



UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

TITLE UNKNOWN

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL COMPUTACIÓN

ÁLVARO BALMACEDA CELEDÓN

PROFESOR GUÍA:
NELSON BALOIAN

SANTIAGO DE CHILE
DICIEMBRE 2014



UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

TITLE UNKNOWN

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL COMPUTACIÓN

ÁLVARO BALMACEDA CELEDÓN

PROFESOR GUÍA:
NELSON BALOIAN

MIEMBROS DE LA COMISIÓN:
UNKNOWN COGUA
UNKNOWN INTEGRANTE

SANTIAGO DE CHILE
DICIEMBRE 2014



UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

TITLE UNKNOWN

ÁLVARO BALMACEDA CELEDÓN

COMISIÓN EXAMINADORA

CALIFICACIONES

Labor y Nombre

Nota (número)

Nota (texto)

Firma

Profesor Guía

Nelson Baloian:

.....

.....

.....

Profesor Co-Guía

UNKNOWN COGUIA:

.....

.....

.....

Profesor Integrante

UNKNOWN INTEGRANTE:

.....

.....

.....

Nota Final

Exámen de Título

.....

.....

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL COMPUTACIÓN

SANTIAGO DE CHILE

DICIEMBRE 2014

RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL COMPUTACIÓN
POR: ÁLVARO BALMACEDA CELEDÓN
FECHA: DICIEMBRE 2014
PROF. GUÍA: NELSON BALOIAN

TITLE UNKNOWN

Resumen

Dedicado a...

Agradecimientos

Quiero agradecer...

Índice general

1. Introducción	6
1.0.1. <i>E-Commerce</i> en la actualidad	9
1.0.2. <i>E-Commerce</i> en el futuro	9
1.1. Motivación	10
1.2. Alcances y Objetivo General	11
1.3. Objetivos Específicos	11
1.4. Características deseables	11
1.5. Estructura de la memoria	14
2. Estado del Arte	15
2.0.1. OpenCart	15
2.0.2. PrestaShop	16
2.0.3. Magento Community Edition	16
2.0.4. Zen Cart	17
2.0.5. Spree Commerce	17
2.0.6. Drupal Commerce	18
2.0.7. osCommerce	18
2.0.8. simpleCart	19

2.0.9. WooCommerce	20
2.0.10. WP e-Commerce	20
2.0.11. Jigoshop	21
2.1. Justificación para el desarrollo del proyecto	21
2.1.1. Base de datos	21
2.2. Tecnologías	22
2.2.1. Base de datos	22
2.2.2. SQL and the Relational Model	22
2.2.3. MongoDB	24
2.2.4. Angularjs	24
2.2.5. Bootstrap	25
2.2.6. Nodejs	25
2.2.7. CoffeeScript	25
2.2.8. Grunt	25
2.2.9. Docker	25
2.3. MongoDB y E-Commerce [25]	26
2.3.1. <i>Catalog Management</i>	26
2.3.2. Shopping Carts and Orders	28
2.3.3. <i>Querying Orders</i>	29
2.3.4. <i>Aggregation</i>	29
2.3.5. Updating Orders	30
2.3.6. Inventory	31
2.3.7. <i>Transactions, Consistency y Durability</i>	32
2.3.8. <i>Scalability</i>	32

2.3.9. Conclusion	33
3. Conclusiones	35
3.1. Trabajos Futuros	35
Glosario	36
Bibliografía	37
A. Apendice A	I
B. Teoria de Grafos	II
C. Catalog management handled with relational databases	III

Índice de figuras

2.1. OpenCart <i>website</i> [12].	15
2.2. PrestaShop <i>website</i> [14].	16
2.3. Magento <i>website</i> [9].	17
2.4. Zen Cart <i>website</i> [21].	17
2.5. Spree Commerce <i>website</i> [16].	18
2.6. Drupal Commerce <i>website</i> [5].	18
2.7. osCommerce <i>website</i> [13].	19
2.8. simpleCart <i>website</i> [15].	19
2.9. WooCommerce <i>website</i> [18].	20
2.10. WP e-Commerce <i>website</i> [20].	20
2.11. Jigoshop <i>website</i> [8].	21
2.12. <i>Schemas</i> de un producto.	27
2.13. <i>Vertical scaling</i> o <i>Scaling up</i>	33
2.14. <i>Horizontally scaling</i> o <i>Scaling out</i>	33
C.1. schemas of Magento e-commerce framework.	III
C.2. schemas of the Apache's OfBiz.	IV

