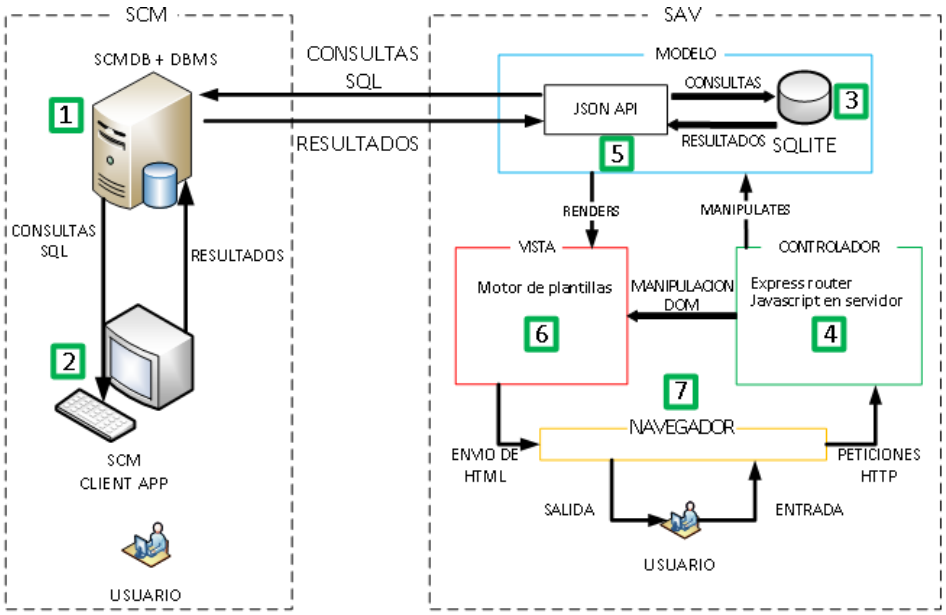


Figura 28 - Diagrama de despliegue del sistema SAV y su interacción con SCM

Los límites de los respectivos sistemas se indican con las líneas discontinuas. Los elementos designados [1] y [2] constituyen el sistema existente SCM con su equipo físico de Windows Server y SQL Server y la aplicación cliente que corre en estaciones de trabajo Windows. El componente modelo que contiene el JSON API [5] se encarga de interactuar con el gestor de la base datos de SCM y la base datos SQLite [3]. El API se diseñará en base a los requisitos funcionales del sistema. Para cada RF habrá uno o más "servicios" en el API que apoyen en cumplir con dicho RF. Por ejemplo, para la gestión de cuentas de usuario habrían al menos dos servicios: `/createUser` y `/updateUser`. Se accede a estos servicios a través del navegador [7] que hace peticiones HTTP al controlador [4]. El API en turno envía los datos al motor de plantillas dentro del componente vista [6] para que este rellene las vistas y las envié al navegador como HTML. Completando el ciclo de entrada y salida, el usuario puede nuevamente realizar otra solicitud en el navegador para que este se comunique con el controlador, etc.

8.6 Arquitectura del sistema

El sistema utiliza la arquitectura MVC (Modelo-Vista-Controlador). Estos subsistemas corresponden a

- Vista (Interfaz grafico): Se encarga de la entrada y salida de datos con el usuario, consiste en controles visuales con cuales el usuario debe interactuar para utilizar el sistema.
- Controlador: Se encarga de la recepción de datos del cliente web (navegador), el procesamiento y la ejecución de consultas al modelo.
- Modelo: Es el encargado a la gestión de las bases de datos

Tabla 19 - Elementos de los subsistemas MVC

Vista	Controlador	Modelo
HTML CSS Javascript, jQuery Bootstrap Jade	node.js express	sqlite3 mssql

En el interfaz gráfico empleamos tecnologías de diseño tales como Jade, HTML, CSS, y JavaScript. También aprovechamos *frameworks* y librerías tales como Bootstrap y jQuery siguiendo la buena práctica de la reutilización de código.

La parte del controlador fue implementada utilizando node.js, un JavaScript *run-time environment* que permite ejecutar código JavaScript en el lado del servidor (*server-side*) y por tanto la unificación del desarrollo web bajo un solo lenguaje de programación. Combinando Express (el máximo *framework* web) con node.js se obtiene un servidor web capaz de atender peticiones de un navegador. Por último se utiliza sqlite3 para registrar datos internos del SAV y mssql (Microsoft SQL) para las consultas de datos del ERP de la empresa.

8.7 Diseño del interfaz de usuario

A partir de los requerimientos funcionales de usuario, se trabajan bosquejos de cómo podría verse el interfaz de usuario vista por vista. Esto permite obtener la retroalimentación del usuario antes de invertir recursos en la codificación de interfaces que podrían no satisfacer al cliente a pesar de cumplir con la parte funcional de la especificación técnica. Luego de lograr la aprobación del concepto se proceder a trabajar en la versión estática (que no reacciona a ningún evento de manipulación de datos) antes de finalmente crear la versión dinámica en la fase de implementación.

Bosquejo en Microsoft Visio	Versión dinámica en Google Chrome

Figura 29 - El progreso del interfaz de usuario de bosquejo hacia versión dinámica

En Figura 29 se puede observar el interfaz grafico desde su primer concepto hasta su versión final. El interfaz consiste en un encabezado (*header*) en la parte

superior que contiene un menú de navegación. También se presenta un área de información donde se muestran datos relevantes al documento en proceso, un área de contenido y un pie de página (*footer*).

A continuación se muestran algunos interfaces del sistema. Por razones de brevedad sólo se presentan las pantallas que revelan las características principales. Para ver más en detalle y de manera secuencial el uso de la herramienta se incluye el manual de usuario en los anexos del presente documento.

Figura 30 - Pantalla de bienvenida (Versión móvil)

Al acceder al sistema después de ingresa un nombre de usuario y contraseña correctamente, el navegador es dirigido a la página de bienvenida. Aquí se plasman los cambios más recientes al sistema. También se ofrece un *videomanual* embebido para que el operador pueda aprender sobre el uso correcto del sistema sin tener que leer documentación que en la práctica suele ser ignorada por la mayor parte de los usuarios.

AGREGAR	Alterno	Original	Descripcion	Aplicacion	Marca	Precio	CA	CARS	TODO	Detalle
	0K2A3-34-700E-MANDO	0K2A3-34-700E	AMORTIGUADOR	SEPHIA 1(S-Car),GAS	MANDO	2138.94	12	0	18	Mostrar
	0K2A3-34-900E-MANDO	0K2A3-34-900E	AMORTIGUADOR	SEPHIA 1(S-Car),GAS	MANDO	2138.94	12	0	18	Mostrar
	0K30A-28-700H-MANDO	0K30A-28-700H	AMORTIGUADOR	RIO TRAS, R/L	MANDO	1313.05	3	0	11	Mostrar
	0K30A-34-700K-ONN	0K30A-34-700K	AMORTIGUADOR	RIO DEL, RH 03	ONNURI	1689.73	4	0	6	Mostrar
	0K30A-34-700K-PMC	0K30A-34-700K	AMORTIGUADOR	RIO DEL, RH	PMC	1660.36	8	0	12	Mostrar
	0K30A-34-700-MANDO	0K30A-34-700	AMORTIGUADOR	RIO DEL, RH	MANDO	1962.38	33	3	50	Mostrar
	0K30A-34-900K	0K30A-34-900K	AMORTIGUADOR	RIO DEL, RH	IST	1545.3	0	0	3	Mostrar
	0K30A-34-900K-ONN	0K30A-34-900K	AMORTIGUADOR	RIO DEL, LH 03	ONNURI	1689.73	1	0	1	Mostrar
	0K30A-34-900-MANDO	0K30A-34-900	AMORTIGUADOR	RIO DEL, LH	MANDO	1962.38	48	3	70	Mostrar
	0K34C-62-620	0K34C-62-620	AMORTIGUADOR	KIA RIO 99-02	MOBIS	793.15	2	0	4	Mostrar
	0K34C-62-620-PMC	0K34C-62-620	AMORTIGUADOR	RIO 99-02 ASCENSOR PUERTA TRAS RH	PMC	572.83	1	0	14	Mostrar

Figura 31 - Resultados de búsqueda de productos (Escritorio)

Aquí se presenta el módulo de búsqueda de productos. Además de ver precios y existencias generales como muestra Figura 31, el usuario puede ver detalles de existencias por bodega física para tener la mejor idea posible de la disponibilidad del inventario. En Figura 32 se puede apreciar el diálogo que muestra estas existencias.

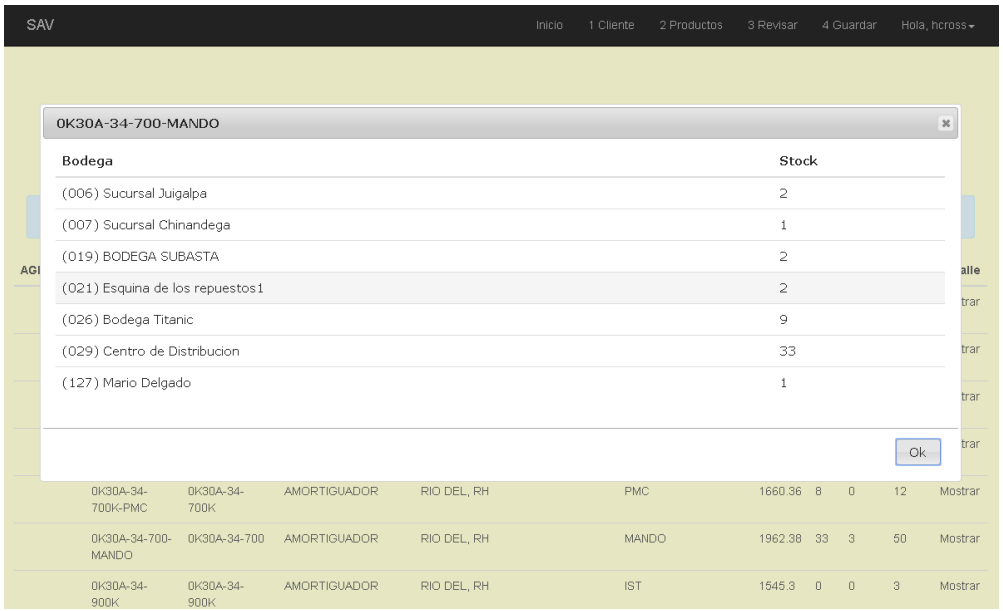


Figura 32 - Diálogo de existencias por bodega (Escritorio)

Si el usuario decide agregar un producto a la cotización, se abre el siguiente diálogo de Bootstrap en el cual debe elegir precios, cantidades y descuentos (Figura 33). Por último el usuario puede guardar la cotización en la base datos de SCM. Al guardarse correctamente se presenta un mensaje de éxito (Figura 34) y la cotización puede ser visualizada en el módulo de reportes (Figura 35).

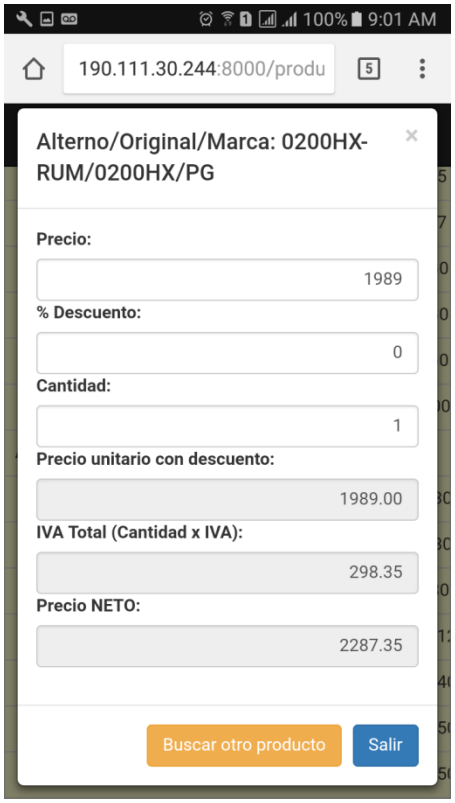


Figura 33 - Diálogo de selección de precios, descuentos y cantidades (Móvil)

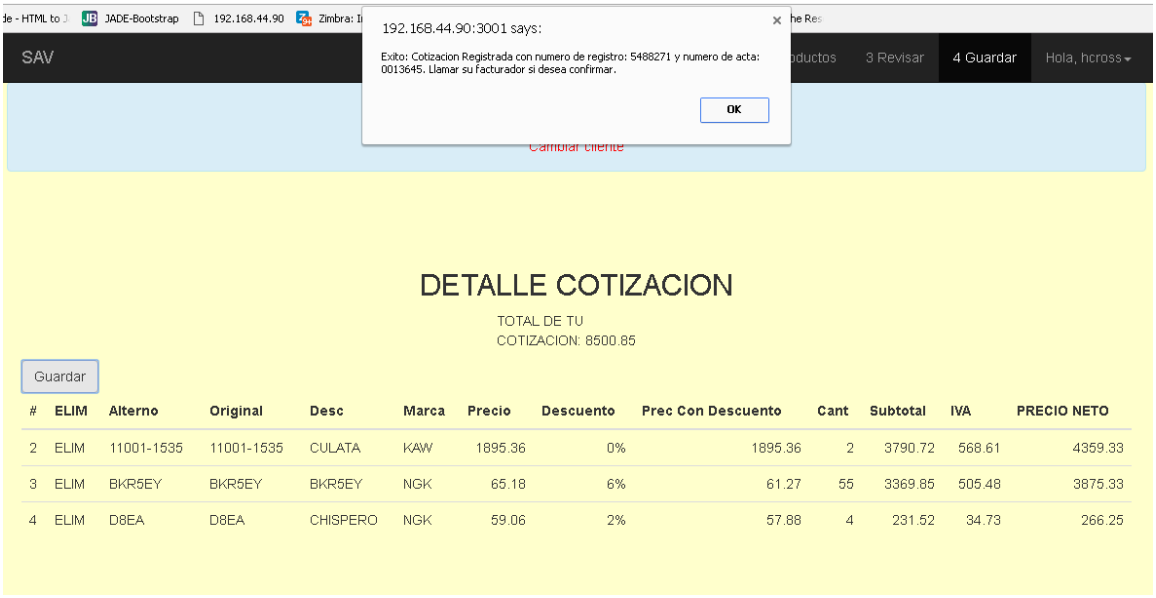


Figura 34 - Página de revisión antes de guardar cotización (Escritorio)

Figura 35 - Página de reporte de cotizaciones ya registradas con dialogo de detalle (Escritorio)

