Universidad Nacional de La Matanza

Fundamentos de Comercio Electrónico (e-Commerce)

Versión: 2.1

Cátedra: Ing. Rolando Daumas

Historial de modificaciones

Versión	Modificación	Fecha
1.0	Versión Inicial	10-04-2016
2.0	Revisión 2016	01-04-2016

Indice

Objetivo de la materia Fundamentos de Comercio Electrónico

El incremento en el comercio electrónico es exponencial. Ya ha tomado un rol protagónico entre los proyectos de las empresas más importantes. Y la confianza que inspiran las plataformas en los usuarios es fundamental para su crecimiento. Según la Cámara Argentina de Comercio Electrónico, el crecimiento interanual en el año 2016 fue del 51%, con una facturación de $102.700 millones. El 90% de los argentinos conectados hicieron al menos una compra por medio de algún sitio de Comercio Electrónico. Y en promedio cada comprador gastó $2185, lo que implica un 21% de incremento con respecto al año 2015.

El objetivo de la materia es que el profesional egresado tenga el conocimiento necesario como para involucrarse y tomar un rol protagónico en proyectos de esta índole.

La implementación del comercio electrónico en las empresas no consta solamente de montar una tienda en internet. El intrincado esquema de interacciones con otras áreas hace necesario la intervención de profesionales con conocimiento global de todo lo que implica el comercio electrónico.

Es por ello que este curso intentará darles los conocimientos básicos para poder insertarse en cualquier proyecto de e-commerce. Se verá, en mayor o menor profundidad, desde el modelo de negocio, pasando por el ciclo de vida de los proyectos de e-commerce, integraciones con otros sistemas, arquitecturas típicas, patrones de diseños utilizados, pasarelas de pago, seguridad y rendimiento, hasta conceptos de marketing y posicionamiento.

Para lograr el objetivo el curso se dividirá en 3 secciones temáticas en función del objeto de estudio. La primera se abocará a dar los fundamentos básicos del comercio electrónico y su impacto en las empresas. La segunda sección se enfocará en lo que refiere a la dinámica de los equipos de trabajos que lleve adelante el desarrollo y operación de un sitio de e-Commerce típico. Y por último se verán los aspectos técnicos (software y hardware) que son comunes en la mayoría de los proyectos de implementación de un sitio de e-Commerce.

UNIDAD 3: Arquitectura

Objetivo: Describir todos los elementos técnicos involucrados en un proyecto de Comercio Electrónico.

3.1 Atributos de Calidad de la arquitectura de un sitio de e-Commerce.

El campo de la Arquitectura de Software se enfrenta al tema de la complejidad al definir un sistema de software desde un punto de vista abstracto. Partiendo el problema en pequeñas partes lo que hace más fácil analizar entidades separadas y la relación entre ellas. Implica el proceso de definir una solución estructurada que cumple con todos los requerimientos técnicos y operacionales, mientras se optimizan los atributos de calidad comunes.

La arquitectura de un sistema de software define la estructura general en términos de componentes e interacciones entre estos componentes. Un componente, o un módulo, puede ser identificado con una unidad que efectúa cierta función. Un ejemplo de un componente es una base de datos o algún objeto. Las interacciones entre los componentes ocurren a través de conectores como ser una llamada a un procedimiento, o algún protocolo. Finalmente, las propiedades de un componente especifican cómo puede realizarse una conexión con otro componente por intermedio de un conector.

El dejar de lado detalles insignificantes permite desarrollar una serie de planos que muestran la estructura general de un sistema desde distintas perspectivas. Como un plano de plomería de una casa es diferente de un plano de electricidad de la misma casa, las vistas estáticas y dinámicas de un sistema de software transmiten distintos significados.

El razonamiento lógico detrás de desglose de los componentes y la relación entre los componentes no es casual. Determinar solo el funcionamiento de un sistema es solo la mitad de la batalla. Es importante direccionar la forma en la que la funcionalidad es diseñada, desarrollada y desplegada dado que puede ocurrir que dos sistemas con los mismos requerimientos de funcionalidad pueden tener diferentes necesidades en lo que refiere a cualidades. Por ejemplo, si uno de los sistemas debe ser distribuido sobre múltiples plataformas y el otro sistema debe correr exclusivamente sobre la plataforma Windows Server, entonces las decisiones arquitecturales hechas en cada caso van a ser diferentes.

3.1.1 Atributos de Calidad de una Arquitectura de Software

Desarrollar software de alta calidad es difícil, especialmente cuando el término “calidad” está parcialmente basado en el ambiente en el que es usado. Con el fin de saber si la calidad deseada se ha alcanzado, o degradado, ésta debe ser medida, pero justamente determinar qué medir y como es la parte difícil.

Los Atributos de Calidad del Software **son puntos de referencia (benchmarks) que describen el comportamiento previsto del sistema para el ambiente para el cual fué construido**. Los Atributos de Calidad proporcionan los medios para medir la aptitud e idoneidad de un producto. Una definición de arquitectura de software tiene de una forma u otra un profundo efecto sobre las cualidades de un sistema, y los Atributos de Calidad de Software afectan directamente a la arquitectura.

Identificar la calidad deseada de un sistema antes de que éste sea construido permite al diseñador del mismo dar forma a una solución (partiendo de su arquitectura) que satisfaga los requerimientos deseados dentro del contexto de restricciones (recursos disponibles, interfaces con sistemas legacy, etc). Cuando un diseñador entiende las cualidades deseadas antes de que el sistema sea construido se mejora la probabilidad de seleccionar o crear la arquitectura adecuada.

A continuación se puede ver una tabla con los atributos de calidad más utilizados divididos en distintos aspectos de la arquitectura que definirán. Estas definiciones son a nivel general. En la siguiente sección se verán cuáles de ellos son importantes para los sitios de eCommerce.

Categoría	Atributo de Calidad	Descripción
Cualidades de Diseño	Integridad Conceptual	La integridad conceptual define la consistencia y coherencia del diseño general. Esto incluye la forma en la que los componentes o módulos son diseñados así como los factores como estilo de codificación y nombrado de variables.

	Mantenibilidad	La mantenibilidad es la capacidad del sistema para someterse a cambios con cierto grado de facilidad. Esos cambios pueden impactar en componentes, servicios, funcionalidades e interfaces cuando se agregan o cambian funcionalidades, cuando se corrigen errores y cuando se cumplen con nuevos requerimientos del negocio.
	Reusabilidad	La reusabilidad define la capacidad de los componentes y subsistemas de ser utilizados en otras aplicaciones y/o escenarios. La reusabilidad minimiza la duplicación de componentes y también el tiempo de implementación.
Cualidades en Tiempo de Ejecución	DIsponibilidad	La disponibilidad define la proporción del tiempo en el que el sistema es funcional y está operativo. Este puede ser medido como un porcentaje del total del tiempo en el que el sistema está no operativo con respecto a un rango de tiempo predefinido. La disponibilidad será afectada por los errores del sistema, problemas de infraestructura, ataques maliciosos y carga del sistema.
	Interoperabilidad	Interoperabilidad es la habilidad de un sistema o diferentes sistemas de operar exitosamente comunicando e intercambiando información con otro sistema externo, construido y ejecutado por partes externas. Un sistema interoperable hace fácil el intercambiar y reutilizar información tanto interna como externamente.
	Operabilidad	La operabilidad define cuán fácil es para los administradores de un sistema manejar la aplicación, a través de las herramientas ofrecidas por la misma para su control, depuración y rendimiento.
	Performance	La performance indica de la capacidad de respuesta de un sistema para ejecutar cualquier acción dentro de un intervalo de tiempo dado. Se puede medir en términos de latencia o rendimiento. La latencia es el tiempo necesario para responder a cualquier evento. El rendimiento es el número de eventos que tienen lugar dentro de un período de tiempo determinado.
	Confiabilidad	La confiabilidad es la capacidad de un sistema de funcionar de acuerdo a lo esperado en el transcurso del tiempo. La confiabilidad es medida como la probabilidad que un sistema no falle y realice sus funciones definidas por un tiempo específico de tiempo. Por ejemplo, si un sistema se rompe o empieza a no responder adecuadamente, automáticamente se redirecciona a los usuarios a un sistema de respaldo. Otro ejemplo sería cuando un sistema empieza a tener salidas inconsistente, el sistema autodetecta esta condición mediante el monitoreo de logs y deriva el tráfico de procesos a alguna instancia que pueda hacer responder adecuadamente.
	Escalabilidad	La escalabilidad es la capacidad de un sistema para manejar ya sea aumentos en la carga sin impacto en el rendimiento del sistema, o la capacidad de ser ampliado fácilmente.
	Seguridad	La seguridad es la capacidad de un sistema para evitar acciones maliciosas o accidentales fuera del uso diseñado, y para evitar la divulgación o pérdida de información. Un sistema seguro tiene como objetivo proteger los activos y evitar la modificación no autorizada de información.
Cualidades de Sistema	Compatibilidad	Compatibilidad es la capacidad de un sistema de proveer información que sirva para identificar y resolver problemas cuando este falla y no trabaja correctamente.
	Capacidad de ser probado	Testability es una medida de cuán fácil es crear criterios de prueba para un sistema y sus componentes, ejecutar esas pruebas con el objeto de determinar si los criterios anteriores son cumplidos. Una buena capacidad de ser probado hace más probable que una condición de fallo en un sistema pueda ser aislado de forma oportuna y eficaz.
Cualidades	Usabilidad	Usabilidad define como la aplicación cumple los requerimientos de los usuarios y

de Usuario		consumidores de ser intuitiva, fácil de localizar y globalizada, proveyendo buen acceso para usuarios con capacidades diferentes, y resultando en una buena experiencia global.

3.1.2 Orden de importancia de Atributos de Calidad para sitios de e-Commerce.

A continuación se da una lista con los atributos de calidad más preponderantes en lo que refiere a sitios de e-Commerce. Las compañías que implementan un e-Commerce dependen de que sus clientes utilicen el sitio, y lo que es más importante, que vuelvan a usarlo en reiteradas oportunidades. De esta forma, no como otras aplicaciones web, los sitios de e-Commerce solo hacen dinero si los sitios satisfacen las necesidades de los clientes.

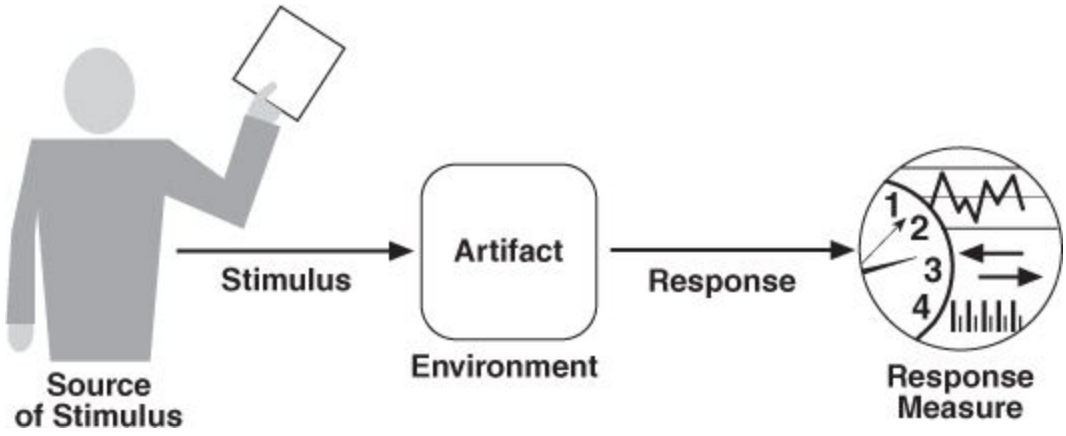
1. **Confiabilidad:** Es crítico para el éxito en el negocio la fiabilidad (*no fallar en condiciones normales de operación*) del sitio de e-Commerce. Y llegar a los objetivos propuestos depende de este atributo de calidad. Por ejemplo, uno tiene pocas o ninguna opción de seleccionar el procesador de texto que uno quiere utilizar, pero hay docenas de librerías en línea entre las que se puede optar. Muchos sitios de e-Commerce compiten en su mercado con otros muchos sitios de similar índole. Más competencia significa que los usuarios serán capaces de realizar selecciones de acuerdo a que tan bien funcione un sitio. Si un sitio no funciona bien, el cliente no pensará mucho al momento de irse a buscar otro sitio y simplemente pondrá otra url en su browser.
2. **Usabilidad:** Las aplicaciones de software tradicionales tienen usuarios, pero los sitios de e-Commerce tienen clientes. Estos clientes esperan que los sitios sean simples de utilizar como lo es comprar en una tienda tradicional. *Los clientes esperan ser capaces de usar el sitio sin entrenamiento.* De esta forma, el software debe fluir de acuerdo a las expectativas de los usuarios, presentando solo la información necesaria y facilitando controles de navegación claros y obvios.
3. **Seguridad:** Cuando los sitios de las empresas eran solo catálogos en línea, las consecuencias de los agujeros de seguridad eran relativamente pequeños. En cambio, en los sitios de e-Commerce de hoy día, si una empresa falla en el aspecto de seguridad tiene que hacer frente a una significativa pérdida de ganancias, grandes costos de reparación, consecuencias legales y puede perder credibilidad con sus clientes. Es por eso que el sitio manipule los datos de los clientes y otra información electrónica de forma lo más segura posible.
4. **Disponibilidad:** En las tiendas tradicionales, dado que son atendidas por personal hay momentos en los que las mismas se encuentran cerradas (por almuerzo, vacaciones, feriados, fines de semana). Pero en una tienda de e-Commerce, el cliente siempre espera que esté disponible, sea la hora que sea, sea el día que sea. Cuando una tienda se encuentra cerrada a la navegación, el cliente puede irse a otra y no volver nunca más.
5. **Escalabilidad:** En una tienda tradicional la cantidad de clientes depende del tamaño del vecindario en el que se encuentra. En cambio el vecindario de una tienda de e-Commerce es el mundo entero mismo, por lo tanto puede ser visitado por un número ilimitado de clientes. Es por esto que una tienda de e-Commerce debe estar preparada para crecer rápidamente en términos de número de clientes atendidos y servicio prestados.
6. **Mantenibilidad:** El nivel de actualizaciones requerido por los sitios de e-Commerce es alto en comparación con otras aplicaciones. Es por eso que generalmente estas soluciones se basan en la tecnología web. Porque en éstas el ratio de actualizaciones es mayor y las actualizaciones pueden estar online inmediatamente sin largos ciclos de desarrollo, pruebas y despliegue.
7. **Performance:** Si un sitio es muy lento , los clientes pierden la paciencia y terminan abandonando la tienda. Los clientes pierden la concentración si tienen que esperar más de unos pocos segundos y abandonan el sitio y si esa espera es mayor a 30 segundos, nunca más volverán. En esta forma la performance es importante, pero es importante también notar que la performance es dominada por el tráfico de internet y frecuentemente