Índice de tablas

2.1. Comparación entre diferentes opciones <i>e-Commerce</i>	22
2.2. Resumen NoSQL vs. SQL	24
2.3. NoSQL vs. SQL en relación a <i>e-Commerce</i>	26

Capítulo 1

Introducción

Historia de e-Commerce

Una de las actividades más populares en la Web es comprar. La razón es por que tiene un encanto especial - puedes comprar en tiempo libre, cualquier momento, incluso usando pijamas. Literalmente cualquiera puede tener páginas para mostrar sus productos y servicios.

La historia de e-Commerce se remonta a la invención de la básica noción de “vender y comprar”, electricidad, cables, computadores, modems, y la Internet. E-Commerce se hace posible en 1991 cuando la internet estuvo disponible para uso comercial. Desde entonces miles de negocios se han establecido en sitios webs.

En un inicio, el término e-Commerce se refería al proceso de ejecución de transacciones electrónicas comerciales con la ayuda de tecnologías líderes tales como Electronic Data Interchange (EDI) y Electronic Funds Transfer (EFT) las cuales dieron la oportunidad a los usuarios para intercambiar información de negocios y realizar transacciones electrónicas. La oportunidad para utilizar estas tecnologías surgió a finales de 1970s y permitió a los negocios de las compañías y organizaciones enviar documentación electrónica comercial.

Aunque la Internet comenzó a ganar popularidad entre el público general en 1994, tomó aproximadamente cuatro años desarrollar protocolos de seguridad (por ejemplo HTTP) y SSL los cuales permitieron un acceso rápido y conexiones persistentes a la Internet. En 2000 un gran número de empresas comerciales en los Estados Unidos y Europa Occidental presentaron sus servicios en la World Wide Web. En ese entonces el significado de la palabra e-Commerce fue cambiado. Las personas comenzaron a utilizar el término e-Commerce como el proceso de compra de productos y servicios disponibles en internet utilizando conexiones

seguras y servicios de pago electrónico. Aunque el colapso de “dot-com” en 2000 dirigió a desafortunados resultados y muchas compañías e-Commerce desaparecieron, el “brick and mortar” retailers reconocieron las ventajas del comercio electrónico y comenzaron a la integración de tales características a sus sitios web. Para finales de 2001, el modelo de negocio más grande de e-Commerce, Business-to-Business (B2B), había ganado al rededor de \$700 billones en transacciones.

————— AQUÍ UNA FOTO —————

Acorde a toda información disponible, las ventas e-Commerce continuaron creciendo en los siguientes años y, a finales de 2007 ventas e-Commerce representaron el 3.4 % de las ventas totales en el mundo.

E-Commerce tiene muchas ventajas sobre “brick and mortar” tiendas y catálogos de venta por correo. Los consumidores pueden fácilmente buscar a través de una base de datos muy extensa de productos y servicios. Pueden ver los precios reales, definir una orden de compra en varios días y enviar un correo como un “wish list” esperando que alguien pague por sus productos seleccionados. Clientes pueden comparar precios con un simple click del mouse y comprar los productos seleccionados al mejor precio.

Proveedores online, en sus turnos, también tienen claras ventajas. La web y sus motores de búsqueda proveen una manera para encontrar clientes sin campañas de publicidad costosas. Incluso tiendas online pequeñas pueden alcanzar mercados globales. Tecnología web también permite realizar un seguimiento sobre las preferencias de los clientes y para ofrecer la comercialización individual personalizada.