9. Capítulo IV: Implementación del Sistema

9.1 Estructura del código fuente

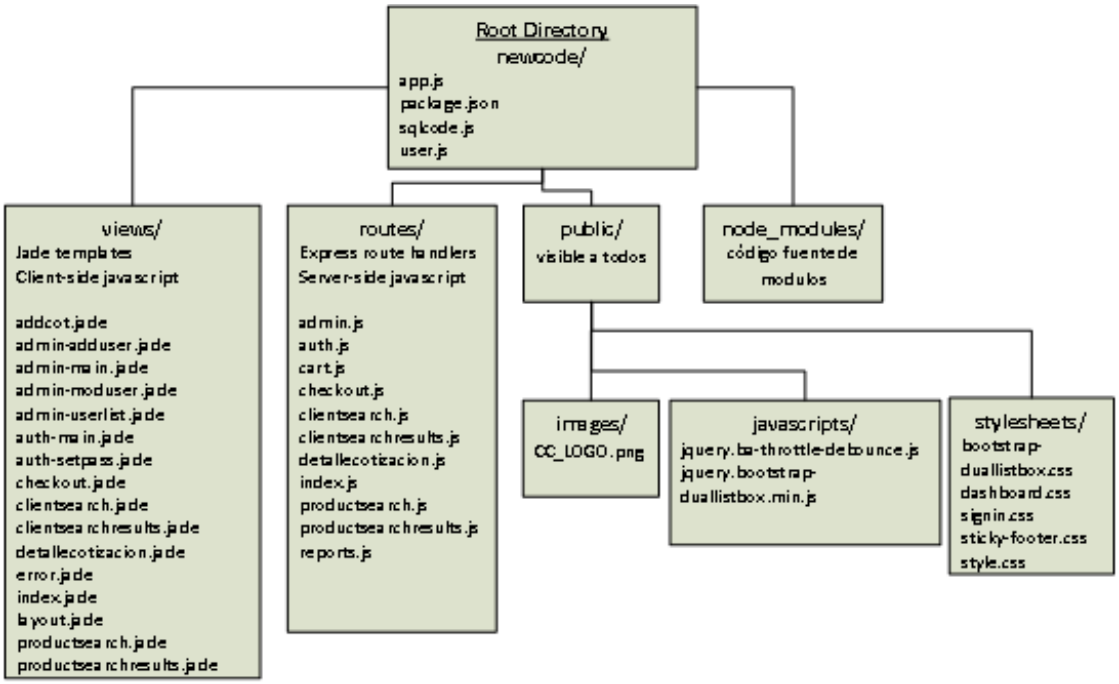


Figura 36 - Estructura del árbol de código fuente

9.1.1 La estructura de un proyecto NodeJS

Todos los proyectos NodeJS tienen un archivo llamado `package.json` que guarda varios detalles del proyecto tales como su nombre, la versión, la descripción, la licencia, el punto de arranque y las dependencias. Una dependencia es una relación entre un proyecto y los módulos sin los cuales se tendría que implementar funcionalidad que ya está disponible. Este mecanismo de proyectos y dependencias permiten a los ingenieros en software reutilizar código que ya ha sido desarrollado para enfocarse mejor en las metas de los proyectos. Con el package.json bien configurado correctamente, el proyecto puede ser instalado con solo un llamado del comando "`npm install`" desde la línea de comando ya sea en UNIX (Linux, BSD, Mac OSX, etc.) o Microsoft Windows.

Un buen ejemplo de esto es la dependencia de nuestro proyecto del módulo "node-mssql." Este módulo provee un API para interactuar con gestores de bases de datos de tipo Microsoft SQL. Con este módulo el ingeniero puede despreocuparse por los detalles de la interacción de NodeJS con el SGBD (Sistema Gestor de Base Datos) y operar bajo la suposición de que se va a poder acceder y manipular los datos tal como sea necesario dentro de la aplicación.

```
{
  "name": "newcode",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "async": "~2.5.0",
    "bcrypt": "^1.0.2",
    "bluebird": "^3.4.7",
    "body-parser": "~1.15.2",
    "cookie-parser": "~1.4.3",
    "debug": "~2.2.0",
    "express": "~4.14.0",
    "express-session": "~1.15.1",
    "jade": "1.11.0",
    "jade-bootstrap": "1.0.9",
    "jsdoc": "^3.5.4",
    "moment": "^2.18.1",
    "morgan": "~1.7.0",
    "mssql": "~3.3.0",
    "serve-favicon": "~2.3.0",
    "sqlite3": "^3.1.8"
  },
  "main": "app.js",
  "devDependencies": {},
  "repository": {
    "type": "git",
    "url": "ssh://hubert@tropez.org:/home/hubert/repos/coticross.git"
  },
  "keywords": [],
  "author": "Hubert Cross III",
  "license": "ISC",
  "description": "Sistema de Apoyo a Ventas para Casa Cross"
}
```

Figura 37 - El *package.json* de SAV

En los archivos .JADE se encuentran las plantillas que se utilizan para generar el HTML que se presenta al navegador. También contienen el JavaScript que se ejecuta al lado de navegador (client-side javascript) para las validaciones de campos y la manipulación del DOM.

Tabla 20 - Archivos de vista JADE

Archivo de vista JADE	Descripción
admin-adduser.jade	Página de creación de usuario
admin-main.jade	Página de panel administrativo
admin-moduser.jade	Página de modificación de usuario existente
admin-userlist.jade	Página de listado de usuarios
auth-main.jade	Página de ingreso de nombre de usuario y contraseña
auth-setpass.jade	Página para restablecer contraseña
checkout.jade	Página de especificación de precios, cantidades, descuentos
clientsearch.jade	Página de búsqueda de cliente
clientsearchresults.jade	Página de muestra de resultados de búsqueda de cliente
detallecotizacion.jade	Página en cual se muestran todos los detalles de la cotización antes de guardar
error.jade	Página de error
index.jade	Pagina de bienvenida
layout.jade	Plantilla reutilizable que contiene el encabezado (*header*) y el pie de página (*footer*)
productsearch.jade	Formulario de búsqueda de productos
productsearchresults.jade	Página de muestra de resultados de búsqueda de productos
reports-cotlist.jade	Página de mostrar reportes de cotizaciones
reports-vendedor.jade	Panel de reportes de vendedor

Tabla 21 - Archivos de controlador

Archivo de ruta EXPRESS	Descripción
admin.js	Funciones administrativas
auth.js	Gestión de usuarios
cart.js	Registro de detalles de cotizaciones en proceso
checkout.js	Servicio de guardar cotización
clientsearch.js	Servicio para servir formulario de búsqueda de clientes
clientsearchresults.js	Gestión de la búsqueda y procesamiento de resultados
detallecotizacion.js	Funciones para presentar la cotización lista para registrar
index.js	Servicio de *renderizar* la página de bienvenida
productsearch.js	Servicios para servir formulario de búsqueda de productos
productsearchresults.js	Gestión de la búsqueda y procesamiento de resultados
reports.js	Gestión de reportes

Tabla 22 - Archivos de modelo

API entre Node y SCM	Descripción
sqlcode.js	Contiene el API que permite registrar, actualizar y consultar datos en SQLITE y MSSQL
user.js	Contiene el API solamente para la autenticación y gestión de usuarios en SQLITE

9.1.1 Creación de usuario

Para crear un usuario se ha implementado un servicio bajo la ruta /admin que abarca todas las operaciones administrativas. Bajo esta ruta se ofrece “adduser” siendo el URL completo /admin/adduser para acceder a esta funcionalidad. El navegador primero realiza un GET a /admin/adduser. Este servicio renderiza la plantilla JADE “admin-adduser” que contiene el formulario de creación de usuario.

```
521 router.get('/adduser', misc.requireLogin, function(req, res) {
522     var bodegas = [];
523     var supervisors = [];
524     sqlcode.getSupervisorUsers(db)
525         .then(function(rows) {
526             for(var i = 0; i < rows.length; i++) {
527                 supervisors[supervisors.length] = [rows[i]['iduser'], rows[i]['username']];
528             }
529             sqlcode.getAllBodegasFromSQLITE(db)
530                 .then(function(rows) {
531                     for (var i = 0; i < rows.length; i++ ) {
532                         bodegas[bodegas.length] = rows[i]['idbodega'] + "_" + rows[i]['bodeganame'];
533                     }
534                     res.render('admin-adduser', {
535                         title: 'Creacion de usuario',
536                         userjson: req.session.userjson,
537                         bodegalist: bodegas,
538                         supervisorlist: supervisors
```

Figura 38 - Servicio HTTP GET de creación de usuario

La función res.render toma dos argumentos críticos. El primero es el nombre de la plantilla que se va a renderizar y el segundo es un objeto JSON con datos claves que alimentaran dicha plantilla. Abajo se detalla la plantilla admin-adduser.jade con énfasis en los datos que transmite el servidor node a la plantilla y por extensión al navegador:

(**bodegalist** es la lista de bodegas y **supervisorlist** es la lista de supervisores.)

```
223     #listbox
224       +bodegaJade(bodegalist)
225   +whitespace(10)
226   +whitespace(10)
227   .container-center

262   .container-center
263     .row
264       .col
265         #inputComboBoxSupervisor
266           | Elegir supervisor del vendedor:
267         +combobox2("formInputSupervisorComboBox", supervisorlist)
268
```

Figura 39 - Representación de controles en plantilla JADE

A continuación se muestra el resultado final de esta ejecución en la página de creación de usuario. Se indican los dos componentes relevantes, el DualListBox y el ComboBox con las bodegas disponibles y asignadas y los supervisores, respectivamente.

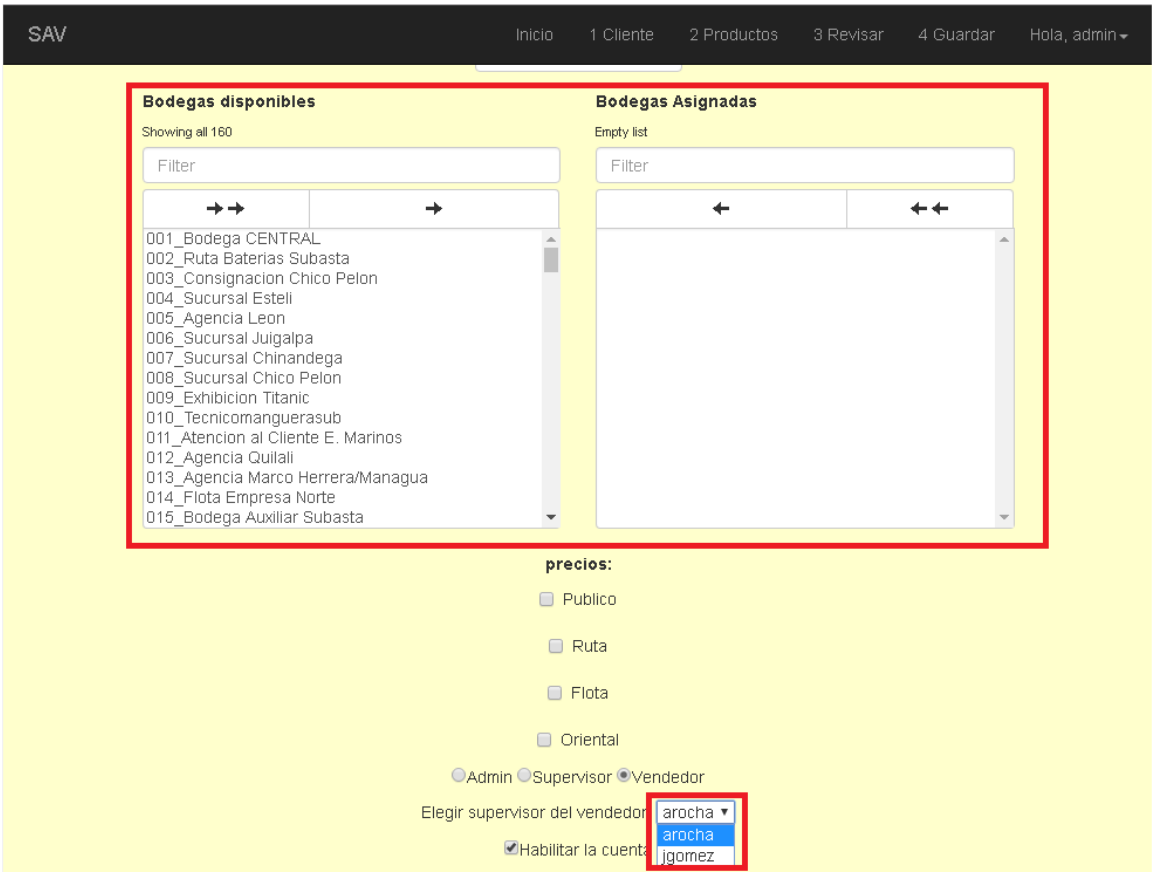


Figura 40 - Presentación de controles en interfaz gráfico

Luego de llenar los campos exigidos en el formulario de creación de usuario, el administrador da *click* en el botón de aceptación. En este momento se envía una petición HTTP de tipo POST a la ruta /admin/adduser. En el cuerpo de la petición (req.body) van los datos que se ingresaron en el formulario y pueden ser accedidos en el servidor.

```
418     /**
419      * Takes user creation parameters from AJAX does SQLITE insert.
420      * /
421      * Toma parametros de creacion de usuario desde AJAX y realiza una insercion en SQLITE
422      * @module /admin
423     */
424 router.post('/adduser', misc.requireLogin, function(req, res) {
425     var db = req.db;
426     const username          =   req.body.formInputUsername;
427     const password          =   req.body.formInputPassword1;
428     const fullname          =   req.body.formInputFullname;
429     const email             =   req.body.formInputEmailAddress;
430     const codigoVendedor    =   req.body.formInputCodigoVendedor;
431     const serie             =   req.body.formInputSerie;
432     const codsuc            =   req.body.formInputCodSuc;
433     var enabled;
434     const authlevel         =   req.body.authlevel;
```

Figura 41 - Servicio HTTP POST de creación de usuario

Más abajo dentro del mismo servicio de /admin/adduser, la contraseña pasa por dos procesos conocidos como *salting* y *hashing* con bcrypt, una librería basada en el codificador Blowfish.(Schneier, 1994, pp. 38-40) Una vez que solo se cuenta con el hash de una cadena de caracteres ya desconocida, solamente se puede verificar si una cadena corresponde a ella a través del algoritmo de validación.

```
474     const saltRounds = 10;
475     bcrypt.hash(password, saltRounds, function(err, hash) {
476         var paramobj =
477         {
478             username : username,
479             hash : hash,
480             fullname : fullname,
481             email: email,
482             supervisor: supervisor,
483             codigoVendedor : codigoVendedor,
484             idauth : authlevel,
485             bodegas : cleanBodegas,
486             prices : prices,
487             serie : serie,
488             codsuc : codsuc,
489             enabled : enabled
490         }
491         user.createUserEntry(db, paramobj)
492             .then(function() {
493                 user.assignUserPrices(db, paramobj)
494                     .then(function() {
495                         if (cleanBodegas.length > 0) {
496                             user.assignUserBodegas(db, paramobj)
497                                 .then(function() {
498                                     res.render('admin-main',
499                                     {
500                                         title: 'Admin Panel',
501                                         userjson: req.session.userjson,
502                                         success: true,
503                                         successuser: paramobj['username']
504                                     });
505                                 })
506                         }
507                         else {
508                             res.render('admin-main',
509                             {
510                                 title: 'Admin Panel',
511                                 userjson: req.session.userjson,
512                                 success: true,
513                                 successuser: paramobj['username']
514                             });
515                         }
516                     })
```

Figura 42 - Proceso de creación de usuario

Una vez creado el usuario se renderiza la pantalla del panel administrativo con una mensaje de confirmación y la cuenta de usuario se puede ser utilizada.