3.1.3 Escenarios

Expresiones como “el sistema debe tener alta performance” o “el sistema debe ser amigable con el usuario” son aceptables en las primeras etapas de la definición de los requerimientos. Pero ni bien más información de torna disponible, las expresiones anteriores se tornan inútiles dado que no significan nada para el propósito de diseñar una solución. En cada

uno de los ejemplos anteriores se hace un intento de describir el comportamiento de un sistema. Ambas expresiones son inútiles dado que no proveen una forma tangible de medir el comportamiento de un sistema.

Un Atributo de Calidad debe ser definido en términos de escenarios, como por ejemplo “cuando 100 clientes inician el paso ‘Completar Pago’, el componente de pagos, bajo circunstancias normales, procesara los requerimientos con una latencia promedio de 3 segundos”. Esta expresión, ó escenario, le permite a un arquitecto crear argumentos cuantificables acerca de un sistema. Un escenario define la fuente de estímulo (usuarios), el estímulo actual (iniciador de transacción), el artefacto afectado (componente de pago), el ambiente en el cual éste existe (operación normal), el efecto de la acción (transacción procesada), y la respuesta medida (dentro de los tres segundos)



Una forma común de describir un escenario se divide en las siguientes partes:

- **Fuente del Estímulo:** Un estímulo debe tener una fuente, éste debe venir de algún lado. **El estímulo puede ser un humano, un sistema o cualquier otro agente que interactúa con el sistema**. La fuente del estímulo puede afectar la forma en que es tratada por el sistema. El requerimiento de un usuario no autenticado puede ser tratado de otra forma con respecto a un usuario autentificado.
- **Estímulo:** Se usa el término “Estímulo” para describir el **arribo de un evento al sistema**. Para las cualidades de desarrollo del sistema se utilizará también el mismo término “Estímulo”, de esa forma para el atributo de calidad “Modificabilidad” el estímulo es un requerimiento de modificación, y para el de “Testeabilidad” podría ser el hecho de haberse completado un ciclo de desarrollo.
- **Ambiente:** El ambiente de un requerimiento es el **conjunto de circunstancias en el cual el escenario sucede**. El ambiente actúa como un clasificador del estímulo. Por ejemplo, un requerimiento por una modificación en el sistema que arriba después de que el código ha sido congelado por una versión puede ser tratado en forma diferente que uno que arriba antes del congelamiento de código. Una falla en un componente que es la quinta de un conjunto de fallas consecutivas puede ser tratada en forma diferente que una falla que es la primera del mismo componente.
- **Artefacto:** El artefacto es la **porción de sistema sobre el cual es requerimiento se aplica**. Generalmente es el sistema entero, pero ocasionalmente puede ser porciones específicas del mismo. Una falla en el almacenamiento de datos puede ser tratado en forma distinta que si la falla se da en el almacenamiento de metadatos. Las modificaciones de la interfaz de usuario puede tener tiempos de respuesta más rápidos que modificaciones en la capa media del sistema.
- **Respuesta:** La forma en que el sistema puede responder al estímulo también debe ser especificado. **La respuesta consiste en las responsabilidades que el sistema o los desarrolladores pueden llevarse a cabo en respuesta a un estímulo**. Por ejemplo en un escenario de performance, un evento arriba (el estímulo) y el sistema debe procesar este evento y generar la respuesta. En un escenario de modificabilidad, un requerimiento de modificación arriba (el estímulo) y el desarrollador debe implementar la modificación -sin efectos colaterales- entonces prueba y despliega la modificación.
- **Medida de la respuesta:** Determinar cuándo una respuesta es satisfactoria -cuando el requerimiento es satisfecho- solo es posible si se puede medir la respuesta. **La medida de la respuesta es el valor de una o más variables contra las cuales se comparará la respuesta del sistema al evento**. Para performance puede ser una medida de

latencia o rendimiento; para modificabilidad puede ser el tiempo que se espera para desarrollar, probar y desplegar la modificación.

Ejemplos de escenarios de atributos de calidad: https://www.cs.unb.ca/~wdu/cs6075w10/sa2.htm

3.2 Desempeño y Escalabilidad

El desempeño y la escalabilidad son dos factores técnicos fundamentales para la vida de un sitio de eCommerce. Generalmente, ambos factores son abordados en forma conjunta con una misma solución. Esto es porque finalmente ambos, desempeño y escalabilidad, se refiere a la habilidad del sito web para atender tráfico rápidamente (desempeño) y con la suficiente capacidad para que éste continúe desempeñándose bien bajo un tráfico que se incrementa en corto tiempo (escalabilidad).

3.2.1 ¿Qué es el desempeño (performance) de un sitio web?

El desempeño de un sitio web es definido como la velocidad en la que sus páginas son descargadas y mostradas a los visitantes del sitio o potenciales clientes del eCommerce. Muchos estudios indican que un mejor desempeño conducen a un mayor número de páginas vistas, a una menor tasa de rebote (rebote es cuando un usuario llega a un sitio y en la primer página visitada abandona el mismo) y a un incremento de las ganancias.

Más allá de que un sitio web con buen desempeño influye directamente con la experiencia de usuario y por consiguiente con la permanencia de los usuarios en el sitio. Hay que tener en cuenta que los buscadores también consideran el desempeño de los sitios para generar las listas de resultados de los buscadores. Los sitios con un desempeño mejor posicionan mejor en las búsquedas y si se hace un buen trabajo en lo que respecta a Search engine Optimization (SEO) esto aumenta las posibilidades de aparecer en los primeros puestos de las búsquedas orgánicas.

A continuación se puede ver un estudio del sitio Section.io (https://www.section.io/) el cual mediante métricas sacadas de un sitio de eCommerce de tamaño grande, se puede ver el correlato que existe entre la velocidad de la primera página visitada y la propensión a realizar una conversión en el sitio.

Este gráfico demuestra la clara relación entre la velocidad de la página y el ratio del checkout. A medida que la página se muestra más lenta para este eCommerce, el ratio del checkout cae.

Otros estudios indican que los usuarios a los cuales les han cargado las páginas más rápido han visto 16% más páginas, 9% más páginas de producto, han llegado un 15% más al checkout y se han visto incrementado los checkouts exitosos en un 16%.

Existe un variado número de métricas utilizadas para medir el desempeño de un sitio web. Pero el principal a tener en cuenta es:

Tiempo del Primer Byte (TTFB, Time to First Byte) o Tiempo de Carga de HTML (HTML Load Time).

Cuando un usuario ingresa una url en su navegador, existen un par de pasos que ocurren antes de la que página empieza a ser cargada. Por ejemplo, si el usuario ingresó la url sin el www, éste deberá ser primero redireccionado a la dirección con www. Podría existir aún otro redireccionamiento, como por ejemplo si el usuario tiene que ser enviado a uno y otro sitio si es que ingresa desde una computadora de escritorio o desde un dispositivo móvil. El navegador necesita reunir toda esa información y conectarse a su servidor web (donde el código es alojado) antes que todo esto suceda.
Cuando la conexión se establece, el servidor empieza a enviar información en la forma de documento HTML que le dice al browser que acciones necesita tomar y que archivos necesita recuperar para construir la página. Aquí es donde el código de la aplicación es ejecutado y los llamados a la base de datos son efectuados, y si el documento HTML no está en caché, este proceso sucede una y otra vez.

Un TTFB ideas es alrededor de 200 milisegundos lo cual puede ser logrado cuando el documento HTML es servido desde el caché.

Tiempo de inicio de visualización (SRT, Start Render Time)

Una vez que el navegador recibe el documento HTML, este comienza a construir la página efectuando llamadas adicionales al servidor, por cada recurso propio de la página (imágenes, logos, texto, etc). En promedio una página puede necesitar cerca de 100 request al servidor para recuperar todo el contenido necesario.
El tiempo de inicio de visualización es una medida importante porque es cuando el usuario ve por primera vez a la página en su navegador.

Tiempo de carga de página (PLT, Page Load Time).

El PLT es probablemente la métrica más comúnmente utilizada para evaluar la velocidad de una página, y es la cantidad de tiempo (medido en segundos) que transcurre desde que el usuario ingresa la URL en el navegador hasta que la misma es completamente cargada.

Algunos navegadores indican gráficamente el estado de carga de la página, mediante barras de progreso o con imágenes móviles. Es fácil entender porqué estos indicadores visuales son referenciados frecuentemente. Sin embargo, si la página a cargar tiene una imagen grande o algún archivo que se carga por debajo de la línea de visión inicial (el usuario tiene que hacer un desplazamiento hacia abajo para verlo), esto podría retardar la velocidad de carga de la página inclusive si el usuario ve como si la página ya estuviese cargada.
El tiempo de carga de la página ideal es por debajo de los 2 segundos, de esta forma se previene el rebote e incrementa el compromiso con el sitio.

3.2.2 ¿Qué es la escalabilidad?

En términos de sitios de eCommerce, la escalabilidad es la habilidad de un sitio de poder afrontar exitosamente tanto incrementos como reducciones de tráfico en un lapso de tiempo acotado.
Generalmente los eCommerce tienen que afrontar variaciones de visitantes durante la temporada alta o cuando alguna acción de marketing atrae usuarios al sitio. Si el sitio de eCommerce no escala satisfactoriamente, eso podría significar que todos esos clientes potenciales se van a encontrar con un sitio lento o inclusive no pudiéndose conectar directamente al mismo tiempo en el que el propietario del eCommerce está tratando de aprovechar el incremento en el tráfico.

3.2.3 Métricas de desempeño

Anteriormente hemos visto porque en importante el desempeño y la escalabilidad de un sitio de eCommerce. En esta sección se enumeran algunas herramientas que nos proveerán de las métricas necesarias para medir a los sitios web, las cuales nos permitirán.

Tiempo de carga de página (Page Load Time):

Es el tiempo que transcurre desde que empieza la negociación inicial hasta que la página está completamente cargada en el navegador. Esta métrica será la más grande dado que ella incluye todos los pasos para la carga de la página; pero es importante no perder de vista a las métricas más chicas.

Tiempo de Redireccionamiento (Redirect Time):

Es la cantidad de tiempo que insume el redireccionamiento total a el dominio correcto. Por ejemplo cuando la url ingresada es TuSitio.com y el sitio está publicado en www.TuSitio.com. Si no hay direccionamiento, este tiempo es 0.