La historia de e-Commerce es impensable sin Amazon e Ebay los cuales estuvieron entre las primeras compañías que permitían transacciones electrónicas. Gracias a sus fundadores ahora tenemos un sector considerable de e-Commerce y disfrutar de comprar y vender gracias a la Internet. Actualmente hay 5 de los más grandes y más famosos worldwide internet retailers : Amazon, Dell, Staples, Office Depot y Hewlett Packard. De acuerdo a las estadísticas, las categorías de productos más vendidos en World Wide Web son música, libros, computadores, artículos de oficina y otros dispositivos electrónicos.

Amazon.com, Inc. es uno de las más famosas compañías e-Commerce para vender productos sobre la internet. Después del colapso “dot-com” Amazon perdió su posición de modelo de negocio exitoso, sin embargo, en 2003 la compañía hizo su primer año con utilidades el cual fue el primer paso para el desarrollo futuro.

Al principio Amazon.com fue considerado como una tienda *online* de libros, pero con

el tiempo una variedad de productos electrónicos fueron agregados, software, DVDs, juego de video, CDs de música, MP3s, prendas de vestir, calzado, productos de salud, etc. El nombre original de la compañía fue Cadabra.com, pero rápidamente después que se volviera popular Internet Bezos decidió cambiar el nombre de su negocio a “Amazon” después del río voluminoso mas grande del mundo. En 1999 Jeff Bezos fue nombrado como la persona del año por Time Magazine en reconocimiento al éxito de la compañía. Aunque la sede principal de la empresa se encuentra en USA, WA, Amazon ha establecido sitios web separados en otros países tales como United Kingdom, Canada, France, Germany, Japan, y China. La compañía apoya y opera *retail web sites* para muchos negocios famosos, incluyendo Marks & Spencer, Lacoste, la NBA, Bebe Stores, Target, etc.

Amazon es uno de los primeros negocios *e-Commerce* en establecer un programa de marketing para los afiliados, y actualmente la compañía obtiene cerca del 40 % de sus ventas desde afiliados y vendedores de terceras partes que lista y vende productos en el sitio web. En 2008 Amazon penetro en el cine y actualmente esta patrocinando la pelicula “*The Stolen Child*” con *20th Century Fox*.

Acorde a las investigaciones en 2008, el dominio Amazon.com a traído cerca de 615 millones de clientes cada año. La característica mas popular del sitio *web* es el *review sistem*, i.e., la habilidad de los visitantes de presentar sus *reviews* y *rate* cualquier producto en un *raiting scale* de uno a 5 estrellas. Amazon.com es también *well-know* por su *clear and user-friendly* avanzado sistema de búsqueda que permite a los visitantes buscar por *keywords* en el texto completo de muchos libros en la *database*.

Otra compañía ha contribuido mucho en el proceso de desarrollo de *e-Commerce* es Dell Inc., una compañía americana posicionada en Texas, que se situa en el tercer lugar de ventas de computadoras después de Hewlett-Packard y Acer.

Lanzado en 1994 como una pagina estatica, Dell.com ha realizado rápidos pasos, y para finales de 1997 fue la primera compañía en lograr el *record* de un millón de ventas *online*. Su única estrategia de ventas sobre la *World Wide Web* sin *retail outlets* y sin intermediarios han sido admirados por un sin fin de clientes e imitados por un gran numero de *e-Commerce businesses*. El factor de éxito de Dell es que Dell.com permite a los clientes elegir y controlar, i.e., visitantes pueden explorar el sitio y ensamblar PC piesa por piesa eligiendo el mas mínimo componente basado en sus presupuestos y requerimientos. De acuerdo a las estadísticas, aproximadamente la mitad de las ganancias de las compañías proviene desde su *web site*.

En 2007, Fortune magazine *ranked* Dell como la *34th-largest* compañía en *Fortune 500 list*, y *8th* en su *Top 20 list* anual de las mas exitosas y admiradas compañías en USA en

reconocimiento al *bussiness model* de la compañía.

La historia de *e-Commerce* es nueva, un mundo virtual que esta evolucionando de acuerdo a las ventajas del cliente. Es un mundo que todos construyen en conjunto ladrillo por ladrillo, estableciendo una base segura para las nuevas generaciones.