9.1.2 Gestión de procesos asíncronos a través de Promises y SQL Transactions

Uno de los mayores retos en la programación de servicios y sistemas web es la programación **asíncrona**. En la paradigma convencional de software, el programador puede escribir su código con confianza en que este se ejecutará en secuencia en la aplicación. Esta es la programación **síncrona**. En node.js es necesario manejar el hecho de que las peticiones enviadas a servicios externos como los SGBD (Sistema Gestor de Base Datos) no tienen garantía de ser atendidas en un tiempo determinado o del todo en caso de error. Por tanto existe el concepto de las **promesas o "Promise"** por su nomenclatura original en inglés.

La promesa es análoga a una función tal que mientras que una función retorna o falla, una promesa se "resuelve" o se "rechaza" (*resolve* y *reject* son los términos que aparecen en la documentación original). Esto permite gestionar el flujo de ejecución de los programas. Así se puede encadenar varias promesas para que sean ejecutadas en secuencialmente, con un bloque de código esperando que el anterior concluya antes de ser ejecutado. El API que nos permite trabajar con Microsoft SQL Server, "mssql", ya viene "promisificado" (promisified). Esto significa que su conjunto de funciones ya devuelven promesas y pueden ser encadenados en una serie de bloques "then" y "catch" (*then* significando "luego").

```
function sendCotizacion(db, userName, productsObjArray, totalsObj, clientData, res, req) {
    const pool = new sql.connect(config, err => {
        console.log("err: " + err)
    });
    const transaction = new sql.Transaction(pool);

    transaction.begin(err => {
        if (err) {
            console.log("err" + err);
        }
        var rolledBack = false;
        transaction.on('rollback', function(aborted) {
            rolledBack = true;
        })
        //const request = new sql.Request(transaction)

    var numRegObj = { numreg : "", ActConsecutivo: "" };
    var exchangeRateAndDateTime;
    var parametersObject = {};
    var returnMessage;
    var returnNumReg;
    var returnActCon;

    return sqlcode.NumRegUnico(numRegObj, transaction)
    .catch( function (err) {
        misc.timestamp("Caught rejection: " + err)
        throw "throwing copper";
    })
    .then( function() { //begin then1
        returnNumReg = numRegObj['numreg'];
        misc.timestamp(userName + " sqlcode.GetExchangeRateDateAndTimeFromMSSQL")
        return sqlcode.GetExchangeRateDateAndTimeFromMSSQL(transaction)
        .catch( function (err) {
            misc.timestamp("Caught rejection: " + err)
            throw "throwing copper";
        })
        .then(function(recordsets) {
        console.log("recordsets2: " + JSON.stringify(recordsets[0]) );
        exchangeRateAndDateTime = recordsets[0][0];
    })
    .catch( function (err) {
        misc.timestamp("Caught rejection: " + err)
```

Figura 43 - Transacción SQL y el inicio de una cadena de promesas

La figura anterior y la siguiente demuestran, respectivamente, el inicio tanto de una transacción SQL y una cadena de promesas como la finalización de la transacción en un *rollback* o *commit* al terminar la ejecución en éxito o fracaso.

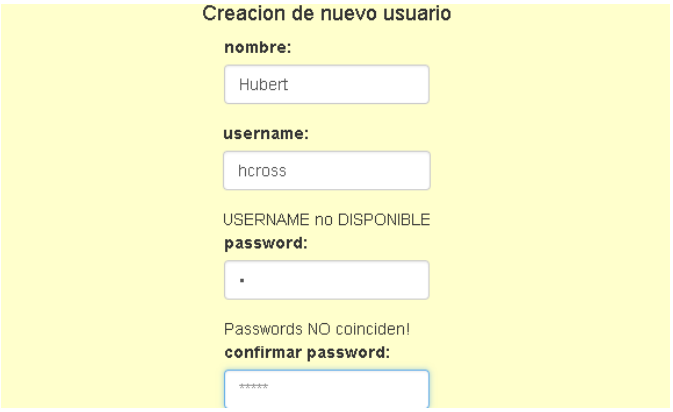
```
        .catch( function (err) {
            misc.timestamp("Caught rejection7: " + err)
            returnMessage = "ERROR";
            console.log("rolledback: " + rolledBack);
            if (!rolledBack) {
                console.log("Rolling back");
                transaction.rollback(function(err) {
                    if (err) {
                        console.log("rollback error: " + err)
                    }
                    else {
                        console.log("rollback success");
                    }
                })
            }
        })
        .then( function() { // end then8, begin then9
            misc.timestamp(userName + " Done saving cotizacion data " + returnMessage);
            if (req.session.userjson) {
                console.log("req.session.userjson defined");
            }
            if (req.session.userjson && ( returnMessage != "ERROR" )   )    {
                console.log("balls to the walls")
                req.session.userjson.cid = undefined;
                req.session.userjson.cname = undefined;
                returnMessage = "TODOBIEN";
                transaction.commit(function(err) {
                    if (err) { console.log("commit error: " + err)}
                })
            }
            res.send(
            {
                message : returnMessage,
                numreg : returnNumReg,
                act: returnActCon
            });
        })

    }) // end then4
    // }) // end then2
}) // end then1

}) // transaction begin
```

Figura 44 - Finalización de cadena de promesas en rollback o commit

Podemos ver que en el último bloque catch se maneja cualquier error o rechazo de promesa con un *rollback*. Al no suceder ningún error o rechazo se ejecuta un *commit* antes de proceder a *renderizar* la página de aceptación con los números consecutivos del documento de la cotización.

9.1.3 Validación de campos en tiempo real (VTR)

La siguiente captura de pantalla de muestra un ejemplo de la validación en tiempo real (VTR). Algunos campos críticos requieren ser validados de esta manera para evitar duplicación de datos tales como los nombres de usuario. Con la VTR se evita la necesidad de que el usuario haga múltiples intentos de enviar datos en la creación de una cuenta de usuario. Esto se logra a través del manejo de eventos en el navegador web con jQuery y AJAX. El código fuente pertinente se muestra junto con el interfaz de usuario a cual corresponde.




```
// debounce permite regular la tasa de peticiones enviadas al servidor
// debounce throttles the rate at which requests are sent to the server
$('#inputUsername input').keyup( $.debounce( 800, ajaxCheckUserExists ) );

/* Revisar si un dado username ya esta ocupado
   Check if a given username is already taken */
function ajaxCheckUserExists() {
  const username = $('#inputUsername input').val();
  console.log("Username to check availability: " + username)
 $.ajax({
  url: '/admin/checkusername',
  type: 'POST',
  contentType: 'application/json',
  data: JSON.stringify({
  uname: username
  }),
  success: function(returndata) {
   console.log("success ajax");
   console.log("ugh " + JSON.stringify(returndata).replace(/(^")|("$)/g, ''));
   if (JSON.stringify(returndata).replace(/(^")|("$)/g, '') == "USERNAME AVAILABLE") {
    $('#inputUsernameAlert').html("USERNAME DISPONIBLE");
    validateUsername = true;
    validateSubmit();
   }
   else {
    $('#inputUsernameAlert').html("USERNAME no DISPONIBLE");
    validateUsername = false;
```

Figura 45 - Validación de campos en tiempo real

9.1.4 Publicación iterativa

Uno de los objetivos específicos del proyecto fue la implementación de manera iterativa del sistema. La primera versión funcional de SAV se aplicó de manera acelerada en Marzo 2017 en una emergencia ocasionada por el fallo repentino del servidor en cual residía el interfaz web anterior (Véase antecedentes). Las subsecuentes liberaciones se vinieron dando conforme se alcanzaban los hitos de desarrollo a lo largo del proyecto.

Tabla 23 - Plan de publicación de iteraciones

Iteración	Liberación	Características	Limitaciones
MARK I	Marzo 2017	• Autenticación y gestión de precios y series de usuarios con tablas existentes del interfaz web anterior • Registro de cotizaciones	• Complejidad en la creación de cuentas de usuario • No llevaba encriptación • Modelo de navegación muy rígido • Pocos parámetros para las búsquedas de productos y clientes
MARK II	Mayo 2017	• Mejora en interfaz de usuario • Más flexibilidad en modelo de navegación • Más campos de búsqueda de clientes y productos	• Bitácora muy limitada • Errores
MARK III	Agosto 2017	• Autenticación y gestión de usuarios propia en base datos local • Encriptación de contraseñas	• Falta de reportes
MARK IV	Septiembre 2017	• Encriptación total • Reportes	

9.2 Pruebas del software

En este acápite se presenta una muestra del uso de recursos por parte del servidor web en Linux. Después se plasman las pruebas de aceptación y seguridad. Abajo se puede observar la eficiencia de node, que opera con solamente 110 Megabytes de memoria.

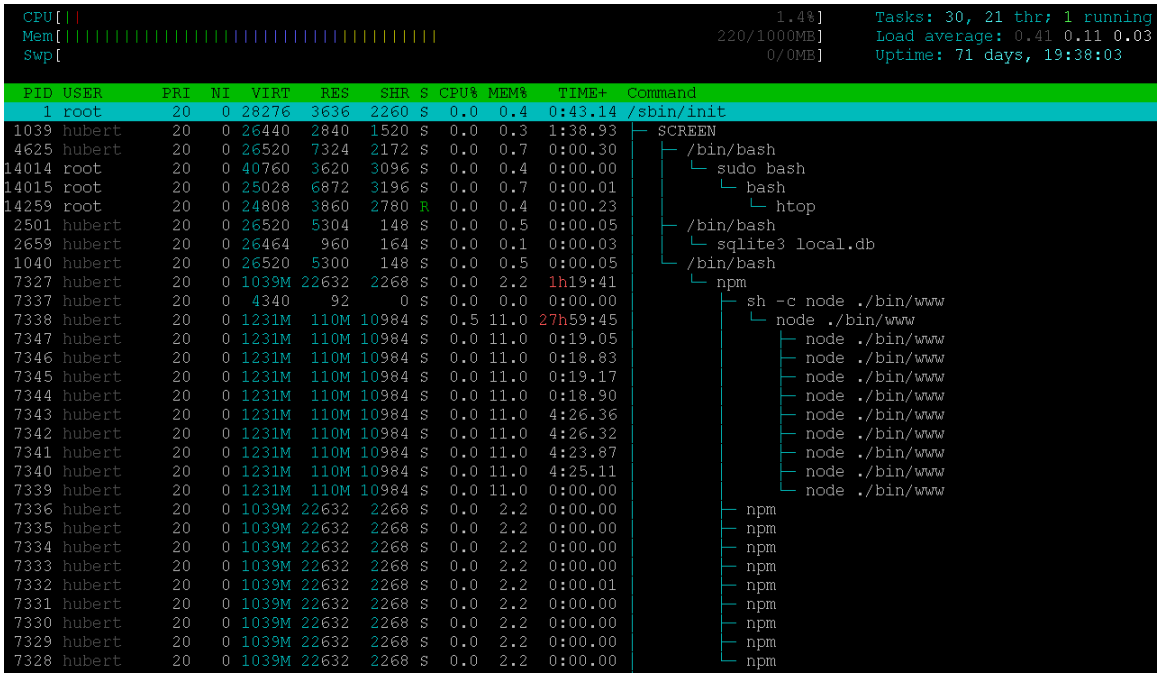


Figura 46 - Muestra de uso de recursos computacionales

9.2.1 Pruebas de aceptación del sistema

ID	CPA-01
Nombre	Creación de usuario
Fecha	03-AGO-2017
Servicio a probar	POST /admin/adduser
Objetivo	Comprobar funcionamiento de servicio de registro de usuario
Descripción: Se construyó una solicitud HTTP de tipo POST con los parámetros necesarios para efectuar el registro de un usuario nuevo	
Criterios de éxito:	Respuesta HTTP positiva
Criterios de falla:	Respuesta HTTP negativa
Perfil del usuario	Administrador
Precondiciones: 1. Usuario autentificado en el sistema. 2. Entrar al formulario de registrar un nuevo usuario. 3. Ingresar la información requerida en el formulario.	
Flujo Normal: 1. El usuario inicia sesión en el sistema. 2. Se dirige al módulo de los usuarios, da *click* en el botón Crear Usuario. 3. El usuario es redirigido al formulario para crear una nueva cuenta de usuario. 4. Ingresa la información solicitada. 5. Presiona el botón Crear Usuario	
Pos condiciones: Un nuevo usuario es creado en la base de datos exitosamente.	

Figura 47 - CPA-01 Creación de usuario

ID	CPA-02
Nombre	Modificación de usuario
Fecha	06-AGO-2017
Servicio a probar	POST /admin/moduser
Objetivo	Comprobar funcionamiento de servicio de modificación de usuario
Descripción: Se construyó una solicitud HTTP de tipo POST con los parámetros necesarios para modificar el registro de un usuario existente	
Criterios de éxito:	Respuesta HTTP positiva
Criterios de falla:	Respuesta HTTP negativa
Perfil del usuario	Administrador
Precondiciones: 4. Usuario autentificado en el sistema. 5. Entrar al formulario de registrar un nuevo usuario. 6. Ingresar la información requerida en el formulario.	
Flujo Normal: 6. El usuario inicia sesión en el sistema. 7. Se dirige al módulo de los usuarios, da *click* en el botón Modificar Usuario. 8. El usuario es redirigido al formulario para modificar una cuenta de usuario. 9. Ingresa la información solicitada. 10. Presiona el botón Guardar Cambios	
Pos condiciones: Un usuario es modificado en la base de datos exitosamente.	