Localización DNS (DNS Lookup):

Es la cantidad de tiempo que le insume al navegador encontrar la dirección IP del sitio que se busca.

Negociación HTTPs (HTTPS Negotiation):

Si el sitio sirve contenido por intermedio de https, significando que el contenido está encriptado, el navegador y el servidor deben intercambiar información para verificar la conexión SSL.

Tiempo de Conección al servidor y tiempo de respuesta del servidor (Server Connection and Server Response Time):

Una vez que el navegador encuentra la IP del sitio, estos dos tiempos corresponden al tiempo que insume conectarse al servidor y el tiempo que éste tarda en responder.

Tiempo de carga de HTML(HTML Load Time):

Esta métrica es también llamada TTFB (vista anteriormente) y es el tiempo que tarda el document HTML en ser enviado al navegador web.

Tiempo de inicio de Visualización (Start Render Time):

Es el momento inicial en el el contenido de una página no blanca se torna visible y es visualizado en el navegador web.

Documento completamente cargado o Contenido de documento cargado (Document Complete or Document Content Loaded):

Es cuando el documento HTML termino de cargar, pero otros elementos, como imágenes referenciadas en el HTML aún no han sido totalmente requeridas.

Tiempo de carga total (Fully Loaded Time):

Es el tiempo que transcurre entre la navegación inicial y los dos segundos siguientes luego de que no haya actividad de red en luego del Document Complet. Esto incluye cualquier actividad de javascript que sea disparada luego de la carga principal de la página.

Para ver como el desempeño de un sitio impacta en la experiencia del usuario, también debe tenerse en cuenta las siguiente métricas importantes para el marketing.

Páginas Vistas (Page viewed):

Es el número de páginas vistas por usuario en el sitio en un determinado lapso de tiempo.

Páginas por sesión (Pages/Session):

Es el promedio de páginas vistas por sesión de usuario. Todas las vistas repetidas de una página son contadas.

Duración de la sesión (Session Duration):

Promedio de tiempo en el cual el usuario permanece en el sitio. Teniendo en cuenta que el usuario podría no estar activamente comprometido con el sitio durante todo este tiempo.

Ratio de conversión (Conversion Rate):

Si se configuraron las metas correctamente, el ratio de conversión indicará el porcentaje de usuarios que exitosamente lograron la meta, como por ejemplo realizar una compra.

3.2.4 Formas de medir el desempeño y la escalabilidad.

Hasta aquí hemos visto las métricas importantes. En la siguiente sección veremos las distintas formas que hay para obtener esas métricas del sitio web. Existen 4 tipos principales, que nos indican la forma de encarar la problemática de obtener las métricas. Para cada una de estas formas de medir el desempeño y la escalabilidad existen muchas herramientas en el mercado.

Estas maneras de medir son:

- **Monitoreo Real del Usuario**
- **Pruebas Sintéticas Web**
- **Monitoreo de Desempeño de Aplicación**
- **Pruebas de Carga**

Monitoreo Real del Usuario (Real User Monitoring - RUM)

Esta manera mide cómo los visitantes del sitio actuales experimentan el sitio web, qué páginas ven, cuanto tiempo tardan en cargar las páginas vistas, cuando estos visitantes abandonan o rebotan en el sitio y muchas otras métricas acerca de casi todas las interacciones de estos visitantes con el sitio.

También se registra información respecto a la dirección IP del visitante, localización, tipo de navegador, tipo de dispositivo, así como qué acciones toman los visitantes dentro del sitio.

Se implementa mediante tecnologías pasivas de monitoreo que registran todas las interacciones del usuario en el sitio. Esto quiere decir que la recolección de datos no interfiere con el uso del sitio. El más común ejemplo de herramienta RUM son Google Analytics o Pingdom.

Lo valioso de RUM es que nos muestra cómo los usuarios utilizan actualmente nuestro sitio y puede sacar a la luz problemas que un ciclo de pruebas común no podría encontrar. Ésto porque los usuarios visitan al sitio desde diferentes localidades, navegadores o dispositivos, y tienen distintas formas de navegar el sitio, viendo productos, entrando en el checkout, etc. Y de esta forma RUM provee un amplio rango de análisis que puede darnos una idea del desempeño estándar para todo tipo de usuarios.

Una de las desventajas de RUM es que éste se basa solamente en el tráfico entrante actual. Y si el sitio medido es relativamente nuevo, el mismo no tendrá el suficiente tráfico para proveernos de una buena base de datos histórica como para evaluar el desempeño o la escalabilidad. Si un sitio no tiene el suficiente tráfico como para aplicar un monitoreo RUM, la otra alternativa para obtener datos acerca del desempeño es mediante un Prueba Web Sintética.

Prueba Web Sintética:

Estos tipos de pruebas, también llamadas Monitoreo Activo, es ejecutado por un emulador de navegador web que crea scripts que se ejecutan y miden el desempeño de un sitio como si éste fuese un usuario real. Esto también permite medir el desempeño antes de que el tráfico sea enviado al sitio web, a un sitio de pruebas o de desarrollo, de forma tal que el problema sea identificado por las pruebas y resuelto antes de que los usuarios lo terminan experimentando en carne propia.

Los scripts de este tipo de pruebas se confeccionan para que la navegación pase por determinados caminos dentro del sitio web, lo que no siempre es sinónimo de que el tráfico pase realmente por esos caminos que han sido indicados en los scripts. Sin embargo, este tipo de pruebas puede ser utilizado para asegurar que ciertos circuitos dentro del sitio web funcionan según lo esperado, como por ejemplo el checkout, luego de que se han realizado modificaciones al mismo; o para probar la carga de una página y ver que elementos del sitio desempeñan bien y cuales tienen oportunidad de optimizarse.

Las pruebas sintéticas sirven también para ver como el sitio se desempeña en distintos navegadores y locaciones geográficas, inclusive si el sitio no tiene visitantes de esa posición.

Una de las herramientas más útiles de este tipo de pruebas es WebPageTest.org, una herramienta que permite correr pruebas sintéticas desde una locación específica, navegador y velocidad de conexión. Esta herramienta provee de una vista de cascada de la página probada con las métricas claves, como el TFB, SRT y PLT.

Monitoreo de Desempeño de Aplicación (Application Performance Monitoring, APM):

Este es un monitoreo más técnico que ve a la aplicación por detrás, como ser la ejecución de código, cuántas llamadas a la base de datos se hace y cuánto tiempo toman en ejecutarse, y como los recursos del servidor son utilizados. Un ejemplo de herramienta de este tipo de monitores es New Relic.

Pruebas de Carga (Load Testing):

Para obtener una visión de cómo sería el desempeño de un sitio web con tráfico adicional hay que hacer una prueba de carga. Hay varias herramientas online para ejecutar pruebas de carga, que simulan y envían determinados volúmenes de usuarios virtuales al sitio web al mismo tiempo.

Este tipo de pruebas se realizan para determinar la cantidad de tráfico que un sitio puede manipular y para identificar puntos débiles en la arquitectura del sitio. Desafortunadamente, no es posible simular el comportamiento de usuarios reales. Variables como el tipo de navegadores de los usuarios, indexadores que realizan rastreos en tiempos aleatorios, localización de los usuarios, velocidad de los navegadores de los usuarios, número de páginas vistas, nuevos usuarios versus los que retornan, caches de browsers, ingresos al checkout o acciones de agregar productos al carrito y más forman una miríada de opciones a considerar al momento de crear los scripts para generar y simular carga desde una distribución válida de navegadores reales.

En el mejor de los casos, las pruebas de carga pueden convertirse rápidamente en una actividad muy costosa que le dará sólo una indicación de los puntos de carga en su aplicación. En el peor de los casos, las pruebas de carga pueden proporcionar resultados altamente engañosos. En cualquiera de los escenarios, puede asegurarse que la experiencia de carga real diferirá de los resultados de la prueba de carga.

Una vez que haya utilizado herramientas para medir el rendimiento y la escalabilidad del sitio actual, se podrá tener una buena base para mejorar. Una buena práctica es tomar nota de las métricas actuales, realizar las mejoras que se consideran necesarias y luego realizar nuevamente las pruebas y comparar resultados.

3.3. Diseño de plataforma de e-Commerce

Patrones de diseño utilizados frecuentemente. MVC, Observer, Subscriber.

3.3.1 Arquitectura en capas.

Modelo de 3 capas

A fines de los 90's, y principios del presente siglo la industria del desarrollo de software se inclinaba por un modelo de arquitectura de sistemas llamado de tres capas. Mediante ella se buscaba obtener un bajo acoplamiento y una alta cohesión en el código de una aplicación. Una descripción de estas capas se detalla a continuación.

Arquitectura de tres capas.

El diseño de aplicaciones por capas es extremadamente popular porque incrementa en la aplicación el desempeño, la escalabilidad, flexibilidad, la reutilización de código, y otros beneficios. En el diseño clásico de tres capas, las aplicaciones se dividían en las tres mayores áreas funcionales: la capa de acceso a datos, la capa de negocio y la capa de presentación. Dentro de cada capa también podrían existir una serie de sub-capas que proveen una mayor granularidad en las áreas funcionales de la aplicación.

Capa de Acceso a Datos (Data Access Layer - DAL): esta capa administra el almacenamiento físico y el recupero de los datos. En pocas palabras, este ejecuta los Select, Insert, Update y Delete contra la base de datos.

Capa de Lógica de Negocio (Business Logic Layer - BLL): Es la parte esencial de todo el sistema, éste mantiene las reglas de negocio y la lógica. La relevancia del diseño de la capa de lógica de negocio está asociada a la particularidad de la lógica implementada en un e-Commerce, como ser encuestas de marketing, crear órdenes de compra, administración de carros de compra, proceso de pago.

Capa de Presentación: Esta capa incluye la interfaz de usuario y el código relacionado con la presentación de la aplicación. En esta capa no debiera incluirse lógica de negocio. El código con lógica en la capa de presentación sólo debe referirse a lo elementos incluidos en la interfaz de usuario.

Modelo de 7 capas.

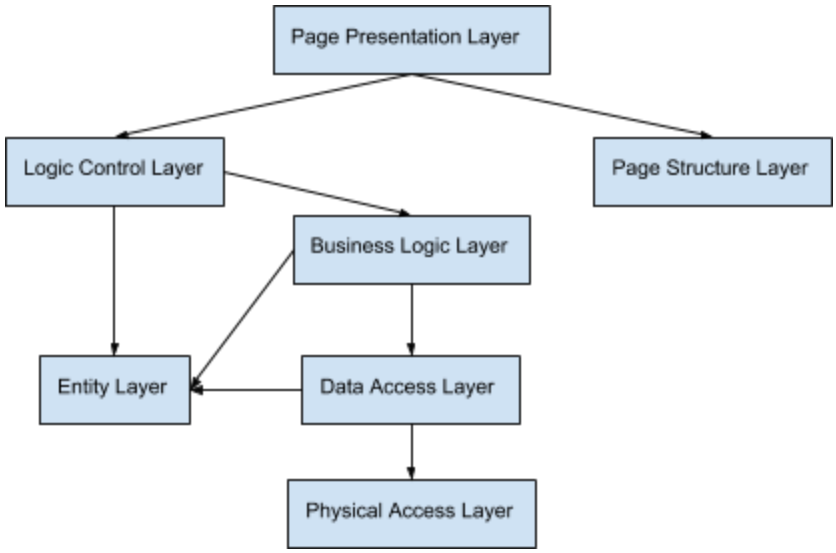
Con el pasar del tiempo y el avance de las tecnologías de desarrollo y comunicaciones, el modelo de tres capas fue dando paso a un modelo con elementos más chicos y cohesionados. Basado en la arquitectura de tres capas, se dividió a las capas de presentación y de datos. La DAL (Data Access Layer) fue dividida en capa de dato físicos (Physical Data Layer), capa de acceso a datos (Data Access Layer) y capa de entidad (Entity Layer). La UI fue dividida en la capa de estructura de página (Page Structure Layer), capa de control de lógica (Logical Control Layer) y capa de presentación de página (Page Presentation Layer).

Al usar este modelo de desarrollo, los desarrolladores solo se involucran con solo una capa de todas del sistema, cualquier implementación de una capa puede ser fácilmente reemplazada por una nueva realización, y se reducen las dependencias entre capas. Todos estos beneficios conducen a la estandarización y a la reusabilidad de cada capa.

A continuación se da una pequeña descripción de cada una de estas capas.