1.0.1. E-Commerce en la actualidad

En la actualidad, *e-Commerce* es una experiencia destacable. Transformo las compras tradicionales mas alla de los reconocible. La experiencia es mucho mejor que cualquier otra manera de comprar que actualmente a seducido a una gran cantidad de *e-Commerce lovers*.

Si hace algunos años atras *e-Commerce* fue una palabra de moda, ahora se ha convertido en la orden del día. Al parecer las personas comprar literalmente en cualquier parte - en sus lugares de trabajo durante el almuerzo, en las horas punta cuando no hay nada mas por hacer salvo encender el computador y comerzar a navegar.

En el presente, *e-Commerce* ha ganado tanta popularidad debido a que su tecnologia subyacente esta evolucionando a pasos agigantados. Incluso se esta ofreciendo "*feel*" el producto con un *mouse 3D* para comprender mejor su forma, tamaño y textura. ¿Para que salir cuando todo lo que debes hacer es hacer un pedido, elegir la forma de envio, pararte y esperar hasta que la orden sea entregada hasta tu puerta?

E-Commerce hoy ofrece tanto lujo que incluso las tiendas convecionales han encendido las alarmas. Aungue, cada uno esta de acuerdo que hay un gran camino para que *e-Commerce* reemplace las tiendas, la posibilidad existe que ocurra en el futuro. *E-Commerce* del cual somos actualmente testigos trae tanta aventura a nuestras vidas que es disfrutado por toda la comunidad *online*.

En la actualidad, *e-Commerce* tiene impuestos

1.0.2. E-Commerce en el futuro

Expertos predicen un prometedor y glorioso futuro para *e-Commerce*. En un presumible futuro, **e-Commerce**

1.1. Motivación

Acá va la motivación...

- **Shopping-Cart:**

- **Reservas horas medicas:** Un sistema para gestionar las horas medicas que muestre sugerencias de acuerdo a la distancia que se encuentran los especialistas desde la geo-posicion del sistema que realiza la solicitud.
- **Consultas de libros en una biblioteca:** Un sistema para consultar la disponibilidad, la cantidad, la posibilidad de reservarlo, e incluso cuando sera devuelto.
- **Catalogo de Moteles:** Un catalogo de moteles de acuerdo a la geo-posicion pueden ser listados para determinar las mejores opciones disponibles. Adicionalmente se podría mostrar los horarios disponibles de las habitaciones, e incluso poder generar reservas y pagar la habitación. De esta manera se optimiza el tiempo de los clientes, y los recursos de los proveedores.
- **Catalogo de Medicamentos:** Un catalogo completo de los medicamentos oficialmente aceptados podrían ser listados para entregar información relevante de acuerdo a las dosis así como de efectos adversos, si es necesaria una receta, etc. Otra cosa interesante, es dar la opción de mostrar todos los remedios genéricos que existen en el mercado. El escenario ideal seria ademas mostrar el precio de estos, así como la distancia a las droguerías en donde el producto se encuentra disponible.
- **Reserva de horas en Registro Civil:** Al menos en Chile, realizar un tramite en el registro civil es sinónimo de una perdida de tiempo absurda solo para ser atendido. Si la reserva de hora puede ser realizada desde cualquier sitio, el sistema se volvería drásticamente mas eficiente.
- **Catalogo de Fiestas:** Un catalogo con las fiestas disponibles tanto al a brevedad posible, como en el futuro. Que muestre información relevante

Todas estas situaciones tienen algo en común, y es que los usuarios desean consultar información para tomar una decisión. El sistema idealmente permite contrastar información entre diferentes proveedores de servicios similares, así como de dejar información clara y relevante, para que futuros usuarios puedan tomar una decisión aun mas informados.

Como se puede concluir, el modelo de negocio de estas situaciones es en su mayoría el mismo, solo cambiado la presentación para hacerlo *ad-hoc* a la situación.