Figura 48 - CPA-02 Modificación de usuario

ID	CPA-03
Nombre	Búsqueda de productos
Fecha	15-JUN-2017
Servicio a probar	POST /productsearchresults
Objetivo	Comprobar funcionamiento de búsqueda de productos
Descripción: Se construyó una solicitud HTTP de tipo POST con los parámetros necesarios para realizar una búsqueda de producto	
Criterios de éxito:	Respuesta HTTP positiva
Criterios de falla:	Respuesta HTTP negativa
Perfil del usuario	Administrador
Precondiciones: 1. Usuario autentificado en el sistema. 2. Entrar al formulario de búsqueda de productos. 3. Ingresar la información requerida en el formulario.	
Flujo Normal: 1. El usuario inicia sesión en el sistema. 2. Se dirige al módulo de productos. 3. El usuario es redirigido al formulario de parámetros de búsqueda de producto. 4. El usuario ingresa sus parámetros. 5. Presiona el botón Buscar	
Pos condiciones: Se presentan los resultados de la búsqueda.	

Figura 49 - CPA-03 Búsqueda de productos

ID	CPA-03
Nombre	Búsqueda de clientes
Fecha	15-JUN-2017
Servicio a probar	POST /clientsearchresults
Objetivo	Comprobar funcionamiento de búsqueda de productos
Descripción: Se construyó una solicitud HTTP de tipo POST con los parámetros necesarios para realizar una búsqueda de cliente	
Criterios de éxito:	Respuesta HTTP positiva
Criterios de falla:	Respuesta HTTP negativa
Perfil del usuario	Administrador
Precondiciones: 1. Usuario autentificado en el sistema. 2. Entrar al formulario de búsqueda de clientes. 3. Ingresar la información requerida en el formulario.	
Flujo Normal: 1. El usuario inicia sesión en el sistema. 2. Se dirige al módulo de clientes. 3. El usuario es redirigido al formulario de parámetros de búsqueda de cliente. 4. El usuario ingresa sus parámetros. 5. Presiona el botón Buscar	
Pos condiciones: Se presentan los resultados de la búsqueda.	

Figura 50 - CPA-03 Búsqueda de clientes

ID	CPA-04
Nombre	Registro de cotización
Fecha	29-MAR-2017
Servicio a probar	POST /save
Objetivo	Comprobar correcto registro de cotización
Descripción: Se construyó una solicitud HTTP de tipo POST con los parámetros necesarios para realizar el registro de una cotización en SCM	
Criterios de éxito:	Respuesta HTTP positiva
Criterios de falla:	Respuesta HTTP negativa
Perfil del usuario	Administrador
Precondiciones: 1. Usuario autentificado en el sistema. 2. Hay un cliente elegido y hay productos agregados 3. El usuario tiene permiso de registrar cotizaciones.	
Flujo Normal: 1. El usuario inicia sesión en el sistema. 2. Se dirige al módulo de guardar. 3. El usuario da una ultima revisión a los productos y cliente antes de guardar 4. Presiona el botón Guardar	
Pos condiciones: Se presentan una aletra indicando que la cotización de guardo bien, y se envía un correo electrónico a la dirección de correo electrónico del usuario.	

Figura 51 - CPA-04 Registro de cotización

ID	CPA-05
Nombre	Generación de reporte de cotizaciones
Fecha	12-SEP-2017
Servicio a probar	POST /reports/vmc
Objetivo	Comprobar funcionamiento de generación de reporte de cotizaciones
Descripción: Se construyó una solicitud HTTP de tipo POST con los parámetros necesarios para generar el reporte	
Criterios de éxito:	Respuesta HTTP positiva
Criterios de falla:	Respuesta HTTP negativa
Perfil del usuario	Administrador
Precondiciones: 1. Usuario autentificado en el sistema. 2. Navegar a panel de reportes 3. Ingresar la información requerida en el formulario.	
Flujo Normal: 1. El usuario inicia sesión en el sistema. 2. Se dirige al módulo de reportes. 3. El usuario es redirigido al formulario de parámetros 4. El usuario ingresa sus parámetros. 5. Presiona el botón Visualizar	
Pos condiciones: Se presentan el reporte de las cotizaciones	

Figura 52 - CPA-05 Reporte de cotizaciones

9.2.2 pruebas de seguridad de acceso al sistema

ID	CPS-01
Nombre	Iniciar sesión
Fecha	26/05/2017
Objetivo	Detectar fallas en la seguridad de acceso al sistema de los diferentes tipos de usuarios
Descripción: Se probó el nivel de seguridad que tenía el formulario de inicio de sesión de los diferentes tipos de usuarios.	
Criterios de éxito:	Mensaje de error “Acceso denegado. Datos incorrectos”
Criterios de falla:	Acceso al sistema con datos incorrectos.
Perfil del usuario	Sin autentificar
Precondiciones: 1. Acceder a la dirección url del sistema.	
Flujo Normal: 1. El usuario accede a la dirección url del sistema. 2. Ingresa a los formularios de iniciar sesión del personal administrativo, docente y estudiante. 3. Ingresa datos no válidos en los campos de usuario y contraseña. 4. Presiona el botón acceder.	
Pos condiciones: El sistema no permite el acceso del usuario, debido a ingreso de datos no válidos.	

Figura 53 - CPS-01 Iniciar sesión

ID	CPS-02
Nombre	Prueba de intercepción de datos
Fecha	27/09/2017
Objetivo	Comprobar que los datos en tránsito no pueden ser interpretados para conseguir acceso no autorizado al sistema
Descripción: Se tomó una muestra del tráfico entre el navegador y el servidor WEB al momento de ingresar al sistema.	
Criterios de éxito:	No poder discernir los credenciales del usuario
Criterios de falla:	Poder identificar los credenciales del usuario
Perfil del usuario	Sin autentificar
Precondiciones: 1. Acceder a la dirección url del sistema. 2. Ingresar usuario y contraseña. 3. "Escuchar" en el puerto TCP en cual corre la aplicación web (3000 en el caso de texto plano, 8443 en el caso encriptado)	
Flujo Normal: 1. El usuario accede a la dirección url del sistema. 2. Ingresa un nombre de usuario y contraseña 3. El analista ejecuta el comando "tcpdump" para poder visualizar el tráfico TCP en tiempo real 4. Presiona el botón acceder. 5. El tráfico se observa en tiempo real y no puede ser comprendido porque esta encifrado	
Pos condiciones: No se accede a los credenciales del usuario.	

Figura 54 - CPS-02 Prueba de intercepción de datos

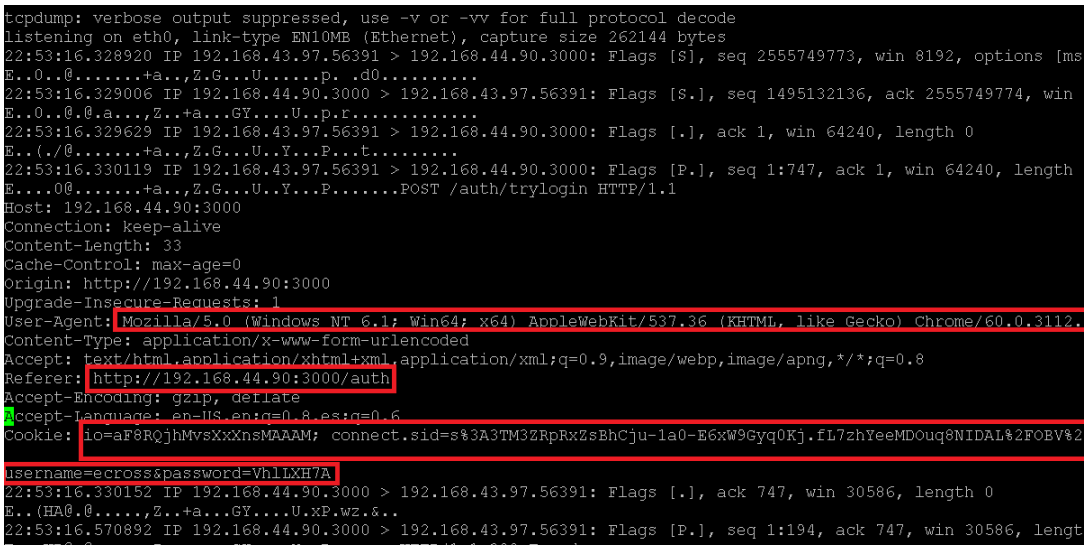


Figura 55 - Muestra de transmisión de datos en texto plano

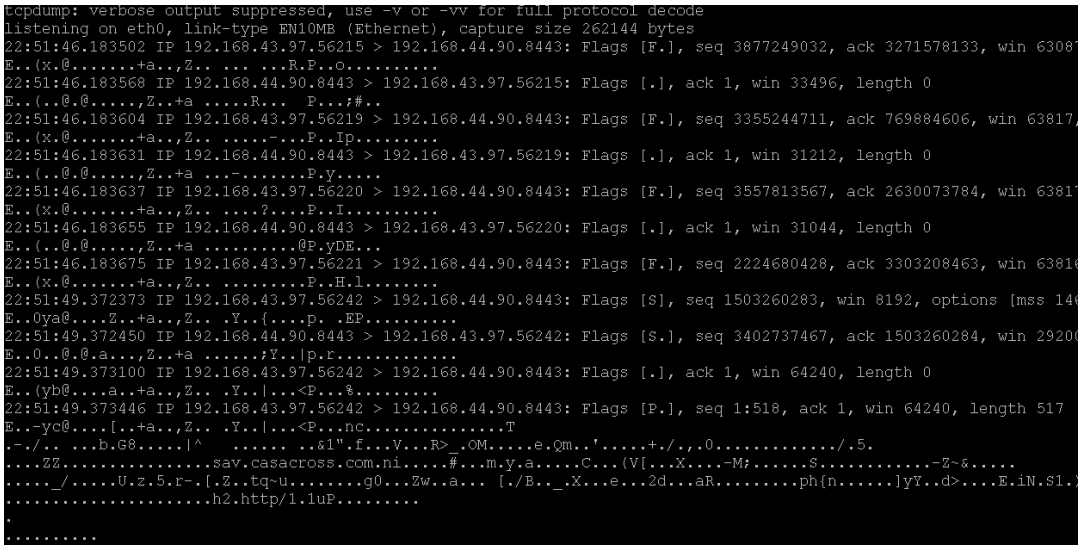


Figura 56 - Muestra de transmisión de datos encriptados

En el caso de la prueba de seguridad CPS-2, Podemos observar en Figura 55 que sin el uso de la encriptación, los credenciales del usuario pueden ser leídos por un tercero que tenga acceso a la ruta entre el navegador y el servidor. Esto significa un riesgo enorme de seguridad pues con el nombre de usuario y contraseña cualquier persona podría acceder al sistema y por tanto a datos sensibles de la empresa.

Para mitigar este riesgo se adquirió un certificado SSL de una autoridad de certificación (Certificate Authority o CA en inglés). A través de estos archivos, esta entidad certifica que un dado *hostname* corresponde a una organización específica, en este caso sav.casacross.com.ni a Casa Cross. Esto permite utilizar el protocolo HTTPS (HTTP encriptado) para la conexión entre el navegador y un servicio web. Con esta misma tecnología funcionan los sitios de comercio electrónico tales como eBay, Amazon, etc.

10. Conclusiones y Recomendaciones

En este último acápite se hace referencia directa a cada uno de los objetivos establecidos para acertar que se han logrado todos. El diagnóstico del sistema antecedente dio resultados que se pueden ver en Apéndice B – Diagnóstico de interfaz web anterior y sirvieron para alimentar el proceso de análisis. El análisis de las necesidades de la empresa y los interesados se detalla en 7. Capítulo II: Análisis del Sistema.

Los resultados del análisis se utilizaron para alimentar el proceso de diseño. Se elaboraron un modelo relacional y un interfaz gráfico y se eligieron las tecnologías más adecuadas para obtener la mejor entrada posible para el proceso de implementación. La implementación se ejecutó de manera iterativa como fue estipulado, cada vez recibiendo la retroalimentación de los usuarios para mejorar la siguiente versión del software.

Los casos de prueba fueron creados en base a los requerimientos funcionales. Los dos tipos de prueba fueron aceptación y seguridad y se realizaron correctamente. También se capacitaron los usuarios a través de reuniones en grupo y con video-manuales para dar una alternativa ligera a la documentación formal que igual se preparó y ofreció. Por último se puso en marcha la cuarta iteración del software en el mes de septiembre del 2017 ya con todos los requerimientos funcionales satisfechos.

En cuanto a las recomendaciones, para mantener el nivel actual de seguridad se debe mantener al día el certificado SSL para poder continuar utilizando encriptación en la transmisión de datos y así mitigar el riesgo de acceso no autorizado al sistema. La recomendación más fuerte es aprovechar la base de código fuente establecida en este proyecto para ir paulatinamente implementando módulos del sistema heredado (“SCM”) en una plataforma más segura, moderna y robusta tal como es node.js y así seguir impulsando a la empresa hacia el horizonte tecnológico para que continúe su historia de éxitos.

11. Bibliografía

A. Ginige, S. M. (2001). Guest editors' introduction: The essence of web engineering-managing the diversity and complexity of web. *IEEE MultiMedia 8* , 22-25.

Alonso, D. (24 de Enero de 2017). *Debian Introduction.* Recuperado el 6 de Marzo de 2017, de Debian - The Universal Operating System: https://wiki.debian.org/es/DebianIntroduction

Ambler, S. W. (2014). *UML 2 Use Case Diagrams: An Agile Introduction*. Recuperado el 28 de Febrero de 2017, de Agile Modeling: http://www.agilemodeling.com/artifacts/useCaseDiagram.htm

Basulto, S. (27 de Noviembre de 2011). *Tutorial de Git en Español*. Recuperado el 18 de Junio de 2016, de Santiago Basulto: http://blog.santiagobasulto.com.ar/programacion/2011/11/27/tutorial-de-git-en-espanol.html#que-es-git

Brown, P. (17 de Enero de 2017). *State of the Union: npm*. Recuperado el 28 de Septiembre de 2017, de Linux.com: https://www.linux.com/news/event/Nodejs/2016/state-union-npm

Bruegge, B. D. (2009). *Object-Oriented Software Engineering Using UML, Patterns, and Java: Solutions to Exercises* (3da ed. ed.). Pearson.

Casa Cross. (15 de 1 de 2017). Recuperado el 30 de 1 de 2017, de Casa Cross: http://www.casacross.com.ni/index.php/quienes-somos

Chen, L., Ali Babar, M., & Nuseibeh, B. (2013). Characterizing Architecturally Significant Requirements. *IEEE Software , 30* (2).

Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. (P. Baxendale, Ed.) *Communications of the ACM , 13* (6), 377-387.

Conallen, J. (1999). Modeling web application architectures with UML. *Communications of the ACM 42* , 63-70.