- Physical Data Layer: Esta incluye el modelado del negocio y el diseño de la base de datos.
- Data Access Layer: Incluye el diseño y la implementación de los componentes de acceso a datos.
- Entity Layer: Incluye el mapeo entre los objetos y los datos.
- Business Logic Layer: Es la parte esencial de todo el sistema, incluye el agregado, borrado, actualización y los método de selección de entidades.
- Page Structure Layer: Incluye elementos de página como ser código html, pero no los estilos que se incluyen en la Page Presentation Layer.
- Logical Control Layer: Ésta controla el contenido de las páginas determinando la disposición de la estructura de las páginas. Esta capa tiene acceso a la Business Logic Layer.
- Page Presentation Layer: Esta capa hace uso de div+css para alcanzar los requerimientos de usuario en la superficie visible de la aplicación. Esta se encuentra separada de la Page Structure Layer.

Un diagrama de las siete capas y sus relaciones se puede ver a continuación.



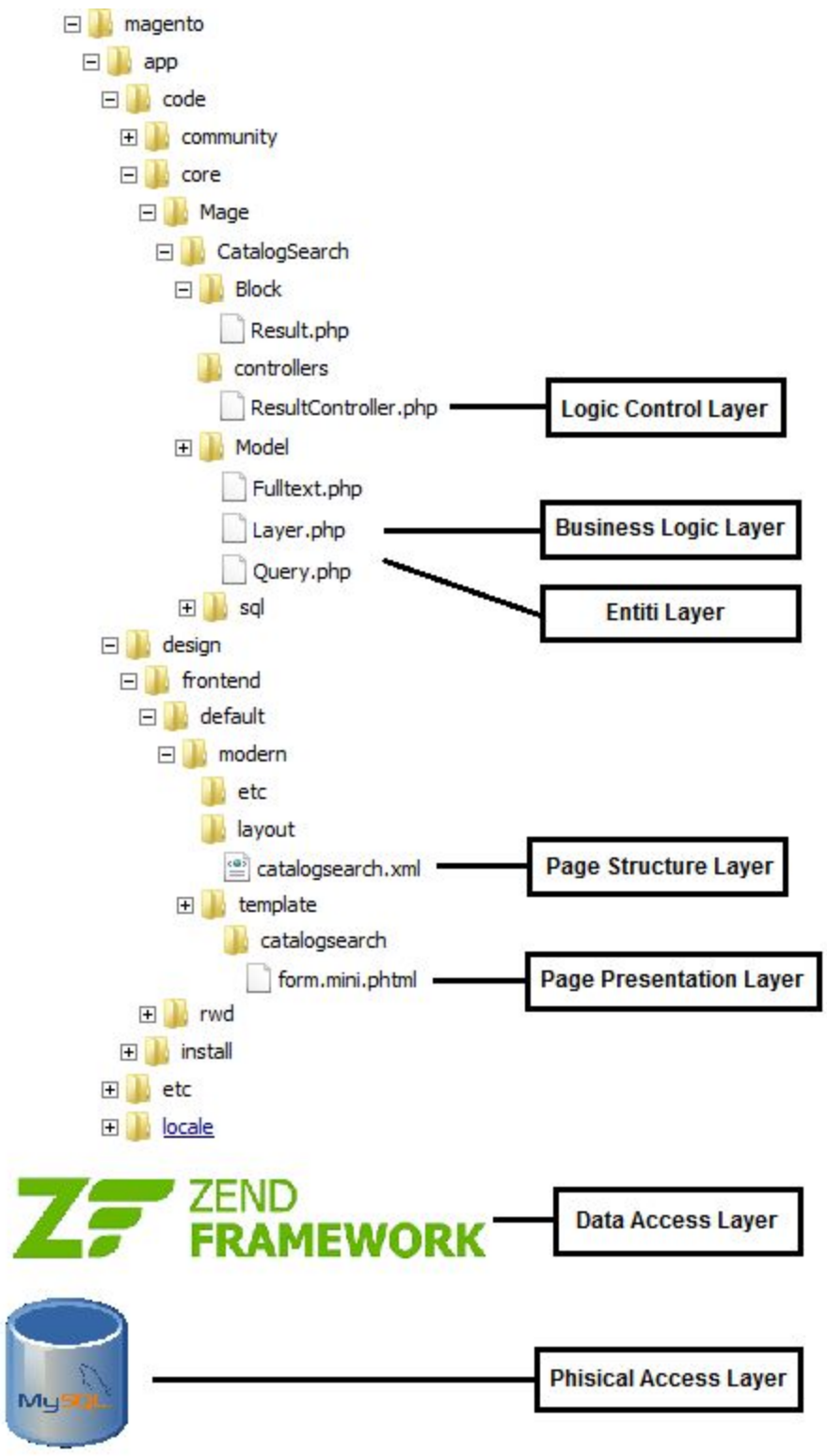
La plataforma de e-Commerce Magento implementa un modelo de 7 capas. Cada tecnología de desarrollo (ASP.Net, PHP, Java, etc) implementa este modelo con algunas diferencias, pero siguen el lineamiento general visto anteriormente. Magento tiene una estructura bastante delimitada en lo que refiere a localización y las convención de nomenclatura de los archivos que implementan las distintas funcionalidades de la plataforma.

En el gráfico que se puede ver abajo, vemos que la Physical Access Layer se encuentra implementada por el RDBMS MySql de Oracle.

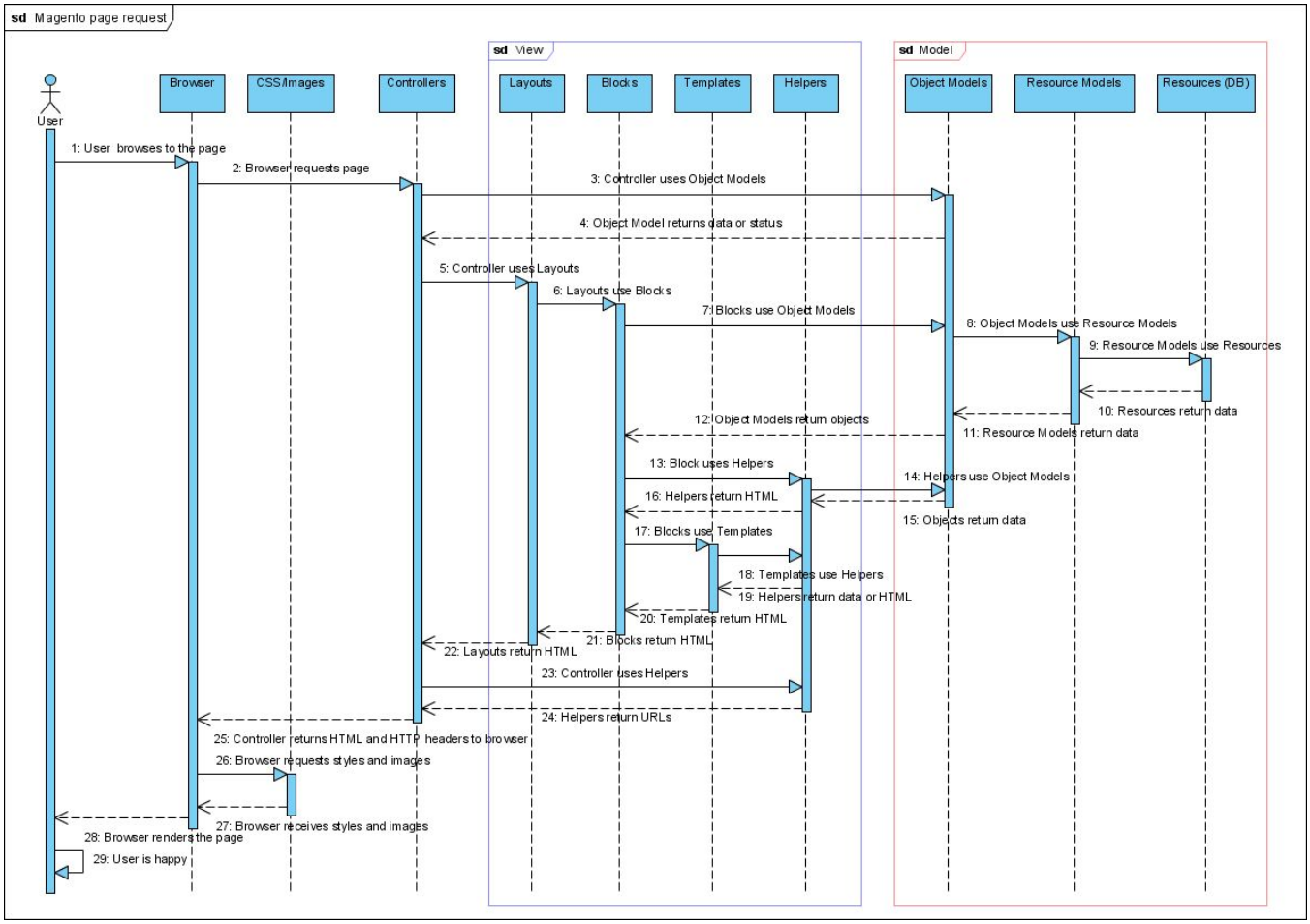
La Data Access Layer se implementa a partir de Zend Framework. Éste ofrece un conjunto de clases para acceder a los datos de la base de datos y servicios de XML-RPC, que es un protocolo de llamada a procedimiento remoto que usa XML para codificar los datos y HTTP como protocolo de transmisión de mensajes.

La Entity Layer y Business Logic Layer se encuentra implementada dentro de la carpeta Model de cada una de los módulos de Magento.

La Logic Control Layer se encuentra implementada por archivos cuyo nombre terminan en "Controller". Las clases contenidas en estos archivos son las encargadas de recepcionar los request enviados por el cliente y determinar el flujo de trabajo del sistema.



Uno de los patrones de diseño que se implementa en la mayoría de las arquitecturas de este estilo es el MVC (Model View Controller). Cada framework de desarrollo (java, .net, php, etc) tienen diferentes formas de llevar a la práctica este patrón. Y dentro de estos frameworks, Magento mismo tiene su forma particular. En el siguiente diagrama se puede ver como es la interacción entre los distintos artefactos ante la llamada a una página por parte del usuario.



En el diagrama se puede ver bien las tres capas del patrón MVC. Por un lado el controller, que en la imagen de la estructura de carpetas de Magento se encuentra en las carpetas homónimas dentro de cada módulo. La vista se encuentra distribuida en las carpetas que se encuentran dentro de la carpeta app/design y las carpetas Blocks que se pueden encontrar en cada módulo en caso de tener particularidades que no pueden ser puestas a nivel general dentro de la carpeta design. Y por último el Modelo, que se encuentra en la carpeta Model de cada módulo.

3.4. Bases de datos de sitios de e-Commerce

Uno de los principales requerimientos funcionales de una plataforma de e-Commerce es la flexibilidad para poder administrar un conjunto heterogéneo de productos. Esto significa que en la tienda puedan existir variedad de productos y cada uno es definido por un conjunto distinto de atributos. Por ejemplo, un producto puede tener el atributo "peso" mientras que otro puede tener "volumen", "talle", "color", etc. A su vez los tipos de productos pueden ser simples, configurables, agrupados, etc. Crear una tabla con cientos de columnas no es claramente la solución, porque la gran mayoría de las columnas tendrán valores nulos.

Para lograr esta flexibilidad a nivel de datos las plataformas de e-Commerce se apoyan en el modelo de datos EAV (Entity Attribute Value), también llamado "modelo de datos genérico", "modelado de base de datos vertical" ó "esquema abierto", el cual se implementa, en su forma más básica, con tres tablas que representan a las entidades, los atributos y los valores. Este modelo contrasta con el Row Modeling que es el que usualmente se utiliza en la gran mayoría de los sistemas que persisten sus datos en bases de datos relacionales.