1.2. Alcances y Objetivo General

Desarrollar un código base que permita desarrollar una familia aplicaciones. Dicha aplicación se utilizara para desarrollar 3 ejemplos a fin de demostrar su utilidad.

1.3. Objetivos Específicos

- Determinar la tecnología mas adecuada para el desarrollo del proyecto
- Determinar la arquitectura adecuada para el desarrollo de software genérico
- Desarrollar un e-Commerce
- Objetivo 4

1.4. Características deseables

- **Add:** Arrange order of product's variants in your shop with drag and drop.
- **Drag & Drop Variant Management:** Arrange order of product's variants in your shop with drag and drop.
- **Drag & Drop Merchandising:** Arrange order of products in your shop with drag and drop.
- **Device Agnostic:** Optimized experience for all mobile, tablet, and desktop devices.
- **Inline Field Editing:** Add and edit your shop's content by clicking any text field.
- **Google Analytics:** Out of the box Google Analytics event tracking.
- **PayPal Integration:** Supports the ability to use PayPal checkout.
- **Real-time Reactive:** Changes made to your shop are instantly seen by your shoppers (without page reloads).
- **Clone Existing Products:** Create new products by cloning any existing products.
- **Multiple Images Per Product Variant:** Add multiple product images per product option.

- **Recursive Tag Taxonomy:** Uses recursive tag taxonomy for categorization.
- **Clone Product Variants:** Create new product variants by cloning any existing product variant.
- **Fully Open Source:** Node.js and Meteor. Completely Open source. All day. Every day.
- **Quantity per Variant:** Inventory support by product variant.
- **Published or Hidden Status:** Ability to have products in a "draft" state before publishing.
- **Product Details:** Supports additional product details in key/value list.
- **SEO Hashtags:** Uses social media hashtags for product urls for simple SEO+social media tracking.
- **One Page Checkout:** Checkout all done on one page.
- **Shop Analytics:** Integrated tracking framework for integration to any analytics system
- **Docker.io Ready:** Build, Ship and Run anywhere.
- **Backorders:** Shoppers can purchase products even when quantity runs out.
- **i18n and l10n:** Internationalization and localization to translate and localize all content for the world.
- **Variant Management:** Easily manage product options (e.g. size/color) and related product photos.
- **App Gallery:** Select from a gallery of apps to extend your shop by adding your favorite tools.
- **Social Media Integration:** Integrated custom product social media messaging (FB, Twitter, Pinterest, Instagram).
- **Custom Domains:** Use your own domain names.
- **Flat Rate Tax Management:** Manage tax rules for your store.
- **Additional Payment Methods:** Support for more payment methods (providers TBD).
- **Global Shipping:** Ship your products around the globe (carriers TBD).
- **User Management:** Invite users and grant permissions.
- **Email Templates:** Manage email templates for your shop.

- **Hero Manager:** Inline management for hero sections on your shop.
- **Search:** Auto-suggest, keyword product search.
- **Braintree Integration:** Braintree Integration.
- **Stripe Integration:** Use Stripe for payments.
- **Native Mobile Apps:** Build and deploy iOS, Android native React apps.
- **Multi-Currency Support:** Support for additional currencies beyond US Dollars.
- **RTL Localization:** Right to left language support.
- **Revision Control:** Revision control with rollback for all edits.
- **History Logging:** Full insight to all actions performed.
- **Cash on Delivery Payments:** Cash on delivery payment methods.
- **API:** Support for both HTTP API and Meteor DDP.
- **Product Inheritance:** Manage product pricing, promotions, visibility through parent-child-clone inheritance.
- **Coupon Codes:** Coupon management and tracking for discounts.
- **Returns and Refunds:** Track, manage, and analytics on returns and exchanges.
- **Multi-Vendor:** Multiple vendors with review, publish, drop shipping.
- **Flexible Tax Management:** Manage and customize tax rules.
- **Subscription Products:** Subscription based product types.
- **Order Entry and Editing:** Administrator addition and editing of orders.
- **Embed Social Content:** Embed reviews, tweets, and other social content.
- **Bitcoin Integration:** Ability to accept Bitcoin payments in your shop.
- **Amazon Payments Integration:** Use Amazon for payments.
- **Promotions:** Ability to manage and track promotions by channels, events, and more.
- **Google Wallet Integration:** Use Google Wallet for payments.
- **Shipwire Integration:** Ability to use Shipwire for order fulfillment.