Dayani-Fard, H. e. (1999). *Legacy Software Systems: Issues, Progress, and Challenges.* IBM.

Debian Project. (20 de Junio de 2016). *Debian History.* Recuperado el 6 de Marzo de 2017, de Debian: https://wiki.debian.org/es/DebianHistory

Encyclopædia Britannica. (2016). *database*. Obtenido de Encyclopedia Britannica: http://www.britannica.com/technology/database

Facebook, Inc. (2017). *Thinking in React*. Recuperado el 10 de Marzo de 2017, de ReactJS: https://facebook.github.io/react/docs/thinking-in-react.html

FIQ, C. (2010). INSTRUCTIVO PARA LA REALIZACIÓN DEL TRABAJO DE DIPLOMA EN LA FACULTAD DE INGENIERÍA QUÍMICA. Managua, Nicaragua: FACULTAD DE INGENIERÍA QUÍMICA, UNIVERSIDAD NACIONAL DE INGENIERÍA.

Freelancer.com. (2017). *Freelancer*. Recuperado el 4 de Septiembre de 2017, de Freelancer: https://www.freelancer.com/projects/php/build-website-15086209/

Google. (5 de Junio de 2007). *Google Security Blog*. Recuperado el 2 de Septiembre de 2017, de Google: https://security.googleblog.com/2007/06/web-server-software-and-malware.html

Griffith, E. (13 de Mayo de 2016). *What Is Cloud Computing?* . Recuperado el 23 de Febrero de 2017, de PC Magazine: http://www.pcmag.com/article2/0,2817,2372163,00.asp

Isocron. (2016). *¿Qué es Open Source?* Recuperado el 21 de 3 de 2016, de Isocron Systems: https://www.isocron.net/node/35

Jeffries, R. (11 de Noviembre de 2005). *Extreme Programming for One*. Recuperado el 7 de Febrero de 2017, de WikiWikiWeb: http://xp.c2.com/ExtremeProgrammingForOne.html

jQuery Foundation. (s.f.). *jQuery*. Recuperado el 1 de Enero de 2017, de jQuery: http://jquery.com/

Laudon, K., & Laudon, J. (2012). *Sistemas de Información Gerencial* (Decimosegundo ed.). (A. Romero, Trad.) México, México: Pearson.

M. Winckler, P. P. (2003). *StateWebCharts: A formal description technique dedicated to navigation modelling of web applications, in: Proc.* Berlin: Springer.

Microsoft. (2016). *Herramientas para desarrolladores de Microsoft*. Recuperado el 3 de July de 2016, de Visual Studio: https://www.visualstudio.com/es-es

Microsoft IIS version 7.0: Security vulnerabilities. (s.f.). Recuperado el 2017 de Febrero de 26, de CVE Details. The ultimate security vulnerability datasource: https://www.cvedetails.com/vulnerability-list/vendor_id-26/product_id-3436/version_id-52075/Microsoft-IIS-7.0.html

Microsoft. (2017). *Product Lifecycle*. Recuperado el 2 de 9 de 2017, de Microsoft Support: https://support.microsoft.com/en-us/lifecycle/search/1163

Murdock, I. A. (6 de Enero de 1994). *El manifiesto de Debian Linux.* Recuperado el 6 de Marzo de 2017, de Una breve historia de Debian : https://www.debian.org/doc/manuals/project-history/ap-manifesto.es.html

Navarro, A. F.-V.-M. (2008). Characterizing navigation maps for web applications with the NMM approach. *Science of Computer Programming* , 1-16.

Nayab, N. (20 de Septiembre de 2010). *Advantages of Extreme Programming*. Recuperado el 2 de Marzo de 2017, de Bright Hub Project Management: http://www.brighthubpm.com/methods-strategies/87839-advantages-of-extreme-programming/

NETMARKETSHARE. (2016). *Desktop Browser Version Market Share*. Obtenido de Market Share Statistics for Internet Technologies: https://www.netmarketshare.com/browser-market-share.aspx?qprid=2&qpcustomd=0

Node.js Foundation. (s.f.). *About Node.js*. Recuperado el 1 de Enero de 2017, de Node.js: https://nodejs.org/en/about/

Node.js Foundation. (s.f.). *Express - Infraestructura de aplicaciones web Node.js*. Recuperado el 30 de Enero de 2017, de Express JS: http://expressjs.com/es/

Oracle Corporation. (20 de mayo de 2016). *MySQL 5.7 Reference Manual*. Recuperado el 11 de 2 de 2016, de MySQL: http://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

Otto, M., & Thornton, J. (2014). *Bootstrap 3, el manual oficial*. Obtenido de http://librosweb.es/libro/bootstrap_3/

Pressman, R. S. (2005). *Ingeniería del Software, 6ta ed.* New York: McGraw Hill.

Pressman, R. S. (2002). *Ingeniería del Software; Un Enfoque Práctico V ed.* Madrid: McGraw-Hill.

Pug Project. (30 de Enero de 2017). *GitHub - pugjs/pug - robust, elegant, feature rich template engine for Node.js*. Recuperado el 31 de Enero de 2017, de GitHub: https://github.com/pugjs/pug

Regalado, A. (31 de Octubre de 2011). Who Coined 'Cloud Computing'? *MIT Technology Review* .

Rouse, M. (April de 2014). *What Amazon EC2 Instance?* Recuperado el 3 de March de 2016, de TechTarget: http://searchaws.techtarget.com/definition/Amazon-EC2-instances

Schneier, B. (1994). The Blowfish Encryption Algorithm. *Dr. Dobb's Journal* , 38-40.

SQLite. (s.f.). *SQLite Home Page*. Recuperado el 7 de Febrero de 2017, de SQLite: http://sqlite.org/

Turek, L. (27 de June de 2007). *Linuxsoft.cz*. Recuperado el 3 de August de 2016, de Interview: Bram Moolenaar: http://www.linuxsoft.cz/article.php?id_article=1477

Wells, D. (1999). *Acceptance Tests*. Recuperado el 10 de Marzo de 2017, de Extreme Programming: http://www.extremeprogramming.org/rules/functionaltests.html

WikiBooks. (s.f.). *Introduction to Software Engineering/Process/Life Cycle*. Recuperado el 7 de Febrero de 2017, de WikiBooks: https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Process/Life_Cycle

Zwass, V. (27 de December de 2011). *Information System*. Recuperado el 2 de 1 de 2016, de Encyclopedia Britannica: http://www.britannica.com/topic/information-system

12. Anexos

Apéndice A – Glosario

ERP: Planificación de Recursos Empresariales.

SCM: Sistema de Control Máximo. Es una aplicación implementada en Visual Basic 6.0 alrededor del año 2000 y adquirido por Casa Cross en el 2003.

AJAX: Siglas inglesas de “Asynchronous Javascript And XML”. Es un método para intercambiar datos con un servidor remoto para actualizar partes del DOM del navegador web sin refrescar la página entera.

SAV: Siglas de Sistema de Apoyo a Ventas.

Software *Legacy*: Pressman lo traduce como “software heredado.” Es cualquier software que se considera de una generación anterior independiente de que esté en uso o no.

SGBD: Sistema Gestor de Base de Datos. DBMS en inglés. Un software que permite manipular el contenido de una base de datos.

VB6: Abreviatura de Visual Basic 6.0, un lenguaje de programación de Microsoft que tuvo auge en los años 90. Se considera totalmente obsoleto desde que se publicó la plataforma .NET. Importante porque SCM está desarrollado en él.

MVC: Siglas de Modelo-Vista-Controlador. La arquitectura de software que emplea SAV en cual los componentes del sistema se separan en la lógica de procesamiento, la visualización y el manejo y registro de los datos.

BSD: Una re-implementación libre y gratuita de UNIX de la universidad UC Berkeley. Ha formado la base tanto para sistemas operativos libres tales como FreeBSD y NetBSD como comerciales como OS X de Apple.

Linux: Una re-implementación libre y gratuita de UNIX iniciada por Linus Torvalds.

UNIX: Sistema operativo propietario desarrollado en los años 70 por Bell Labs. Histórico e importante porque en él se implementó TCP/IP y se formaron los conceptos de redes, servicios, etc.

DOM: Siglas inglesas de Document Object Model, o Modelo de Objeto de Documento. Es una estructura de datos que representa los objetos existentes en una página que se muestra en el navegador. Efectivamente, modificar el DOM modifica lo que el navegador presenta ya sea visible o invisible al usuario.

Promise: Es un método de manejar procesos asíncronos sin bloquear la ejecución del software.

VTR: Validación en Tiempo Real. Una característica que depende del uso de AJAX para validar campos que tiene restricciones que se deben revisar al lado del servidor (*server-side*) sin enviar formularios y recargar páginas web.

RUC: Registro Único de Contribuyente. Es el código que utiliza el estado para distinguir a las empresas y controlar los impuestos que deben. El RUC debe estar presente en documentos oficiales tales como facturas y devoluciones.

Apéndice B – Diagnóstico de interfaz web anterior

El interfaz web anterior tenía varias características que reducían su efectividad. Las plataformas de software y sistema operativo sobre cuales fue construido eran problemáticas debido a su alto grado de inseguridad y la configuración incorrecta de las herramientas. Además no presenta señas de que se siguió alguna metodología coherente de ingeniería de software. En seguida se profundiza sobre estos problemas.

Por ejemplo, el modelo de autenticación para acceder a la base datos SCM desde el servicio web era SQL. Esto en sí no es problemático, pero cuando este usuario tiene acceso completo a la base datos de producción debido a llevar los roles “db_owner”, “db_reader” y “db_writer”, la base datos que es el corazón de la empresa se expone a abusos masivos. Según el modelo de roles en Microsoft SQL Server, el rol db_owner permite ejecutar respaldos, restaurar respaldos, eliminar, modificar y crear tablas, vistas, procedimiento almacenados, y más.

Los riesgos se aumentan cuando se toma en cuenta que el equipo en que residía el antiguo interfaz web tenía asignado un IP público accesible directamente desde el internet. No había ningún firewall para filtrar el acceso a otros puertos TCP en los cuales este equipo estaba escuchando. Agravando aun más la situación, todo esto estaba montado sobre Windows Server 2008 R2, sistema operativo cuyo línea de soporte fue declarada EoL en el año 2015 (End of Life o Fin de Vida) por Microsoft. Esto significa que el fabricante no publicaría más actualizaciones o correcciones de seguridad. (Microsoft, 2017)

Con respecto al propio servidor web IIS, este se ha destacado por su pésimo record de seguridad. Un estudio de Google Security encontró que aunque IIS solo tuviera una cuota del mercado (*marketshare*) del 23% en servidores web en la Web, representaba el 49% de los servidores web que ejecutaban malware o código malicioso. (Google, 2007)

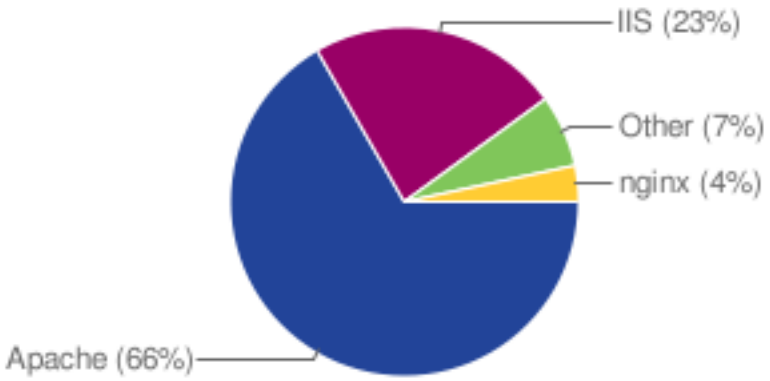


Figura 57 - Cuotas de mercado de servidores web(Google, 2007)

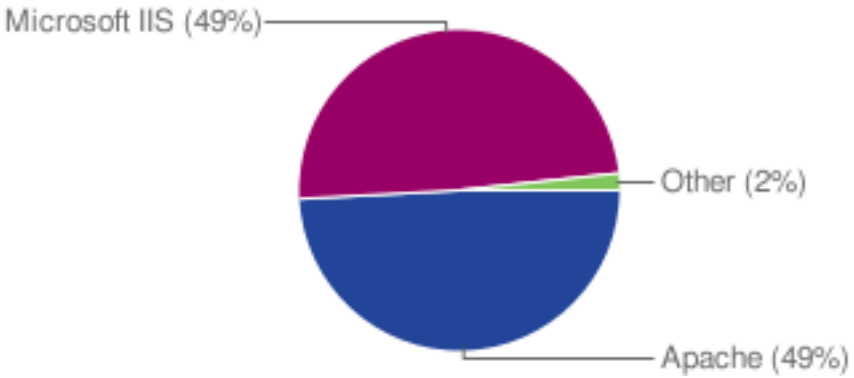


Figura 58 - Distribución de servidores web que ejecutan *malware*(Google, 2007)

Finalmente, se incluye una lista de cotejo que muestra los faltantes básicos del antiguo “cotizador web”.