Las diferencias principales entre estos dos modelos son:

- Una tabla row-modeled es homogénea en los hechos que describe. Mientras que una tabla EAV puede contener cualquier tipo de hechos.
- El tipo de datos de los valores de las columnas en una tabla row-modeled es predeterminado por la naturaleza de los hechos guardados. Por ejemplo una columna que contenga un campo monetario va a ser

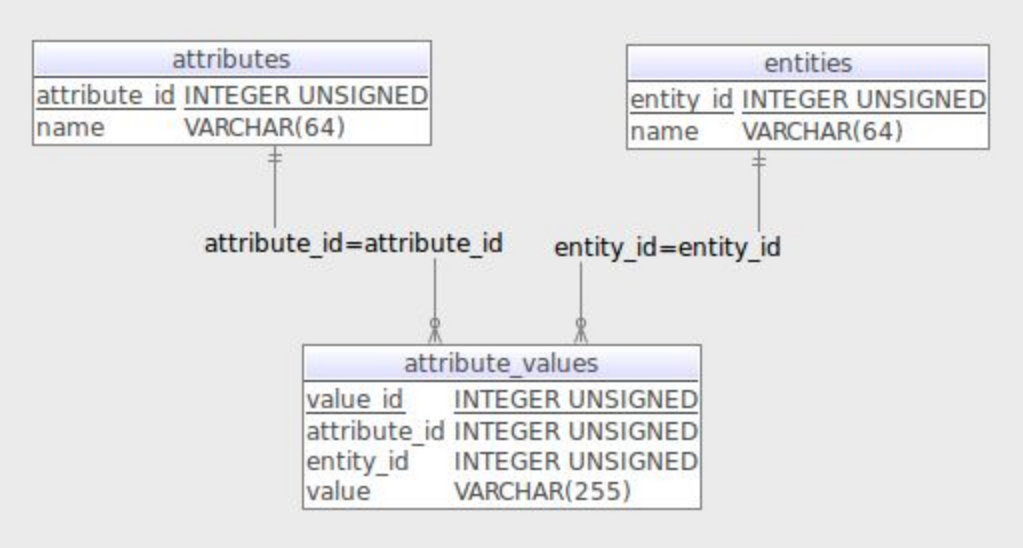
de tipo float o currency, dependiendo el tipo de datos que el motor de base de datos soporte. En contraste, en una tabla EAV, el tipo de dato de un valor de un registro en particular depende del atributo que contenga ese registro/fila. Por ejemplo, una registro de la tabla puede representar a un atributo monetario (currency/float) y el registro inmediato superior puede contener un atributo descripción (varchar).

Este modelo se utiliza para describir entidades en donde el número de atributos que pueden ser usados para describirlas es potencialmente vasto.

Estas tres tablas contienen por un lado la definición de las **Entidades** (tabla ENTITIES en el diagrama) la cual tiene la definición más básica de los objetos en la base de datos. Esta tabla puede contener además todos los atributos generales que se encuentran en todos los objetos del dominio de la aplicación.

En otra tabla se encuentran las definiciones de cada uno de los **Atributos** (tabla ATTRIBUTES en el diagrama) de todos los objetos del dominio de la aplicación. En el diagrama solo se ven los campos identificador único del atributo y su nombre, pero podría contener validaciones a hacer sobre los valores que pueden tomar los atributos ó el tipo de datos del atributo dentro de la aplicación que soporta.

Y por último está la tabla en donde se encuentran los **Valores** de cada uno de los atributos de los objetos de la aplicación (tabla ATTRIBUTE_VALUES en el diagrama).



Este modelo presenta la ventaja de la flexibilidad a la hora de acoger entidades con un vasto conjunto de atributos. Pero en contrapartida presenta ciertas **desventajas** con respecto al modelo row-modeled. Las mismas son:

- **Flacidez**: EAV es bueno en lo que refiere a flexibilidad, pero su implementación implica perder toda estructura. Por lo general, la fiabilidad de las funcionalidades integradas de base de datos tales como la integridad referencial se pierden. Para garantizar que una columna solo toma valores en un rango aceptable, dicha comprobación de integridad debe ser implementada dentro de la aplicación**.**
- **Consultas Ineficientes**: En casos en donde uno sólo necesita ejecutar una consulta simple retornando 20 columnas de una sola tabla en un clásico escenario row-modeled, en EAV uno tiene que lidiar con 20 self-joins, uno por cada columna. Esto hace que el código se torne ineligible y la performance vaya decayendo a medida que el sistema vaya creciendo.
- **Pérdida de funcionalidades**: Mucha de la maquinaria de las modernas bases de datos relacionales, por ejemplo funciones propias del motor de base de datos, se torna inútil y debe ser recreada por el equipo de desarrollo en caso de ser necesario. Por ejemplo, tablas del sistema, herramientas de representación gráfica de consultas, seguridad de datos puntuales, etc.

- **Otras herramientas standard son mucho menos útiles**: Los cursores en las funciones de base de datos no devuelven registros si es que los datos primero deben ser pivoteados. Funciones definidas por el usuario se tornan largas y dificultosas para desarrollar y depurar. Consultas SQL Ad-hoc toman mucho tiempo en escribirse y es costoso poder determinar las juntas (joins) necesarios para recuperar todos los datos necesarios.
- **El formato no es bien soportado internamente por el DBMS**: Los optimizadores de consultas SQL estándar no manipulan los datos correctamente para los datos formateados en el modo EAV y se puede perder mucho tiempo en ajustar la performance hasta llegar a un nivel aceptable de calidad.

En la siguiente figura se puede ver un pequeño ejemplo de cómo estarían alocados los datos en las tablas anteriormente descriptas. El ejemplo consta de tres prendas de vestir con un atributo en común (color) y otros atributos particulares a cada prenda (talles por prenda).

Entidades: Remera color roja, talle Medium;
Pantalón color azul, talle 32;
Zapato color blanco, talle 41;

attributes

attribute_id	name
1	talle_remera
2	talle_pantalon
3	talle_zapato
4	color

entities

entity_id	name
1	Remera
2	Pantalón
3	Zapato

attribute_value

value_id	entity_id	attribute_id	value
1	1	1	Medium
2	1	4	Rojo
3	2	2	32
4	2	4	Azul
5	3	3	41
6	3	4	Blanco

El término "Base de Datos EAV" se refiere a un diseño de la base de datos en el que una porción significativa de los datos es modelada como EAV. Sin embargo, más allá de que una base de datos sea descripta como "EAV-based", muchas otras tablas en el sistema son relacionales como tradicionalmente se las conoce.

3.4. Infraestructura de un sitio de e-Commerce

A desarrollar

3.5. Medios de Pago.

A continuación se detallan todos los tipos de pasarelas de pago existentes. La diferencia entre ellas radica principalmente en que sitio se hace la recolección de los datos y cómo se comunica la pasarela con el sitio eCommerce que lo utiliza.

Pasarelas de Pago Alojadas (Hosted Payment Gateways)

Proceso:

Una pasarela de pago alojada es aquella que lleva al cliente fuera de la página de checkout del sitio de eCommerce. Una vez que el usuario hace click en el botón de "Pagar" en el sitio web, el usuario es redirigido a la página del proveedor de servicios de pago (PSP, por sus siglas en inglés). En ésta página el usuario debe completar sus datos de pago. Cuando el cliente ha completado la transacción es redirigido nuevamente al eCommerce para completar el proceso de checkout.

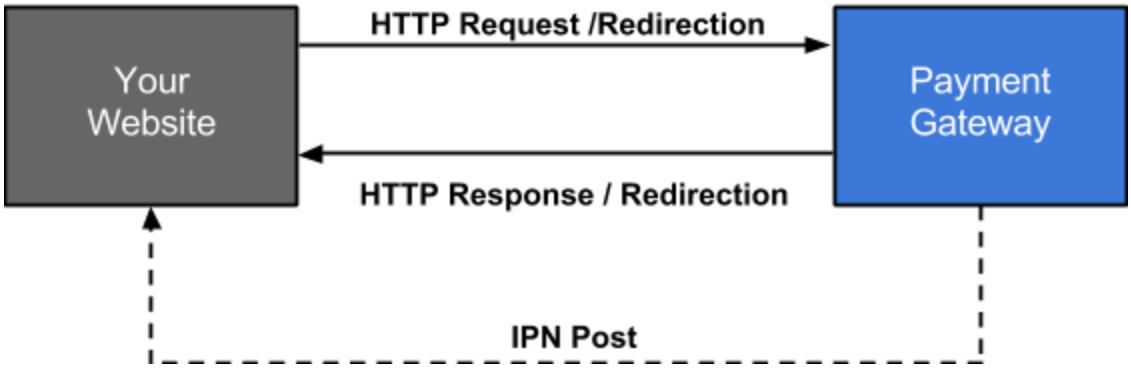
Otra opción utilizada por este tipo de pasarela es utilizar un iframe. En este caso el PSP crea un form (iframe) el cual es insertado en el sitio web del vendedor. Con este método, el vendedor acepta tarjetas de crédito y débito sin tener que capturar ni guardar información de las tarjetas en el sitio web. En cambio esta información de pago es recolectada utilizando un frame en línea (iframe). Éste form es alojado por el PSP, de esta forma, cuando el cliente llena los datos del formulario, el PSP recibe los datos.

Para pagos recurrente, un perfil es creado con la información del usuario, la cantidad de pagos, frecuencia, monto, etc. La pasarela de pago aplicará la deducción de los pagos recurrentes con la ayuda del perfil creado en la primera interacción y envía las notificaciones de los pagos al sitio web.

Las devoluciones y cancelaciones de los pagos deben ser administradas desde el sitio de la pasarela de pagos.

Notificación:

Este tipo de pasarela de pago hace uso de notificaciones para informar acerca de cualquier actividad de pago ejecutada. Para ello el eCommerce debe proveer de una URL mediante la cual la pasarela informe de los movimientos. Generalmente lo que se hace es enviar un identificador de notificación a esta URL y luego el sitio web se encarga de llamar a la API de la pasarela para obtener información acerca de la notificación pasada por parámetro anteriormente. Esta forma indirecta de manejar las notificaciones es más segura, dado que la pasarela solo proveerá de la información sensible a todos los requerimientos de servidores conocidos que estén incluidos dentro de una lista blanca de IPs.



Ejemplos: PayPal Standard, 2Checkout (Standard Checkout)

Ventajas:

- Seguridad: Los datos del tarjetahabiente son capturados en forma segura por el proveedor de la pasarela de pago (PSP en ingles).
- Simpleza: El PSP se encarga de la complejidad de la captura de los datos sensibles. El dueño de la tienda solo tiene que concentrarse en el negocio.
- Personalizable: El logo de la tienda puede ser agregado a la página de pago para que la experiencia del usuario no pierda la identidad de la tienda.

Contras:

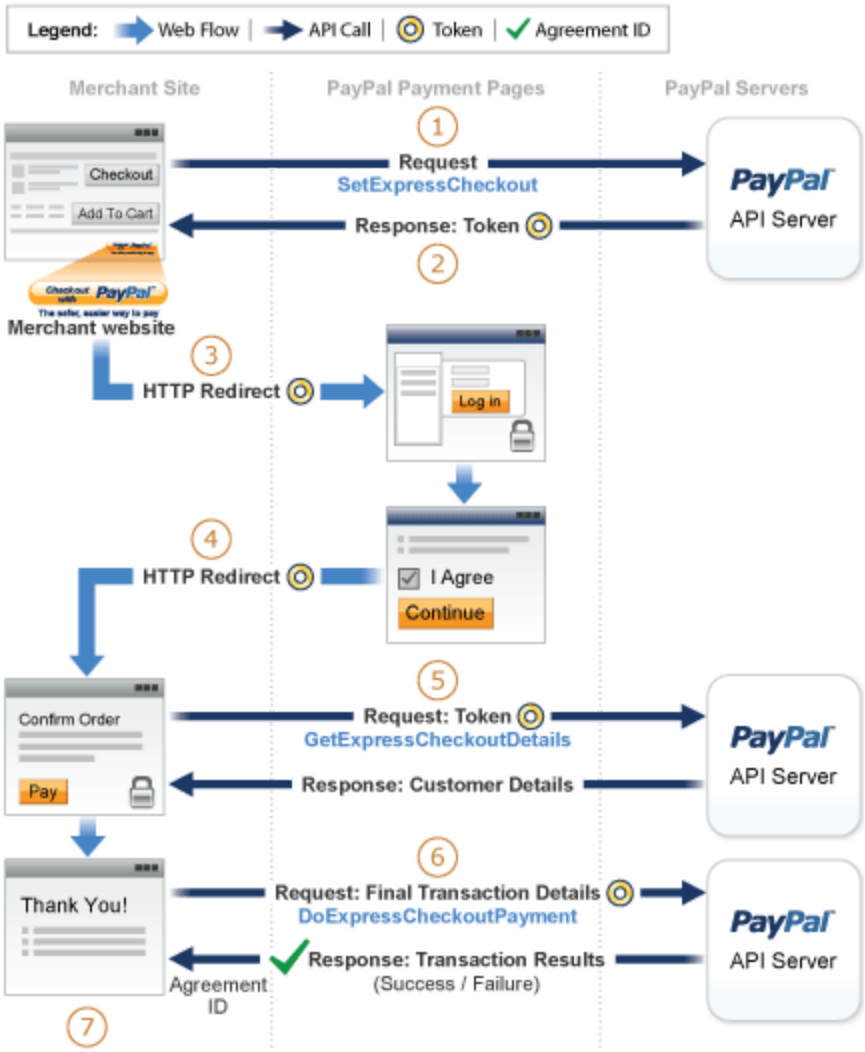
- Experiencia del usuario: La tienda no tiene el control de la experiencia de usuario de principio a fin.