- **ShipStation Integration:** Ability to use ShipStation for order fulfillment.
- **MailChimp Integration:** Use MailChimp to collect and manage emails.
- **Actionable Analytics:** Data driven product presentation, and performance analysis.
- **Hotkeys:** Establish shortcuts for regular tasks to quickly trigger a Reaction action.
- **Theme Gallery:** Select from a gallery of themes to change the design and experience of your shop.
- **Import from Squarespace:** Import your product catalog from Squarespace into Reaction.
- **Import from Shopify:** Import your product catalog from Shopify into Reaction.
- **Import from Magento:** Import your product catalog from Magento into Reaction.
- **Import from Spree Commerce:** Import your product catalog from Spree Commerce into Reaction.
- **Import from Big Commerce:** Import your product catalog from Big Commerce into Reaction.

1.5. Estructura de la memoria

La estructura utilizada en este documento para exponer el trabajo realizado es la siguiente:

- **Capítulo 1. Introducción:** Corresponde a la descripción del tema, la motivación de éste y los alcances y objetivos del trabajo realizado.
- **Capítulo ??.** **??:** Corresponde a la revisión bibliográfica o antecedentes. En este capítulo se explican los conceptos necesarios para la comprensión y contextualización del trabajo.
- **Capítulo 3. Conclusiones:** Se enumeran las conclusiones del trabajo realizado y se proponen trabajos a realizar en el futuro.

Capítulo 2

Estado del Arte

Open-source e-Commerce shopping carts ofrecen muchas ventajas para *small businesses*. Soluciones *Open-source* pueden ser desarrolladas para ajustarse a las necesidades del negociante. Ellos contienen una gran combinación de características a un mínimo costo. Y, aunque las opciones de soporte pueden ser mas limitadas que las propietarias o *hosted platforms*, soluciones independientes *open-source* generalmente tienen grandes comunidades de desarrolladores y socios para ayudar a los nuevos comerciantes.

Aquí hay una lista de 11 *open-source e-Commerce solutions*. El *core* de todas las aplicaciones es *free*. Cada aplicación tiene extensiones *free* y *premium* y opciones de soporte para mejorar el desarrollo de la *store*.

2.0.1. OpenCart

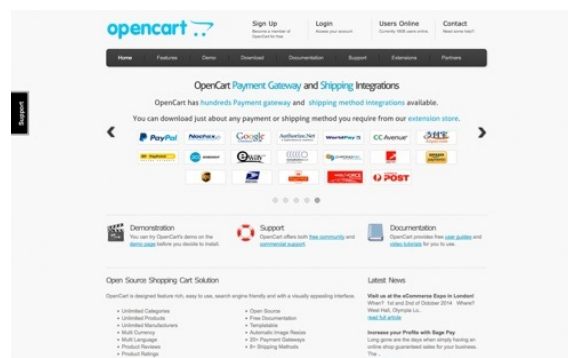


Figura 2.1: OpenCart *website* [12].

OpenCart es una aplicación *open-source*, basada en PHP, es una solución *e-Commerce* para comerciantes *online*. OpenCart tiene una comunidad activa y leal para el apoyo de usuarios, así como una lista de socios comerciales para instalación y customización profesional. En OpenCart existen más de 20 medios de pago, y más de 8 métodos de envío para la descarga por defecto, con cientos de métodos adicionales para el envío y el pago en su directorio de extensiones. OpenCart también fue diseñado para un manejo sencillo de múltiples compras desde una interfaz de administración. Tiene más de 2700 *themes*.