Tabla 24 - Diagnóstico de interfaz web antecedente

Característica	**Presente**
Encriptación de datos en transmisión	NO
Interfaz web *responsive*	NO
Gestión de usuarios a través de interfaz web	NO
Usuario puede cambiar su propia contraseña	NO
Código fuente comentariado	NO
Persistencia de sesiones contra desconexiones	NO

Apéndice C – Diccionario de datos de tablas de cotizaciones en SCM

dbo.[Cotizacion Tarj Desc Doc]			Sistema: SCM
COLUMNA	**TIPO**	**NULL**	**DESCRIPCION**
NUM_REG	int	NO	PK - Numero de registro único
Cod_Emp	nvarchar(2)	SI	Codigo de la empresa al que corresponde el documento, siempre '03' para Casa Cross
COD_SUC	nvarchar(3)	SI	Codigo de bodega relacionada al documento
COD_DIA	nvarchar(8)	SI	Serie del documento
NUM_DOC	nvarchar(7)	SI	Numero del documento
TIP_DOC	tinyint	SI	Tipo de documento,
COD_ID	nvarchar(7)	SI	
DESC_DOC	nvarchar(100)	SI	
FECHA	datetime	SI	Fecha de registro del documento
HORA	datetime	SI	Hora de registro del documento
TIPO_CAMB	float	SI	Tipo de cambio
COD_US	nvarchar(10)	SI	Codigo de Usuario SCM que registra el documento
ESTADO	int	SI	
NUM_REF	nvarchar(50)	SI	En caso de ser solicitud, codigo de solicitud
cod_vend	nvarchar(5)	SI	Codigo de vendedor
NUM_REG_CW	int	SI	Numero de registro de Cotizacion Web (precursor a Cotizacion Con Solicitud de Mercaderia)

dbo.[Cotizacion Venta]		Sistema: SCM	
COLUMNA	**TIPO**	**NULL**	**DESCRIPCION**
NUM_REG	int	NO	PK - Numero de registro único
PLAZO	smallint	NO	
COD_VEND	nvarchar(5)	NO	
NUM_ORD	nvarchar(20)	SI	
MONEDA	tinyint	NO	
TIP_DESC	tinyint	NO	
TIP_EXO	tinyint	NO	
COD_ZON	tinyint	NO	
DIRECCION	ntext	SI	
TELEFONO	nvarchar(15)	SI	
FAX	nvarchar(15)	SI	
OBSERVACIONES	ntext	SI	
VAL_PORC_DESC	float	NO	
VAL_PORC_DESCS	float	SI	
VAL_PORC_IGV	float	NO	
VAL_PORC_IEC	float	NO	
IR	tinyint	NO	
VAL_IR	float	NO	
VALOR_BONIF	float	NO	
COSTO_BONIF	float	NO	
COSTO	float	NO	
DESCUENTO	float	NO	
DESCUENTOS	float	SI	
SUBTOTAL	float	NO	
IGV	float	NO	
IEC	float	NO	
VALOR	float	NO	
tip_vent	tinyint	SI	

Cotizacion Tarjeta de Kardex		Sistema: SCM	
COLUMNA	**TIPO**	**NULL**	**DESCRIPCION**
NUM_REG	int	SI	
COD_SUC	nvarchar(3)	NO	
NUM_LIN	float	NO	
COD_PROD	nvarchar(25)	NO	
CANTIDAD	float	NO	Cantidad del producto a cotizar
COSTO	float	NO	Costo promedio en córdobas
Unidades	float	NO	Siempre 1 en cotizaciones
Cod_Prec	tinyint	NO	El tipo de precio según segmento de mercado en cual se cotiza el producto (Publico, Oriental, Ruta, Flota)
VAL_ORIG	float	NO	El precio sin descuento
VALOR	float	NO	El precio que especifica el usuario
VAL_PORC_DESC	float	NO	El porcentaje de descuento
VAL_NETO	float	NO	El precio neto
BONIFICACION	nvarchar(1)	NO	Vacio para cotizaciones
VAL_ORG_IGV	float	NO	Porcentaje de impuesto, siempre 15 en SAV
VAL_PORC_IGV	float	NO	Porcentaje de impuesto, siempre 15 en SAV
EXIST_ANT	float	NO	No aplica en cotizaciones, siempre 1
COSTO_ANT	float	NO	No aplica en cotizaciones, siempre 0
COSTO_D	float	NO	Costo del producto en USD
TIPO	tinyint	NO	No aplica en cotizaciones, siempre 0
COD_MEC	nvarchar(3)	NO	No aplica a cotizaciones, siempre cadena vacía
CANTIDAD_SOLICITADA	float	NO	No aplica a cotizaciones, siempre cadena vacía
COSTO1	float	NO	No aplica a cotizaciones, siempre cadena vacía

dbo.[Facturacion Clientes]		Sistema: SCM	
COLUMNA	**TIPO**	**NULL**	**DESCRIPCION**
Cod_Emp	nvarchar(2)	NO	Codigo de empresa - para gestion multiempresarial
COD_ID	nvarchar(7)	NO	PK - Codigo unico de cliente
NOMBRE	nvarchar(100)	NO	Nombre del cliente
DIRECCION	ntext	SI	Direccion del cliente
CIUDAD	nvarchar(15)	SI	Ciudad del cliente
NUM_RUC	nvarchar(15)	SI	RUC en caso de ser sociedad
PLAZO	tinyint	SI	Plazo en caso de cliente de credito
LIMITE	float	SI	Limite de credito
LIMITE_D	float	SI	Limite en dolares

dbo.[Facturacion Productos]		Sistema: SCM	
COLUMNA	**TIPO**	**NULL**	**DESCRIPCION**
COD_PROD	nvarchar(25)	NO	Codigo unico del producto
COD_ORIG	nvarchar(25)	SI	Codigo OEM del producto
NOM_PROD	nvarchar(60)	NO	Nombre del producto
COD_GRUP	nvarchar(10)	NO	No aplica a cotizaciones
COD_LIN	nvarchar(10)	NO	Codigo de linea del producto
MARCA	nvarchar(20)	SI	Marca del producto
P_COS_C	float	NO	Costo del producto en NIO
P_COS_D	float	NO	Costo del producto en USD
COS_PROM_C	float	NO	Costo promedio en NIO
COS_PROM_D	float	NO	Costo promedio en USD
APLICACION	nvarchar(80)	SI	Aplicación del producto en caso de repuestos

Apéndice D – Diccionario de datos de tablas en SAV

Tabla: USER			**Sistema: SAV**	
Propósito de la tabla:			Registrar datos de usuarios que acceden al sistema	
Atributo	**Tipo**	**Null**	**Llave (REF)**	**Descripción**
iduser	integer	no	PK	id único
username	text	no	Unique	
hash	text	no		hash de password
fullname	text	no		
supervisor	integer	no	FK USER (iduser)	id del supervisor
idauth	integer	no	FK AUTHLEVEL (idauth)	nivel de autorización
email	text	no		
codigovendedor	text	no		código vendedor en SCM
enabled	integer	no		habilitado o no
serie	text	no		serie con cual cotiza
codsuc	text	no		bodega SCM principal del usuario

Tabla: SHOPPINGLIST			**Sistema: SAV**	
Propósito de la tabla:			Registrar datos de la cotización que está en proceso	
Atributo	**Tipo**	**Null**	**Llave (REF)**	**Descripción**
cotwebusername	text	no	PK Compuesta	id único
clientid	text	no		
codalterno	text	no		código alterno del producto – este sirve de PK en las tablas de SCM
codoriginal	text	no		OEM del producto
preciosistema	real	no		precio que el sistema proporciona
preciousuario	real	no		precio que el usuario digita
qty	integer	no		cantidad a cotizar
descripcion	text	no		nombre del producto
p_cos_d	real	no		costo en USD en SCM
cos_prom_c	real	no		costo promedio NIO en SCM
pricetype	integer	no		tipo de precio con que el producto fue agregado a la cotización
descuento	real			porcentaje de descuento que el usuario indica
marca	text			marca del producto en SCM

Tabla: CURRENTCLIENT			Sistema: SAV	
Propósito de la tabla:			Registra datos del cliente que cada usuario tiene seleccionado en el momento. Reduce la cantidad de consultas que SAV necesita hacer con SCM	
Atributo	**Tipo**	**Null**	**Llave (REF)**	**Descripción**
cotwebusername	text	no	PK Compuesta	id único
clientid	text	no		
clientname	text	no		nombre del cliente
clientaddress	text	no		direccion del cliente
clientcity	real	no		ciudad del cliente
clientplazo	integer	no		plazo al ser cliente de credito
clientcreditlimit	real	no		limite al ser cliente de credito
clientnamechange	integer	no		1 si el cliente es tramite ruta para exigir especificación de nombre en GUI

Tabla: USER_PRICE			Sistema: SAV	
Propósito de la tabla:			Asociar usuarios y los precios a cuales tienen acceso	
Atributo	**Tipo**	**Null**	**Llave (REF)**	**Descripción**
username	text	no	PK, FK USER (username)	id único
idprice	integer	no	PK, FK PRICE (idprice)	
enabled	integer	no		habilitado/no

Tabla: USER_BODEGA			Sistema: SAV	
Propósito de la tabla:			Asociar usuarios y las bodegas con cuales se consultan existencias	
Atributo	**Tipo**	**Null**	**Llave (REF)**	**Descripción**
username	text	no	PK, FK USER (username)	id único
idbodega	text	no	PK, FK BODEGA (idbodega)	
enabled	integer	no		habilitado o no

Tabla: AUTH_OPER			Sistema: SAV	
Propósito de la tabla:			Asociar niveles de autorizacion con operaciones especificas	
Atributo	**Tipo**	**Null**	**Llave (REF)**	**Descripción**
idauth	integer	no	PK, FK AUTHLEVEL (idauth)	id único
idoper	integer	no	PK, FK OPERATION (idoper)	
enabled	integer	no		habilitado o no

Tabla: PRICE			Sistema: SAV	
Propósito de la tabla:			Mantiene registro de los precios de la empresa: Público, Ruta, Flota, Oriental	
Atributo	**Tipo**	**Null**	**Llave (REF)**	**Descripción**
idprice	integer	no	PK	id único
pricename	TEXT	no		

Tabla: BODEGA			Sistema: SAV	
Propósito de la tabla:			Mantiene registro de las bodegas en SCM	
Atributo	**Tipo**	**Null**	**Llave (REF)**	**Descripción**
idbodega	integer	no	PK	id único
bodeganame	TEXT	no		nombre de la bodega

Tabla: OPERATION			Sistema: SAV	
Propósito de la tabla:			Mantiene registro de las operaciones que el sistema ofrece	
Atributo	**Tipo**	**Null**	**Llave (REF)**	**Descripción**
idoper	integer	no	PK	id único
opname	TEXT	no		nombre de la operación

Apéndice E - Procedimientos Almacenados MSSQL

PA1 - uspActConsecutivo

Incrementa el consecutivo del *código de diario* y el *tipo de diario* (también conocidos por "Serie" y "Tipo de Documento") especificado. Este procedimiento ya existía en SCM y se reutiliza en SAV.

Parámetros:

Name	Type	Description
@mCodDia	NVARCHAR(8)	Codigo de Diario
@mTipDia	NVARCHAR(1)	Tipo de Diario

PA2 - uspGetClient

Consulta la tabla SCM dbo.[Facturacion Clientes]

Parámetros:

Name	Type	Description
@p_ClientName	nvarchar (100)	Nombre del cliente
@p_ClientID	nvarchar(7)	ID del cliente
@p_RUC	nvarchar(15)	Registro Único de Contribuyente
@p_HASCREDIT	int	Booleano – 1 de crédito, 0 contado

PA3 - dbo.uspGetExchangeRateDateAndTimeFromMSSQL

Hace cosas varias. Consulta la tabla dbo.[Cofiguracion de POS] (SIC) y consigue el tipo de cambio Cordoba:Dólar. También consigue la hora y fecha del servidor MSSQL. Todo esto se ocupa en grabar los registros de las cotizaciones.

Parámetros: Ningunos

PA4 - uspGetNonExcludedBodegas

Consulta las tablas dbo.[Facturacion Bodegas] y dbo.[NEW_BODEGAS_EXCLUIDAS] para determinar las bodegas que se deben tomar en cuenta en al cálculo de existencias de productos.

Parámetros: Ningunos

PA4 - dbo.uspInsertCTDD

Registra los datos de la cotización pertinentes en la tabla dbo.[Cotizacion Tarjeta Desc Documento]. Reutilizada de SCM.

Parametros:

Nombre	Tipo	Descripción
@mNumero	**Int**	**Numero de Registro Unico**
@mCodEmp	**Nvarchar(2)**	**Codigo de Empresa**
@mCodSuc	**Nvarchar(3)**	**Codigo Sucursal(Bodega)**
@mDiario	**Nvarchar(8)**	**Serie**
@mCodGuardar	**Nvarchar(7)**	
@mCodId	**Nvarchar(7)**	
@mDescDoc	**Nvarchar(100)**	**Descripcion del documento**
@mTipDoc	**Tinyint**	**Tipo de documento**
@mFecha	**Nvarchar(50)**	**Fecha de tramite**
@mHora	**Nvarchar(50)**	**Hora de tramite**
@mTipoCamb	**Float**	**Tipo de cambio**
@mCodigoVendedor	**Nvarchar(5)**	**Codigo del Vendedor**

PA5 - dbo.uspInsertCV

Registra los datos de la cotización pertinentes en la tabla dbo.[Cotizacion Ventas]. Reutilizada de SCM.

Parametros:

Nombre	Tipo	Descripción
@mNumero	**Int**	**Número de Registro Unico**
@mPlazo	**Nvarchar(2)**	**Plazo de crédito**
@mDireccion	**Nvarchar(3)**	**Direccion de entrega**
@mObservaciones	**Nvarchar(8)**	**Observaciones del usuario**
@mCodVend	**Nvarchar(7)**	**Codigo del vendedor**
@mCosto	**Nvarchar(7)**	**Costo total de inventario**
@mDescuento	**Nvarchar(100)**	**Monto de descuento total**
@mSubtotal	**Tinyint**	**Subtotal del documento**
@mIgv	**Nvarchar(50)**	**Monto de impuesto total**
@mValor	**Nvarchar(50)**	**Valor total del documento**

PA6 - dbo.uspSearchProduct_GetStockByBodega

Consulta la existencia de un producto en una bodega especifica. Se utiliza en la búsqueda de productos para conocer la existencia en cada bodega física.

Parametros: Ningunos

Nombre	Tipo	Descripción
@ALTERNO	**Nvarchar(50)**	**Codigo del producto**
@BODEGA	**Nvarchar(3)**	**Codigo de bodega**

PA7 - dbo.uspSearchProduct

Consulta la existencia de un producto en una bodega especifica. Se utiliza en la búsqueda de productos para conocer la existencia en cada bodega física.

Parametros: Ningunos

Nombre	Tipo	Descripción
@ALTERNO	**Nvarchar(50)**	**Código interno del producto**
@ORIGINAL	**Nvarchar(50)**	**Código de fabricante**
@DESCRIPCION	**Nvarchar(50)**	**Descripción del producto**
@APLICACION	**Nvarchar(50)**	**Aplicación del producto**
@PRICETYPE	**Int**	**Tipo de precio a consultar**
@MINSTOCK	**Int**	**Filtrar por existencia o no**
@SUCS	**Nvarchar(4000)**	**Bodegas a consultar**
@COD_LIN	**Nvarchar(10)**	**Código de línea de producto**

Apéndice F – Manual de usuario

F.1 Ingreso y Egreso del usuario y el menú principal

Figura 61 - Portal de ingreso

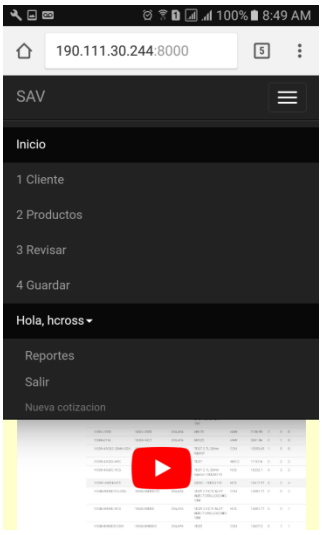


Figura 60 - Menú principal

Figura 59 - Página principal

La página de ingreso o *login* es estándar y básica. En cuanto el usuario ha ingresado credenciales legítimas al formulario de ingreso, se presentará la página de bienvenida. Esta página presenta el logotipo de la empresa e información sobre las actualizaciones de la versión más reciente del sistema, además un video YouTube en cual se explica audiovisual el uso de la herramienta.