Pasarelas de Pago Auto-Alojadas (Self Hosted Payment Gateways)

Proceso:

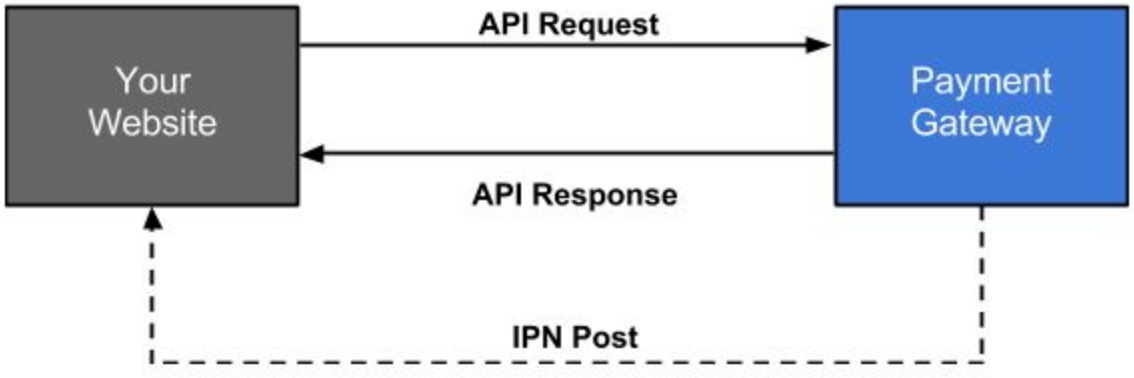
Para este tipo de pasarelas, es necesario recolectar los datos de pago del cliente en la misma tienda. Luego de obtenidos estos datos, los mismos son enviados a una URL provista por la pasarela de pago. Algunas de estas necesitan ser informadas en un formato definido por alguna clave hash o una clave de seguridad o clave secreta. Para obtener este token/hash/key el usuario debe efectuar un login dentro de la plataforma de la pasarela de pago, e inmediatamente volver a la tienda para continuar el proceso de recolección de datos de pago.

En caso de pagos recurrentes, los pagos sucesivos son ejecutados por la pasarela misma, enviando una notificación a la tienda acerca de la misma. El reembolso o la cancelación de los pagos se deben realizar desde el sitio de la pasarela de pago.



Notificación:

Las notificaciones son enviadas en forma silenciosa por la pasarela. Al momento de configurar la misma, es necesario especificar la URL a la cual las notificaciones llegarán. Cada vez que un pago es ejecutado, una notificación es enviada y el script de la tienda realiza los procedimientos especificados.



Pros:

- Fácilmente personalizable. La tienda tiene el control del checkout del inicio al fin, lo que hace que la experiencia de usuario esté totalmente integrada con la tienda.
- Experiencia del Usuario: Los compradores nunca abandonan la tienda, a lo sumo al momento de hacer login en la pasarela de pago, haciendo que la experiencia de compra sea más confidencial.

Contras:

- Seguridad: La tienda tiene que implementar medidas de seguridad para proteger los datos de los tarjetahabientes.

Ejemplo:

Paypal Payment Pro

Pasarelas de Pago no Alojadas / por API (API / Non Hosted Payment Gateways)

Proceso:

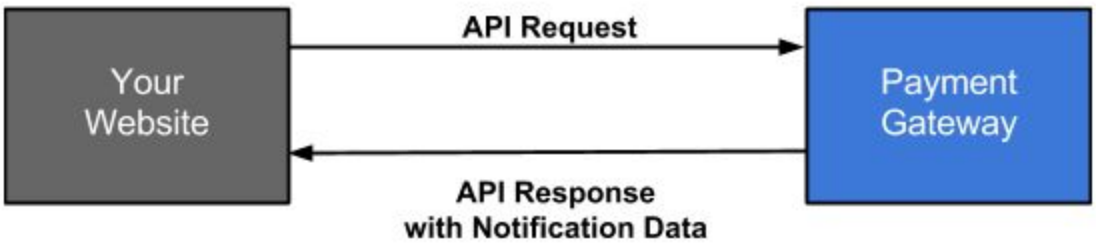
Algunos comerciantes quieren tener el total control de su proceso de checkout y no quieren sacar a los clientes de la página de checkout de su sitio. Para cumplir con ese deseo es necesario implementar una pasarela de pago NO ALOJADA. Ésta permite a los clientes ingresar la información de sus tarjetas de crédito o débito directamente en la página de checkout de la tienda y efectuar el proceso de pago utilizando las APIs de la pasarela de pago o utilizando algunos requests de HTTPS.

Estas pasarelas generalmente soportan los pagos recurrentes así como la corrección de pagos efectuados.

Notificaciones:

A partir de los datos ingresados, la tienda crea internamente el pago con el que llamará a la pasarela de pago. Estos datos pueden ser utilizados para crear un perfil en la pasarela de pagos para los pagos recurrentes, o solo para pagos únicos. Después de crear la llamada y enviada, la pasarela de pago envía la notificación en respuesta a la anterior llamada. Estas notificaciones, que pueden ser mensajes de confirmación o errores, son tratadas en la tienda y mostrada al cliente.

Algunas pasarelas de pago de este tipo proveen interfaces (API) para consultar por pagos, cancelaciones, reintegros, etc.



Ventajas:

- Flexibilidad: La tienda tiene el control total sobre la forma en como se visualiza la interfaz de usuario.
- Experiencia del Usuario: Los compradores nunca dejan el sitio web, impartiendo más confianza a la hora de completar el proceso de compra.
- Versatilidad: Al usar una API se puede integrar la solución de pago sobre internet con cualquier dispositivo conectado a internet (teléfonos móbiles, tablets, etc.

Desventajas:

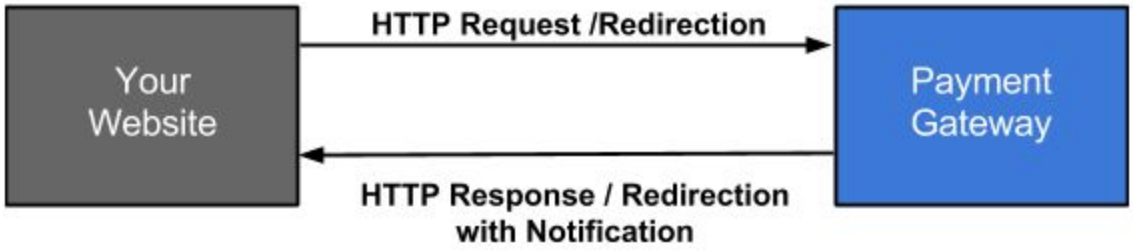
- Seguridad: La responsabilidad por cumplir las normas PCI DSS es de la tienda eCommerce. La mayoría de las veces, dado que el cumplimiento de las normas PCI DSS son difíciles de lograr, estas normas no se cumplen o son parcialmente cumplidas.
- Servicio: El vendedor tiene que adquirir una certificación SSL para su sitio para mayor seguridad.

Integración local con banco

Proceso:

Estas pasarelas de pago son también consideradas como pasarelas de pago alojadas que trabajan en forma directa. El comprador es redireccionado al sitio web de la pasarela de pago, en donde él debe llenar los detalles del pago y datos del contacto. Luego de que se realiza el pago, el comprador es redireccionado nuevamente a la tienda y las notificaciones correspondientes a la operación son enviadas dentro de esta redirección.

Este tipo de pasarelas de pago no soportan pagos recurrentes, reembolsos o cancelaciones. La tienda debe hacerlos manualmente en el sitio de la pasarela de pago.



Ventajas:

- Simpleza: Es una buena opción para negocios pequeños, que necesitan opciones de pago de uso único.

Desventajas:

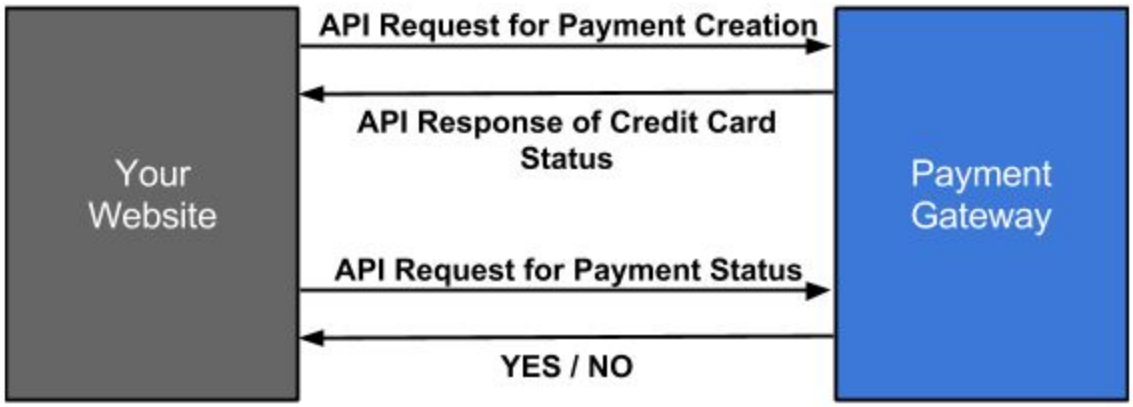
- Sin funcionalidades extras: No tienen algunas de las funciones básicas de las demás pasarelas, como ser reembolsos y pagos recurrentes.

Pasarela de pago directa (Direct Payment Gateway)

Proceso:

Alguno de los procesos de pago no soportan notificación instantáneas de pagos. Estas pasarelas de pago crean perfiles y deduce el monto requerido desde las tarjetas de crédito del comprador en forma programada, pero no informan a la tienda que realizó la orden de pago. La pasarela solo informa si la tarjeta de crédito fue aprobada o no. Es por esto que la tienda tiene que realizar consultas a intervalos regulares a el procesador de pagos para saber si el pago requerido ha sido recibido o no.

Generalmente una transacción de pago consta de varias y sucesivas llamadas a la API de la pasarela de pago. Estas llamadas son distintas y cada una cumplen cometidos diversos. Estas interacciones sirven para autorizar montos de pagos, generar capturas diferidas del monto anteriormente autorizado, reembolso y cancelación de pagos.



Ejemplo: Payflow Pro

Ventajas:

- No es necesario navegar al sitio de la pasarela de pago.

Desventajas:

- Estas pasarelas no proveen el servicio de pago en forma instantánea.
- Se requiere un mayor esfuerzo técnico dado que se necesita realizar múltiples validaciones contra la pasarela de pago.

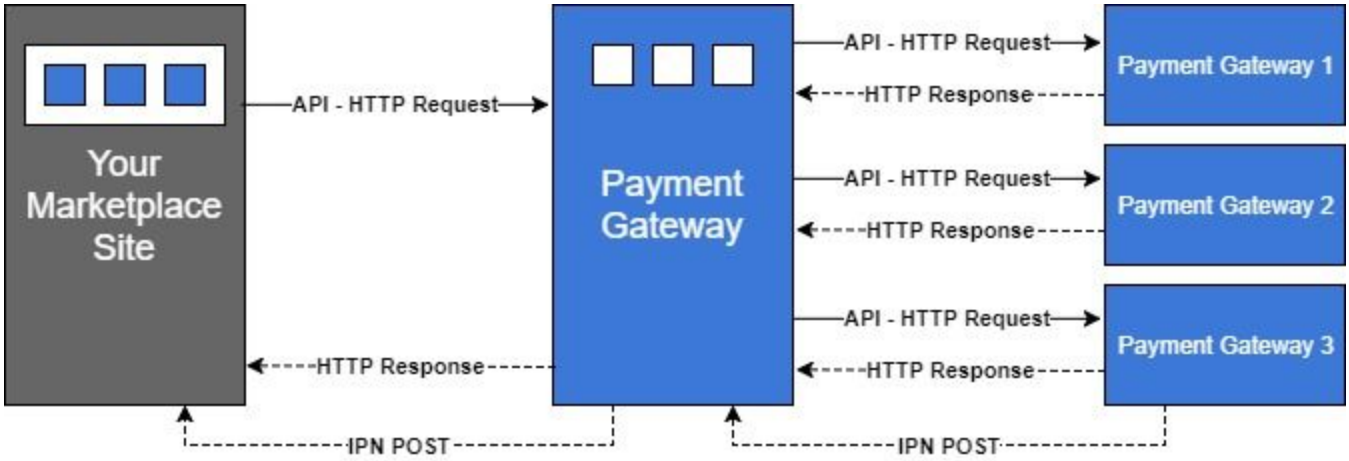
Pasarelas de Pago basada en Plataforma (Platform Based Payment Gateway)

Estas pasarelas permiten diferir una misma transacción en múltiples pagos a distintas pasarelas. Está pensada para dar servicios a los eCommerces del tipo Marketplace. Permitiendo a los usuarios realizar un solo proceso de checkout más allá de que los productos comprados pertenezcan a distintos comercios.