2.0.2. PrestaShop

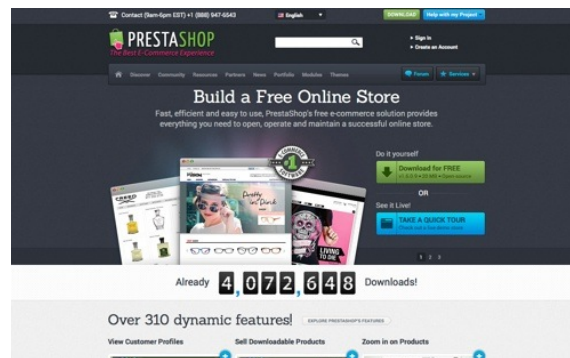


Figura 2.2: PrestaShop website [14].

PrestaShop es una solución *e-Commerce open-source*, escrita en PHP y basada en *Smarty template engine*. PrestaShop viene con más de 310 características integradas y 3500 módulos y *templates*. Cuenta con ventas cruzadas, productos descargables, la exportación de productos, una página de pago, envío, descuentos y mucho más. Descargado más de 4 millones de veces, PrestaShop es usado en 160 países y traducido a 63 idiomas. Tiene más de 600000 miembros en su comunidad.

2.0.3. Magento Community Edition

Magento Community Edition es una versión *free* y *open-source* de una plataforma *e-Commerce*. Los comerciantes pueden acceder a características adicionales instalando las extensiones desde el gran *Magento Connect marketplace*. No existe soporte para Magento Community Edition, así que las respuestas a las preguntas técnicas deben ser resultas a través del foro de usuarios. Un detalle, Magento ha anunciado el cierre de su *hosted solution*, Magento Go, por ahora no habrían problemas con Community Edition. Magento Community Edition soporta más de 200000 sitios de clientes.

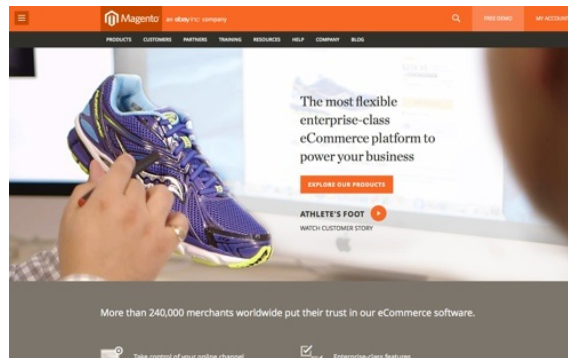


Figura 2.3: Magento website [9].

2.0.4. Zen Cart



Figura 2.4: Zen Cart website [21].

Zen Cart es una aplicación *e-Commerce open-source* escrita en PHP. Zen Cart *branched* desde el código osCommerce en 2003, con una solución que era mas *template-based*. Tiene mas de 1800 *add-ons* en 16 categorías. La comunidad de apoyo de Zen Cart tiene aproximadamente 150000 miembros y 200000 *threads*.

2.0.5. Spree Commerce

Spree Commerce es una solución *e-Commerce open-source* basado en Ruby on Rails. La plataforma modular permite configurar, complementar, o reemplazar cualquier funcionalidad que necesites. Spree Commerce tiene mas de 45000 tiendas la plataforma al rededor del mundo, incluyendo Chipotle [2] has more than 45,000 stores using the platform around the world, including Chipotle. Spree Commerce has been translated into more than 30 languages.



Figura 2.5: Spree Commerce *website* [16].

2.0.6. Drupal Commerce

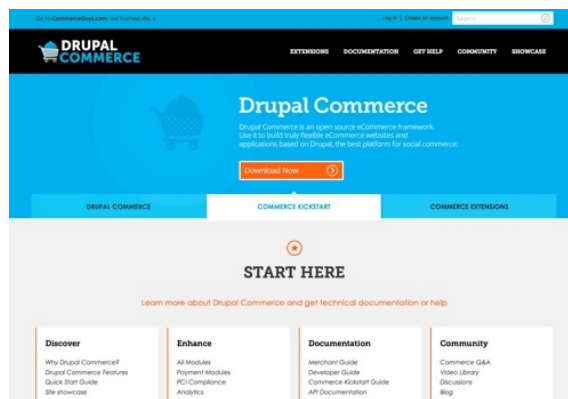


Figura 2.6: Drupal Commerce *website* [5].