El menú principal se encuentra en la parte superior de la pantalla. Para visualizar el menú, dar click en el icono de despliegue en la esquina superior derecha del interfaz. Se presentarán las siguientes opciones en un menú jerárquico:

- Inicio
- Cliente
- Productos
- Revisar
- Guardar
- Hola, <Usuario> [Submenú]
  - Reportes
  - Salir
  - Nueva Cotización

En los siguientes acápites, se explicará el uso de cada uno de estos ítems.

F.2 Clientes

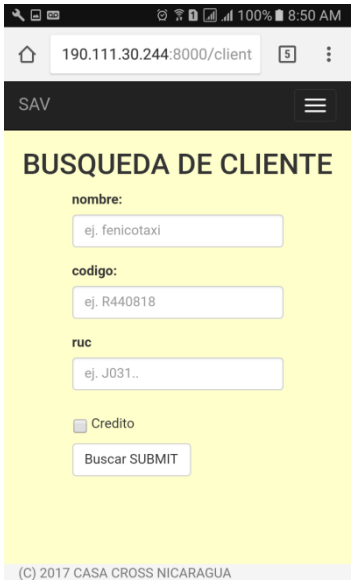


Figura 62 - Formulario de consulta de clientes

Para consultar la cartera de clientes, el usuario debe elegir “Cliente” del menú principal. Se presentará el formulario de consulta o búsqueda de clientes. Aquí habrá al menos 3 campos y 1 *checkbox* con cuales el usuario puede moderar el nivel de detalle de la búsqueda:

Campo	Significado
Nombre	El nombre del cliente ya sea persona natural o jurídica. En el caso de las personas jurídicas es el nombre comercial o la razón social.
Código del cliente	Código único del cliente asignado por el área de cartera en el caso de clientes de crédito. Se crea de manera dinámica en el segmento de mostrador en las tiendas.
RUC	RUC en caso de las personas jurídicas
Checkbox de crédito	Si este control está cotejado, solamente se mostrarán clientes que tienen líneas de crédito

F.3 Elegir clientes y clientes con trámite de ruta

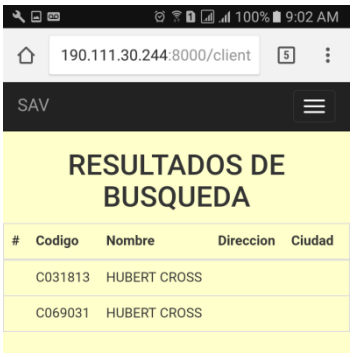


Figura 64 - Resultados de consulta de clientes

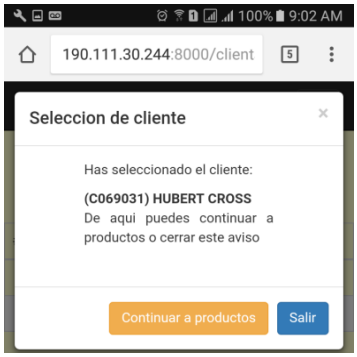


Figura 63 - Confirmación de selección de cliente

Una vez que el usuario ha ingresado datos en el formulario de búsqueda, debe dar *click* en el botón Buscar. Los resultados se presentarán en la siguiente página, “Resultados de Búsqueda”. Los resultados incluyen las siguientes columnas:

Columna en resultados	significado
#	Espacio vacío para mejora de presentación
Código	
Nombre	
Dirección	
Ciudad	
Plazo	Plazo de pago en caso de cliente de crédito
Limite	Limite de crédito en caso de cliente de crédito.

Aquí puede elegir el cliente al cual corresponderá la cotización. Solamente es necesario dar *click* en la fila del cliente que desea elegir y se presentará un *modal* confirmando la selección exitosa del cliente. El mismo *modal* ofrece ir directamente al módulo de productos (normalmente el siguiente paso después de elegir un cliente) o salir para permitir al usuario ir a otra área del sitio.

Cambiar de cliente

Mientras el usuario tenga un cliente elegido, se presentará un mensaje indicando el cliente actual en la parte superior de la página debajo del menú. Esta banda también ofrece “Cambiar el cliente” que es básicamente un enlace para volver al formulario de consulta de clientes.

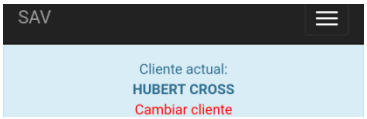


Figura 65 - Notificación de cliente actual y opción de cambiar

F.4 Productos

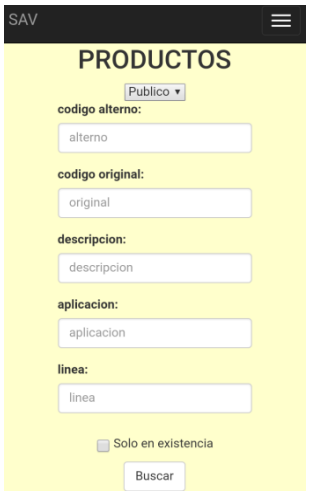


Figura 66 - Formulario de consulta de productos

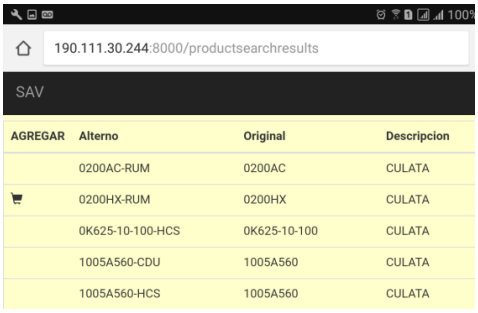


Figura 67 - Resultados de consulta de productos

El módulo de productos permite consultar productos por los siguientes criterios:

Parámetro	Significado
código “alterno”	El identificador único de un producto en la empresa
código “original”	El código que el fabricante asigna al producto
descripción	El nombre del producto
Aplicación	El o los vehículos, motores o sistemas con cual(es) es compatible el producto
línea	La línea de productos
Solo en existencia	Filtrar productos agotados

Al enviar los parámetros de búsqueda se presenta una tabla de resultados con las siguientes columnas adicionales a los parámetros de búsqueda:

Columna	
AGREGAR	Espacio en el cual se puede dar *click* para agregar el producto a la cotización
Precio	
CA	Existencia en bodega principal (“Centro de Almacenamiento”
CARS	Existencia en la bodega de la tienda del vendedor (Centro de Atención y Reabastecimiento)
TODO	Existencia en toda la empresa
Detalle	Se puede dar *click* en esta columna para conseguir un detalle de la existencia del producto en todas las bodegas

Figura 69 - Productos / Ventanilla de negociación

Figura 68 - Productos / Detalle de existencias

Mejores prácticas para buscar productos

Una mejor práctica en la búsqueda de productos es filtrar por existencia e buscar por “alterno”. Un código alterno puede tener muchos “originales” entre cuales hay mejor chance de hallar un repuesto compatible y en existencia. Otra práctica recomendable es consultar las existencias detalladas para saber dónde hay existencia más cerca al cliente.

Agregar productos a cotización

Si el vendedor decide agregar el producto a la cotización dando click en la columna AGREGAR, inmediatamente se presenta la ventanilla de negociaciones mostrada en Figura 69 - Productos / Ventanilla de negociación Aquí el usuario puede negociar con el cliente e ingresar el precio, descuento y la cantidad acordada. También puede apreciar el cálculo de precio con descuento, el IVA total y el precio “neto” utilizando la jerga del área de ventas. Por último puede elegir entre buscar otro producto o salir e ir a cualquier otra área de la aplicación.

Eliminar productos de la cotización

Para eliminar un producto de la cotización en proceso, se puede navegar al área de Guardar Cotización y dar click en la fila “ELIM” que corresponde el producto a descartar.

F.5 Revisión o Reinició de Cotización

Reiniciar la cotización

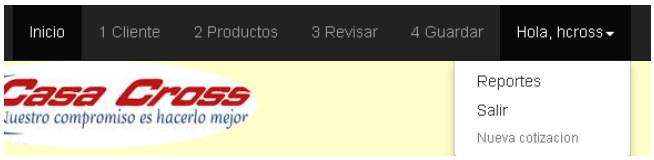


Figura 70 - Ubicación en menú de reinicio de cotización

Si el usuario decide descartar la cotización, puede ir al submenú y elegir “Nueva cotización”. Se presentará un *modal* confirmando si está seguro, y si el usuario confirma se eliminarán todos los datos del documento actual y se volverá a la página principal.

Cambiar precios/cantidades

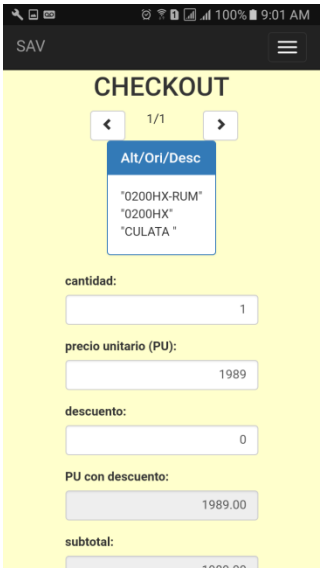
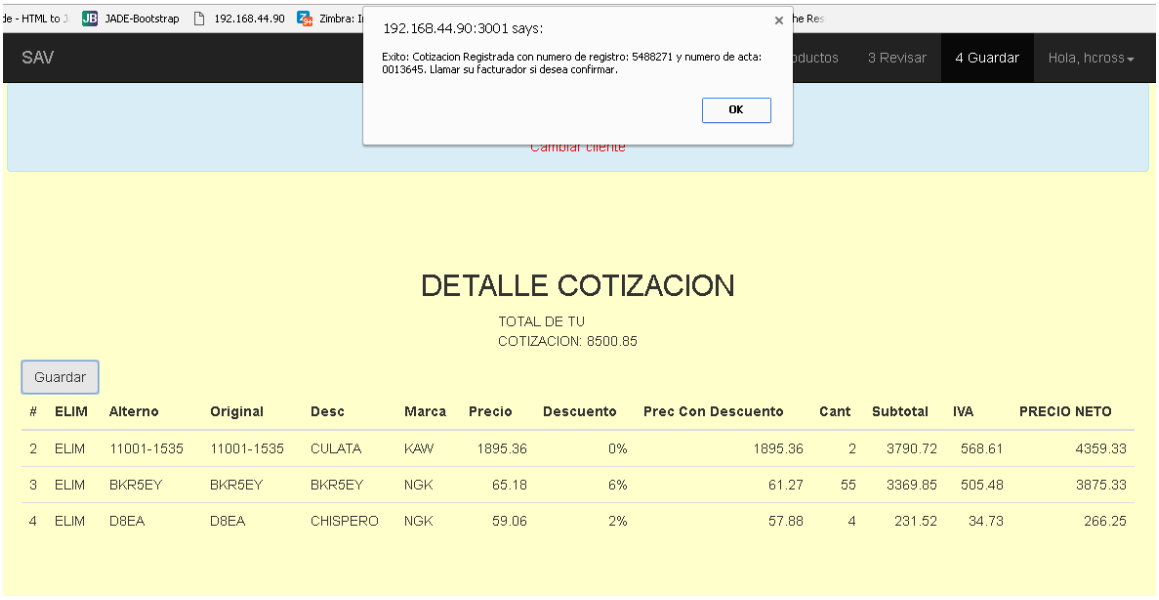


Figura 71 - Revisión de precios, descuentos y cantidades

En el área de “Revisión” el usuario puede repasar por cada ítem y editar los precios, cantidades y descuentos.

F.6 Registro de Cotización



#	ELIM	Alterno	Original	Desc	Marca	Precio	Descuento	Prec Con Descuento	Cant	Subtotal	IVA	PRECIO NETO
2	ELIM	11001-1535	11001-1535	CULATA	KAW	1895.36	0%	1895.36	2	3790.72	568.61	4359.33
3	ELIM	BKR5EY	BKR5EY	BKR5EY	NGK	65.18	6%	61.27	55	3369.85	505.48	3875.33
4	ELIM	D8EA	D8EA	CHISPERO	NGK	59.06	2%	57.88	4	231.52	34.73	266.25

Figura 72 - Confirmación de registro correcto de cotización

En el área de "Guardar" el usuario puede dar una última revisión al documento antes de enviarlo a registrar en la base datos de SCM. Una vez que se envía y guarda exitosamente, se presentará una confirmación de lo mismo. En ese momento la cotización puede ser visualizada en el área de facturación donde el personal correspondiente la puede enviar a la bodega para que se confirmen las existencias relevantes y/o los traslados necesarios para que se pueda facturar.

Figura 73 - Correo de confirmación de registro de la cotización

F.7 Generación de Reportes

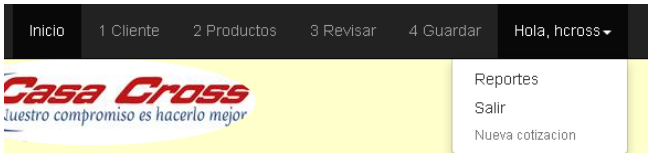


Figura 74 - Acceso al panel de reportes

Se puede acceder al panel de reportes a través del menú principal.

Figura 76 - Panel de reportes

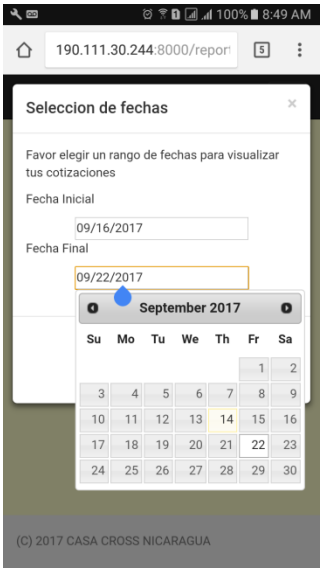


Figura 75 - Selección de parámetros para reporte de cotizaciones

En el panel de reportes va a ver un panel de botones. Favor *click* en “Mis cotizaciones” para visualizar el *modal* de “Selección de fechas”. Luego se pueden ver todas las cotizaciones que el usuario ha registrado en SCM a través de SAV que correspondan al rango de fechas especificadas. Para ver el contenido de una cotización, solo es necesario dar click en su columna.

Figura 77 - Página de reporte de cotizaciones y diálogo detalle de cotización

Se puede dar *click* en la fila de una cotización para ver el detalle de los productos que le corresponden.