Proceso:

El usuario selecciona los productos en la tienda (marketplace) y procede al checkout. La pasarela de pago recibe en un mismo paquete de datos información referida a los distintos productos comprados y de sus correspondientes vendedores. A partir de acá, la pasarela hace una división (split) de la información de pago. Por cada conjunto de productos de un mismo comerciante, opera individualmente con la pasarela de pago correspondiente a cada uno de los comercios.

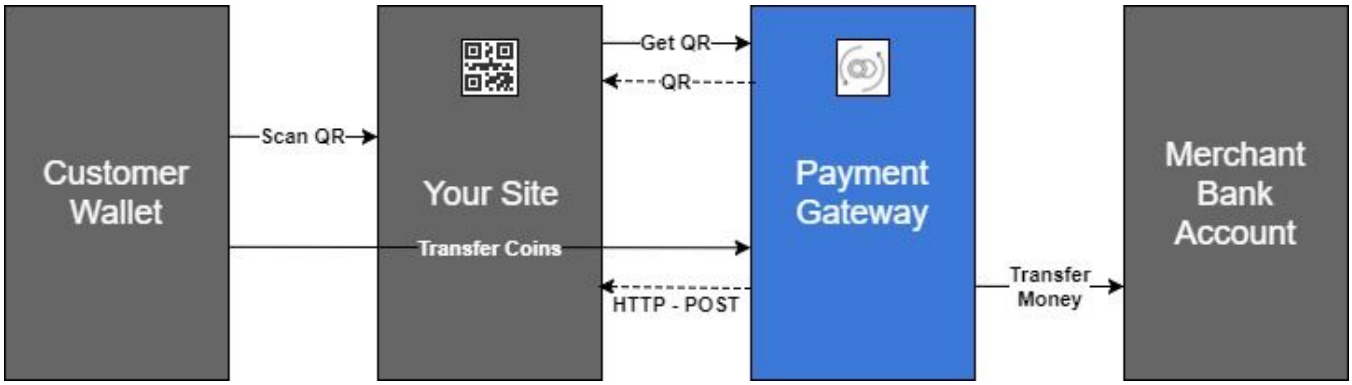
Se recopila la información de resultado y se devuelve el paquete de resultados a la tienda.



Pasarelas de Pago basada criptomonedas

Proceso:

El usuario ingresa al proceso de checkout una vez que ha seleccionado todo sus productos. El monto de la orden es informado al gateway el cual genera un código QR o una dirección de transferencia y lo envía a la tienda. El cliente escanea este código con su billetera de criptomonedas y procede a realizar el pago. Esta transacción se realiza sobre una billetera de la pasarela de pago. La cual transforma el monto de las criptomonedas en dinero circulante según elección del propietario de la tienda.



Ventajas:

- Darle al cliente una opción más de pago
- La tienda no tiene que involucrarse con la complejidad de las operatorias con criptomonedas.

Desventajas:

- El proceso de pago con criptomonedas es poco conocido.
- La volatilidad del precio de las criptomonedas hace necesario dificil saber cual va a ser la cotización a utilizar.

Estandar PCI

Cuando una infraestructura que contiene datos sensibles (datos de tarjetas de pago) es vulnerada, múltiples aspectos del negocio se ven afectados. Y las pérdidas se pueden sentir en toda la empresa. Los principales inconvenientes son:

-Pérdida de confianza de los clientes.
-Pérdida de ventas y ganancias
-Menos uso del comercio online por el temor de los clientes.
-Degradación de la marca y caída del valor de las acciones.
-Multas y penalizaciones por no cumplir con los estándares de seguridad.
-Costos más altos para cumplir con el estándar PCI luego de un evento de robo de información.
-Inhabilitación para aceptar pagos con tarjetas.
-Costos de reinstalación de medios de pago con tarjeta.
-Costo por la resolución de disputas.
-Costos legales

Es por eso que la industria, por intermedio de los estándares PCI, ha fijado procesos y condiciones estándares para que estos datos sean manipulados en forma segura y de esa forma hacer que el medio de pago no pierda credibilidad ante los clientes.

Se entiende por estandar PCI al Estandar para la Seguridad de Datos para la Industria de los Pagos por Tarjeta, Payment Card Industry Data Security Standard PCIDSS son sus siglas en inglés, pero conocido más comunmente como PCI.

Y es un conjunto de prácticas,requerimientos técnicos y operativos que tienen por objetivo hacer que los datos sensibles de los dueños de tarjetas de pago no se vean comprometidos en transacciones realizadas sobre internet.

El estándar es administrado por un foro global y abierto llamado Consejo de Estándares de Seguridad para la Industria de Pagos por Tarjeta (Payment Card Industry Security Standards Council - PCI SSC). El concejo fue fundado en el año 2006 por las mayores marcas de tarjetas: American Express, Discover Financial Services, JCB International, Mastercard Worldwide y Visa Inc.

La PCI DSS se aplica a todas las entidades que participan en el procesamiento de las tarjetas de pago, entre las que se incluyen comerciantes, procesadores, adquirentes, entidades emisoras y proveedores de servicios. La PCI DSS se aplica a todas las entidades que almacenan, procesan o transmiten datos del titular de la tarjeta y/o datos confidenciales de autenticación. Los datos del titular de la tarjeta y los datos de autenticación confidenciales se definen de la siguiente manera:

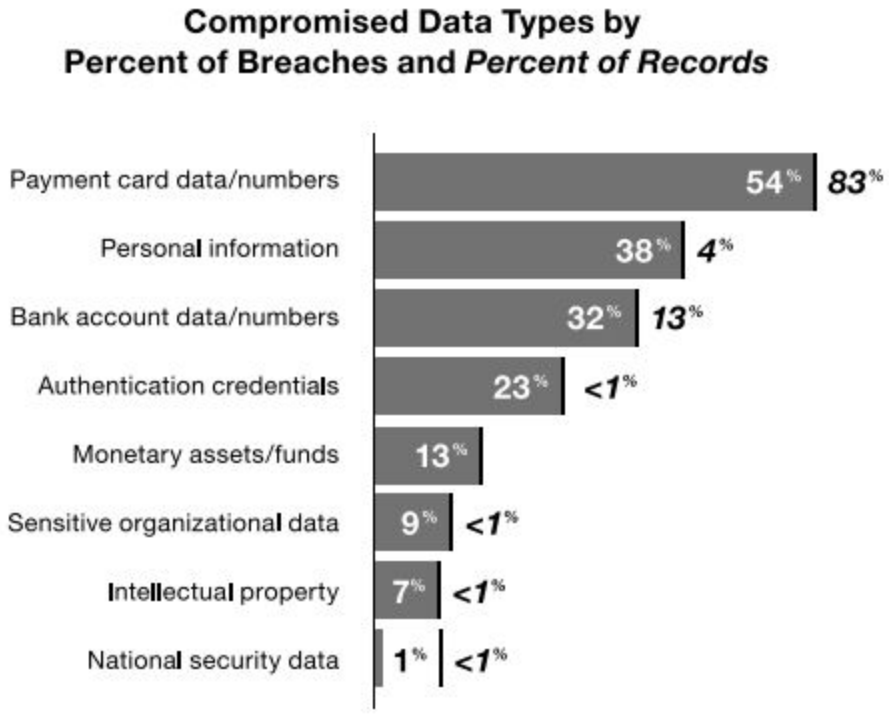
Datos de cuentas	
Los datos de titulares de tarjetas incluyen:	Los datos confidenciales de autenticación incluyen:
● Número de cuenta principal (PAN) ● Nombre del titular de la tarjeta ● Fecha de vencimiento ● Código de servicio	● Contenido completo de la pista (datos de la banda magnética o datos equivalentes que están en un chip) ● CAV2/CVC2/CVV2/CID ● PIN/Bloqueos de PIN

El número de cuenta principal es el factor que define los datos del titular de la tarjeta. Si el nombre del titular de tarjeta, el código de servicio y/o la fecha de vencimiento se almacenan, procesan o transmiten con el PAN (número de cuenta principal) o se encuentran presentes de algún otro modo en el entorno de datos del titular de la tarjeta, se deben proteger de conformidad con los requisitos aplicables de la PCI DSS.

PCI define tres estándares Uno está diseñada para comerciantes y otras organizaciones que aceptan tarjetas de pago; otro para fabricantes de dispositivos de tarjeta utilizados en el punto de venta; y otro para desarrolladores de software de aplicaciones de tarjetas de pago. Cada uno de estos estándares comparten el mismo objetivo primordial: proteger los datos de tarjeta al tarjeta habiente. Estos tres estándares son los siguientes:

- **PCI Data Security Standard (PCI DSS):** es el corazón de todo el estándar, el cual aplica principalmente para comerciantes y procesadores de pagos.
- **Payment Application Data Security Standard (PA-DSS):** esta sección del estándar es para los desarrolladores de software que construyen sistemas que aceptan y procesan pagos con tarjeta.
- **Personal Identification Number (PIN) Transaction Security Requirements (también llamado PTS):** esta sección aplica a los fabricantes de dispositivos para pago por tarjetas utilizados en los puntos de venta.

El estándar para seguridad de datos PCI tiene dos tipos de herramientas o controles para paliar las vulnerabilidades antes mencionadas. Estas son tecnología y procesos.
Tecnología para dar seguridad consiste en software, hardware y servicios de terceros usados para implementar sistemas construidos específicamente para proteger los datos de las tarjetas en múltiples formas.
Los **procesos de seguridad** son conjuntos de procedimientos operacionales usados para implementar y mantener la protección, los cuales pueden o no requerir de un tipo particular de tecnología de seguridad. Esto significa que estos procesos pueden ser definidos como simples medidas operacionales (actividades) llevadas a cabo por los integrantes de la organización .



-Los datos de tarjetas de crédito es la información que más es robada-

The

Payment Card Industry Data Security Standard is the author-
ized program of goals and associated security controls and

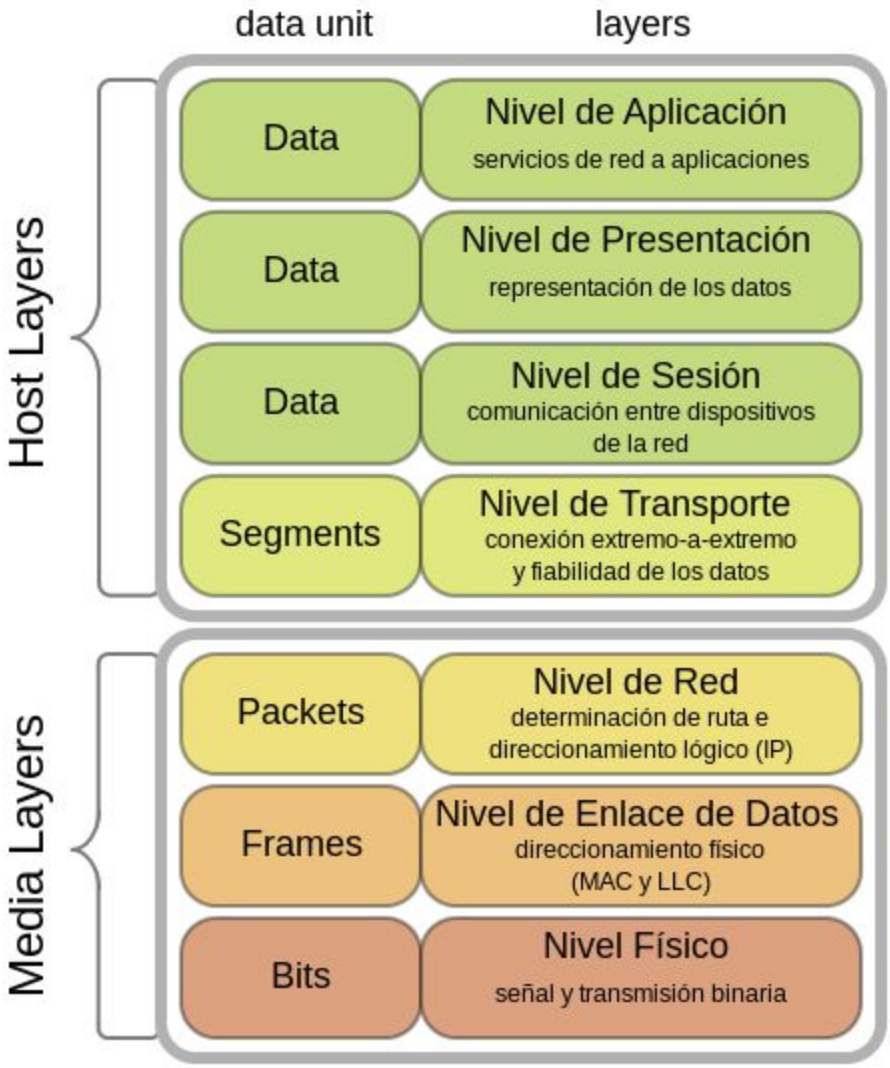
processes that keep payment card data safe from exploitation.

3.6. Seguridad en un sitio e-Commerce.

Básicamente, la seguridad en las aplicaciones de e-Commerce se implementan en distintas capas del modelo ISO. Por un lado, la seguridad en lo que refiere al frontend del sitio se implementa mediante SSL en la capa de transporte. Y la seguridad a nivel de instancias que componen la infraestructura del sitio se implementa mediante IPSec lo que permite crear una VPN entre los distintos nodos (web server, base de datos, etc).

Sirviendo como primera línea de defensa, los firewalls son comúnmente empleados para proteger una intranet (red privada) contra posibles ataques desde internet (red pública). Estos típicamente controlan la admisión de paquetes que intentan entrar en una red. El Internet Protocol (IP) original no presenta características de seguridad. Como opción, IPSec provee servicios de autenticación y encriptación a paquetes IP. Usando IPSec, las empresas y sus socios pueden construir una red privada virtual sobre la pública internet.

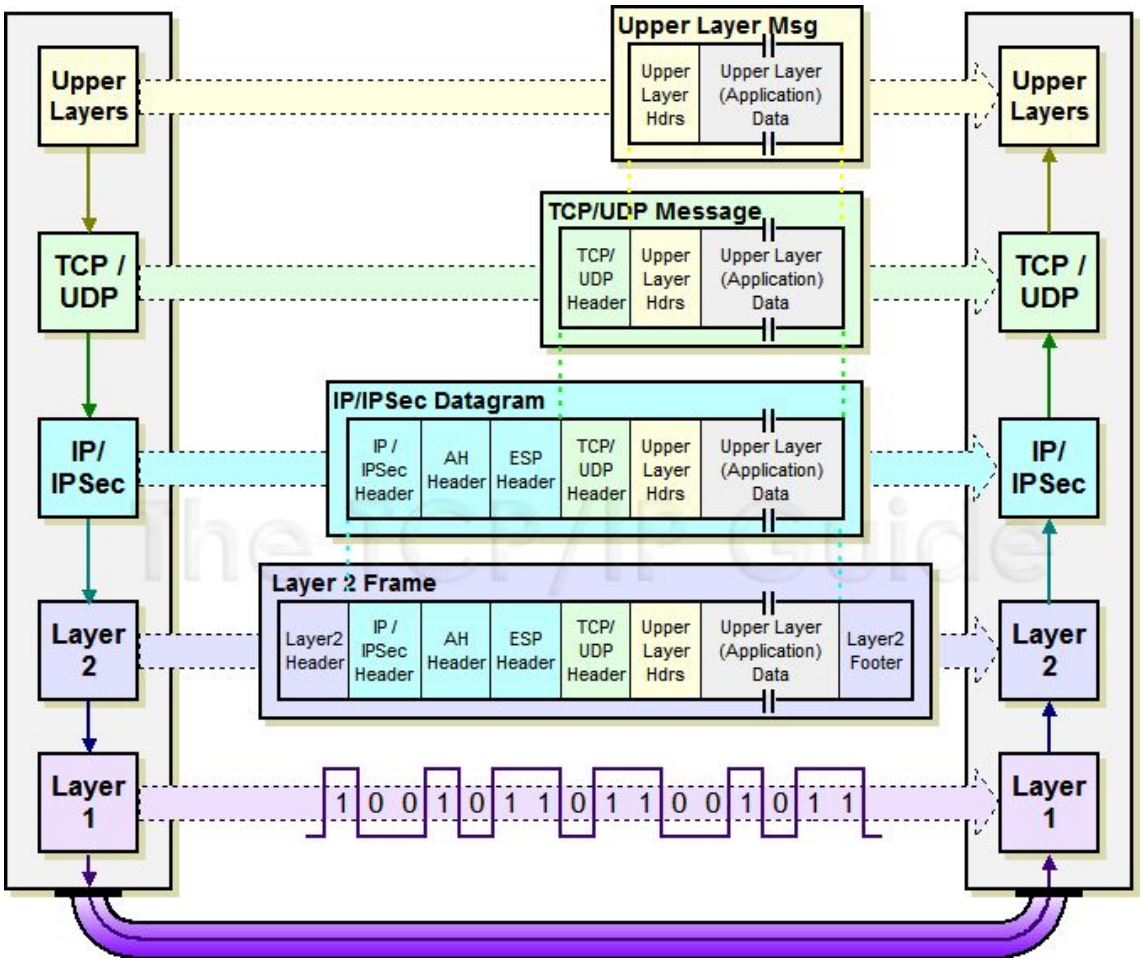
En la capa de transporte, el protocolo Secure Socket Layer (SSL) es usado para dar seguridad a datos transferidos entre dos computadores particulares para HTTP. Típicamente, SSL es usado para transferir datos sensibles (e.g. información de tarjetas de créditos) entre un cliente web y un servidor web. Es por eso que SSL es extremadamente importante en B2C e-commerce. Combinando firewalls, IPSec y SSL, se puede construir un sistema muy seguro de e-Commerce sobre internet.



Modelo OSI (Open System Interconnection)

3.6.1 Protocolo IPSec.

Como una opción segura al actual IP, IPsec soporta servicios de autenticación y/o encriptación en la capa de red. Como se puede ver en la SIGUIENTE FIGURA, esto se hace agregando un encabezado IPSec entre el encabezado de IP y el IP "PAYLOAD". El encabezado es insertado tanto para una computadora de usuario final que soporta IPSec, como para un gateway compatible con IPSec. Este nuevo encabezado IPSec provee la protección necesaria.



Existen dos tipos de encabezados IPSec o servicios IPSec, llamados Authentication Header (HA) y el Encapsulating Security Payload (ESP). El AH verifica la identidad de un paquete IP y asegura la integridad del contenido. En otras palabras, si el contenido es alterado, el receptor puede detectarlo entonces se puede tomar una acción apropiada (e.g. pedir al emisor que vuelva a retransmitir el paquete). Sin embargo, el contenido del paquete no se mantiene confidencial porque el paquete no está encriptado.

A costa de un tiempo de proceso más largo, ESP provee un servicio de encriptación y un servicio de autenticación opcional. Existen dos modos de operación para ambos, AH y ESP, llamados el modo transporte y el otro el modo túnel.

Para el modo transporte, los datos upper-layer (i.e. datos sobre la capa de red) estan protegidos. Ésto es usualmente usado si el usuario final puede soportar IPSec. Para el modo tunel, la protección cubre todo el paquete. En este caso el usuario final no necesita soportar IPSec porque una puerta de enlace que implemente IPSec es empleada para aplicar la protección requerida.

Luego de implementar IPSec entre dos dispositivos, estos necesitan configurar una asociación de seguridad (SA, siglas en inglés) que define una relación de seguridad simple entre éstos. Esto significa, si dos host X e Y quieren utilizar IPSec en ambas direcciones (X a Y, Y a X), se necesitan configurar dos SAs. Nótese que la configuración de las dos SAs pueden ser diferentes. Así, por consiguiente, los host pueden usar diferentes algoritmos criptográficos.

Entre otra información, cada SA define básicamente:

-la protección requerida (AH o ESP)

-los métodos de encriptación y/o autenticación

-las correspondientes llaves (keys) para utilizar las funciones criptográficas.

3.6.2 Protocolo SSL

Secure Socket Layer, ó SSL, es un protocolo que se utiliza para comunicarse a través de internet en un modo seguro. La tecnología SSL se basa en el concepto de criptografía de llave pública o criptografía asimétrica. En un escenario de encriptación normal, las dos partes comunicantes comparten una contraseña o clave, y esta es usada tanto para encriptar como para desencriptar mensajes. Mientras que este es un método simple y eficiente, éste no resuelve el tema de dar la clave a alguien que aún no se conoce o en quien se confía.

En la criptografía de llave pública, cada parte tiene dos llaves, una pública y una privada. La información encriptada con la llave pública de una persona, solo puede ser desencriptada con la llave privada de la misma persona y viceversa. Cada usuario expone públicamente su llave pública, pero mantiene a resguardo su llave privada.

Proceso de obtención de un certificado SSL

Supongamos que la empresa XYZ sa. quiere asegurar su proceso de checkout de clientes, administración de cuentas y correo interno en su sitio xyz.com.

Paso 1: XYZ crea un Certificate Signing Request (CSR) con alguna herramienta para tal efecto, como ser openssl.exe. En el proceso una llave privada es generada.
Paso 2: XYZ se dirige a una autoridad tercera parte certificante (Certificate Authority) como podría ser Trustwave o Verisign. Ésta toma el CSR y valida a XYZ en un proceso de dos pasos. Primero se valida que XYZ tenga control sobre el dominio xyz.com y que XYZ sa. es una organización oficial listada en los registros públicos del gobierno del país correspondiente.
Paso 3: Cuando el proceso de validación está completo, la autoridad certificante (CA) le a a XYZ una llave pública encriptada con una llave privada de la autoridad certificante (CA).
Paso 4: XYZ instala los certificados en sus servidores web.

Comunicación de un cliente con un servidor utilizando SSL

Paso 1: Un cliente realiza una conexión a xyz.com por intermedio de un puerto SSL, generalmente el 443. La conexión es por intermedio de https y no de http.
Paso 2: xyz.com envía su llave pública al cliente. Una vez que el cliente la recibe, el browser decide si debe procesarlo o no en función de las siguientes condiciones:

-La llave pública de xyz.com no debe estar expirada.
-La llave pública de xyz.com debe ser solo para xyz.com.
-El cliente debe tener la llave pública de la Autoridad Certificante instalada en su browser. Si el cliente tiene la llave de la Autoridad Certificante verificada, entonces se puede decir que el browser realmente se está comunicando con XYZ sa.

Paso 3: Si el cliente decide dar por válido el certificado, entonces el cliente enviará a xyz.com su llave pública.
Paso 4: xyz.com creará un hash único y lo encriptará utilizando la llave pública del cliente y la llave privada de xyz.com, y lo devuelve al cliente.
Paso 5: El browser del cliente desencriptará el hash. Este proceso solo se hace para mostrar que xyz.com envía el hash y solo el cliente es capaz de leerlo.
Paso 6: Cliente y website ahora pueden intercambiar información en forma segura.

3.6.3 Construyendo un e-Commerce seguro

La siguiente figura muestra una visión esquemática de este sistema seguro. El centro de este este sistema es la intranet del e-Commerce. Es una red privada consistente en el servidor web, el servidor de emails y otros servidores y computadoras. El tráfico proveniente de internet debe circular a través del proxy de aplicación y el control de acceso de usuarios a internet.

El e-Commerce está conectado con otras empresas socias, oficinas de sucursales y empleados con dispositivos móviles utilizando extranets o redes privadas virtuales (VPN).

Para ello esencialmente, se establecen túneles IP para proveer transferencia de datos segura sobre internet. El público en general puede acceder al e-Commerce a traves de internet. Esto quiere decir que necesitan acceder a los servidores web o de email. Los Firewalls son empleados para restringir a los clientes acceder solo a esos dos servidores. Para lograr un mayor nivel de seguridad, la configuración de DMZ puede ser usada.

Cuando los clientes envían información sensitiva al e-Commerce como datos de tarjetas de crédito, el servicio de SSL es usado para asegurar transferencia de datos segura. En este caso, el servidor de pagos implementa SSL y contiene un certificado digital firmado por una autoridad certificante (CA). Con todo esto en su lugar, puede ser construido un e-Commerce seguro.