Apéndice G – Manual de programación

Módulo: sqlcode

Custom JSON API for SCM's MSSQL and SAV's SQLITE / API JSON echo a la medida para MSSQL de SCM y SQLITE de SAV

Source: sqlcode.js, line 4

Métodos

(inner) ***checkUserExists(db, username)*** → *{object}*

Checks SQLITE3 to see if a product is in the current quotation in process / Revisa si un item está presente en la cotización actual

Parameters:

Name	Type	Description
Db	object	the sqlite3 db object / el objeto bd sqlite3
username	string	the username / nombre de usuario

Source: sqlcode.js, line 189

Returns:

matching row if any / fila resultante si existe

Type object

(inner) ***CotizacionTarjDescDoc(CTDDjson, transaction)*** → *{promise}*
Inserts records into dbo.[Cotizacion Tarj Desc Doc] in Casa Cross' SCM database / Inserta registros en dbo.[Cotizacion Tarj Desc Doc] en la base datos SCM de Casa Cross

Parameters:

Name	Type	Description
`CTDDjson`	object	objeto de parámetros
`transaction`	object	el objeto de la transacción que lleva la conexión actual

Source: sqlcode.js, line 724

Returns:
Type promise

(inner) ***CotizacionTarjetaKardex_BULK(CTKjson, transaction)*** → *{promise}*
Inserts records into dbo.[Cotizacion Tarjeta de Kardex] in Casa Cross' SCM database / Inserta registros en dbo.[Cotizacion Tarjeta de Kardex] en la base datos SCM de Casa Cross

Parameters:

Name	Type	Description
`CTKjson`	object	objeto de parámetros
`transaction`	object	el objeto de la transacción que lleva la conexión actual

Source: sqlcode.js, line 824

Returns:
Type promise

(inner) ***CotizacionVenta(CVjson, transaction)*** → *{promise}*
Inserts records into dbo.[Cotizacion Venta] in Casa Cross' SCM database / Inserta registros en dbo.[Cotizacion Venta] en la base datos SCM de Casa Cross

Parameters:

Name	Type	Description
`CVjson`	object	objeto de parámetros
`transaction`	object	el objeto de la transacción que lleva la conexión actual

Source: sqlcode.js, line 768

Returns:
Type promise

(inner) ***doClientSearch(clientName, clientID, ruc, credit)*** → *{object}*
Search for a client in SCM's database / Buscar cliente en tablas de base datos SCM

Parameters:

Name	Type	Description
`clientName`	string	client name / nombre del cliente
`clientID`	string	client unique ID / identificador del cliente
`Ruc`	string	client taxpayer ID / registro unico de contribuyente del cliente
`credit`	string	string-encapsulated boolean for stored procedure to filter clients with credit / filtrar por credito

Source: sqlcode.js, line 399

Returns:
result set

Type object

(inner) ***doProductSearch(username, codAlterno, codOriginal, txtDesc, txtAp, minStock, priceType, bodegas, codLinea)*** → *{object}*

Search for a product in SCM's database / Buscar cliente en tablas de base datos SCM

Parameters:

Name	Type	Description
username	string	
codAlterno	string	product ID / ID del producto
codOriginal	string	manufacturer's own ID for product / ID del fabricante para el producto
txtDesc	string	description of product / descripción del producto
txtAp	string	product's applications / aplicaciones del producto
minStock	string	minimum stock level to appear in results / nivel mínimo de existencia para salir en resultados
priceType	string	código de precio a mostrar en resultados
bodegas	string	códigos de bodegas a consultar
codLinea	string	código de linea a consultar

Source: sqlcode.js, line 480

Returns:

MSSQL result set

Type object

(inner) ***getAllBodegasFromSQLITE(db, username)*** → *{object}*
Get list of all bodegas / Consigue listado de todas las bodegas

Parameters:

Name	Type	Description
db	object	the sqlite3 db object / el objeto bd sqlite3
username	string	username / nombre de usuario

Source: sqlcode.js, line 257

Returns:
SQLite result set

Type object

(inner) ***getNonExcludedBodegas()*** → *{promise}*
Queries for list of bodegas that are not in MSSQL table NEW_BODEGAS_EXCLUIDAS
Consulta lista de bodegas no presentes en tabla MSSQL NEW_BODEGAS_EXCLUIDAS

Source: sqlcode.js, line 31

Returns:
Results of query / Resultados de la consulta

Type promise

(inner) ***getSupervisorUsers(db)*** → *{object}*
Get all users of the 4 (Supervisor) auth level / Conseguir lista de usuarios de nivel autorización 4 (Supervisor)

Parameters:

Name	Type	Description
db	object	the sqlite3 db object / el objeto bd sqlite3

Source: sqlcode.js, line 168

Returns:
query results / resultados de consulta

Type object

(inner) ***getUserAuthLevelFromSQLITE(db, username)*** → *{object}*
Get the authorization levels for a given user / Consigue niveles de autorización para un dado usuario

Parameters:

Name	Type	Description
db	object	the sqlite3 db object / el objeto bd sqlite3
username	string	the username / nombre de usuario

Source: sqlcode.js, line 303

Returns:
SQLite result set

Type object

(inner) ***getUserBodegasFromSQLITE(db, username)*** → *{object}*
Get the assigned bodegas for a given user / Consigue listado de bodegas asignadas a un usuario

Parameters:

Name	Type	Description
db	object	the sqlite3 db object / el objeto bd sqlite3
username	string	the username / nombre de usuario

Source: sqlcode.js, line 278

Returns:
SQLite result set

Type object

(inner) ***getUserCodigoVendedorFromSQLITE(db, username)*** → *{object}*
Get the Vendor ID for user / Consigue código de vendedor del usuario

Parameters:

Name	Type	Description
Db	object	the sqlite3 db object / el objeto bd sqlite3
username	string	the username / nombre de usuario

Source: sqlcode.js, line 235

Returns:
SQLite result set

Type object

(inner) ***getUserOperationsFromSQLITE(db, username)*** → *{object}*
Get the list of allowed operations for a user / Consigue lista de operaciones permitidas a un usuario

Parameters:

Name	Type	Description
db	object	the sqlite3 db object / el objeto bd sqlite3
username	string	the username / nombre de usuario

Source:sqlcode.js, line 209

Returns:
SQLite result set

Type object

(inner) ***getUserPasswordHashAndEnabledStateInSQLITE(db, username)*** → *{object}*

Get the user's password hash / Consigue el hash del usuario

Parameters:

Name	Type	Description
db	object	the sqlite3 db object / el objeto bd sqlite3
username	string	the username / nombre de usuario

Source: sqlcode.js, line 376

Returns:

SQLite result set

Type object

(inner) ***getUserPricesFromSQLITE(db, username)*** → *{object}*

Get the price access for a given user / Consigue tipos de precios accesibles para un dado usuario

Parameters:

Name	Type	Description
db	object	the sqlite3 db object / el objeto bd sqlite3
username	string	the username / nombre de usuario

Source: sqlcode.js, line 326

Returns:

SQLite result set

Type object

(inner) ***getVendedoresBySupervisorSQLITE(db, supervisorUsername)*** → *{object}*

Get the salespeople that belong to a given supervisor / Consigue los vendedores que le pertenecen a un supervisor

Parameters:

Name	Type	Description
db	object	the sqlite3 db object / el objeto bd sqlite3
supervisorUsername	string	the supervisor's username / nombre de usuario del supervisor

Source: sqlcode.js, line 58

Returns:

matching row if any / fila resultante si existe

Type object

(inner) ***isItemAlternoOnShoppingList(db, alt, username)*** → *{object}*

Checks SQLITE3 to see if a product is in the current quotation in process / Revisa si un item esta presente en la cotización actual

Parameters:

Name	Type	Description
db	object	the sqlite3 db object / el objeto bd sqlite3
alt	object	PK for shoppinglist / código alterno del producto
username	string	the username / nombre de usuario

Source: sqlcode.js, line 147

Returns:

matching row if any / fila resultante si existe

Type object

(inner) ***loopQuery(alterno, arrayOfWarehouseIds)*** → *{promise}*
Queries stock on hand of alterno for each warehouse in array of ware house IDs Consulta existencia del producto para cada bodega en arreglo de IDs de bodegas

Parameters:

Name	Type	Description
`alterno`	string	unique ID of product / identificador unico del producto
`arrayOfWarehouseIds`	array	array of warehouse uniq IDs / arreglo de identificadores únicos de bodegas

Source: sqlcode.js, line 80

Returns:
Results of query, resultados de la consulta

Type promise

(inner) ***setUserPasswordHashInSQLITE(db, username, hash)*** → *{object}*
Set the user's password hash / Reestablece el hash del usuario

Parameters:

Name	Type	Description
`db`	object	the sqlite3 db object / el objeto bd sqlite3
`username`	string	the username / nombre de usuario
`hash`	string	password hash / el hash de la contraseña

Source: sqlcode.js, line 353

Returns:
SQLite result set

Type object

Funciones Globales

assignUserBodegas(db, param)

Assigns specified bodegas to user account / Asigna bodegas especificadas a cuenta de usuario

Parameters:

Name	Type	Description
db	object	the sqlite3 db object / el objecto bd sqlite3
param	object	user data object / objeto de datos del usuario

Source: user.js, line 231

assignUserPrices(db, param)

Assigns prices to user account / Asigna precios a la cuenta de usuario

Parameters:

Name	Type	Description
db	object	the sqlite3 db object / el objecto bd sqlite3
param	object	user data object / objeto de datos del usuario

Source: user.js, line 183

clearUserBodegas(db, param)

Removes all bodega access from a user account as part of the process of changing user bodega access / Elimina todos los accesos a bodegas para que sean reasignados durante modificación del usuario

Parameters:

Name	Type	Description
db	object	the sqlite3 db object / el objecto bd sqlite3
param	object	user data object / objeto de datos del usuario

Source: user.js, line 208

clearUserPrices(db, param)

Removes all price access from a user account as part of the process of changing user price access / Elimina todos los accesos a precios para reasignación durante modificación del usuario

Parameters:

Name	Type	Description
db	object	the sqlite3 db object / el objecto bd sqlite3
param	object	user data object / objeto de datos del usuario

Source: user.js, line 159

createUserEntry(db, param)

Registers a new user. / Registra un nuevo usuario.

Parameters:

Name	Type	Description
db	object	the sqlite3 db object / el objecto bd sqlite3
param	object	user data object / objeto de datos del usuario

Source: user.js, line 135

exists(db, param)

Returns user data if account exists. / Regresa datos de usuario al existir la cuenta

Parameters:

Name	Type	Description
db	object	the sqlite3 db object / el objecto bd sqlite3
param	object	user data object / objeto de datos del usuario

Source: user.js, line 256

getAllUsers(db)

Gets all users and their data for display in /admin/userlist route / Consigue datos de todos los usuarios

Parameters:

Name	Type	Description
db	object	the sqlite3 db object / el objecto bd sqlite3

Source: user.js, line 25

getCotizacionKardex_MSSQL(numreg) → {object}

Query MSSQL to get quotation's kardex details / Consultar MSSQL para obtener detalles de kardex de cotizacion

Parameters:

Name	Type	Description
numreg	string	document unique id / numero de registro único

Source: lib/mssql.js, line 51

Returns:

matching rows if any / filas resultante si existe

Type object

getCotsByVendorCodeAndDateRange_MSSQL(vendedor, startdate, enddate, transaction) → {object}

Retrieve cotización data from cots registered in MSSQL / Consigue datos de las cotizaciones registradas en MSSQL

Parameters:

Name	Type	Description
vendedor	string	salesperson code / código de vendedor
startdate	string	the beginning of date range / fecha de inicio del rango
enddate	string	end of date range / fecha de fin del rango
transaction	object	mssql transaction object

Source: lib/mssql.js, line 28

Returns:

matching row if any / fila resultante si existe

Type object

setPasswordHash(db, username, hash)

Sets the user's password hash in db during user creation and modification. / (Re)establece el hash de la contraseña del usuario en la bd durante la creación y modificación de usuario.

Parameters:

Name	Type	Description
db	object	the sqlite3 db object / el objecto bd sqlite3
username	string	the username / nombre de usuario
hash	string	hashed password / hash de contraseña

Source: user.js, line 62

updateUserDataExceptPasswordHash(db, param)

Updates user's db data leaving the hash unchanged. / Actualiza datos del usuario sin tocar el hash.

Parameters:

Name	Type	Description
db	object	the sqlite3 db object / el objeto bd sqlite3
param	object	user data object / objeto de datos del usuario

Source: user.js, line 91

updateUserDataIncludingPasswordHash(db, param, hash)

Updates user's db data including the password hash. / Actualiza datos del usuario incluyendo el hash.

Parameters:

Name	Type	Description
db	object	the sqlite3 db object / el objeto bd sqlite3
param	object	objeto de datos del usuario
hash	string	user password hash / hash de contraseña del usuario

Source:user.js, line 113

Apéndice H – Asistencia a Capacitaciones

A continuación se presenta la lista de asistencia a la última capacitación dada al equipo de ventas el día sábado 14 de octubre 2017 en el tercer piso del edificio Casa Cross.

Tabla 25 - Lista de asistencia a capacitación final

Ejecutivo	Rubro	Sucursal	Tipo de cuenta
Ana Rocha	Consumo Jefe de Canal	Casa Matriz	Supervisor
Jordan Flores	Asistente a Jefe de Canal Consumo	Casa Matriz	Ninguna
Roberto Gonzales	Aftermarket	Esquina	Vendedor
Carlos Chalupa	Aftermarket	Esquina	Vendedor
Harry Rojas	Aftermarket	Esquina	Vendedor
Wilmer Henriquez	Consumo	Chinandega	Vendedor
Emir Alvarez	Consumo	Chinandega	Vendedor
Darwin Rodriguez	Consumo	Esteli	Vendedor
Franklin Guillen	Consumo	Titanic	Vendedor
Luis Pineda	Consumo	Titanic	Vendedor
Javier Velasquez	Consumo	La Subasta	Vendedor
Orlando Navas	Consumo	Esteli	Vendedor
Marlon Mairena	Consumo	Esteli	Vendedor
Roberto Briceño	Consumo	La Subasta	Vendedor
Jairo Gomez	Aftermarket Jefe de Canal	Casa Matriz	Supervisor
Edgard Silva	Asistente a Jefe de Canal Aftermarket	Casa Matriz	Ninguna
Eduardo Castillo	Aftermarket	Titanic / La Subasta	Vendedor
Alexander Sanchez	Aftermarket	Chinandega	Vendedor
Julio Martinez	Aftermarket	Esteli	Vendedor
Julio Castellon	Aftermarket	Esteli	Vendedor
Ayax Martinez	Aftermarket	Titanic / La Subasta	Vendedor
Jordan Gomez	Aftermarket	La Subasta	Vendedor