Drupal Commerce es una aplicación *e-Commerce* desarrollado por Commerce Guys. Construido sobre Drupal content management system. Drupal Commerce ofrece un sistema de administración de producto completo, carro de compra varios lenguajes y monedas; y forma de pago. La lista de extension de Drupal Commerce es una integración completamente en *third-party* para formas de pago, servicios de cumplimiento, aplicaciones de contabilidad, redes sociales y mucho mas. Paquetes de soporte técnico están disponibles por Commerce Guys.

2.0.7. osCommerce

osCommerce (i.e., *open source Commerce*) es uno de las primeras aplicaciones *e-Commerce open-source*. Mas de 7000 *free add-ons* han sido subidos por su comunidad para

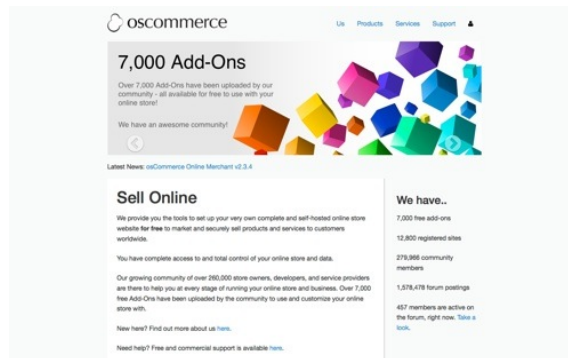


Figura 2.7: osCommerce website [13].

customizar una *online store*. osCommerce es usado por cerca de 13000 sitios registrados. La comunidad de apoyo tiene aproximadamente 280000 miembros los cuales han contribuido con 1.5 millones de posts en los foros. La comunidad directa juntos con miembros de otra comunidad están disponibles en vivo en el *Chat room*.

2.0.8. simpleCart

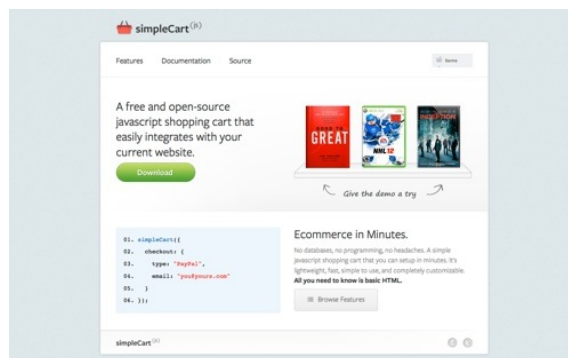


Figura 2.8: simpleCart website [15].

simpleCart (js) es un *free y open-source JavaScript shopping cart*. Con su pequeño tamaño, simpleCart (js) esta diseñado para mantener simple y sitios con alto trafico *running fast*. simpleCart (js) tienen la habilidad de pagar con PayPal Express, Google Checkout, y Amazon Payments. *Email checkout* e integracion con Authorize.Net llegan pronto.

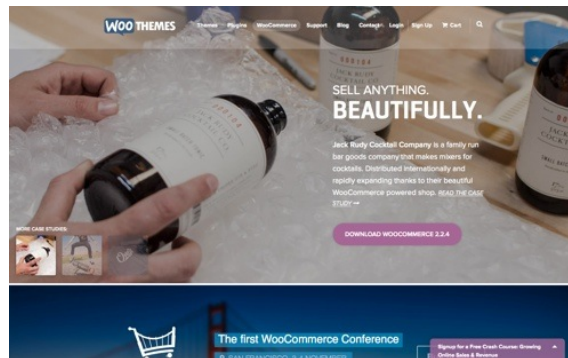


Figura 2.9: WooCommerce website [18].

2.0.9. WooCommerce

WooCommerce es una aplicacione-Commerce free open-source que permite a los comerciantes transformar WordPress sites en stores. WooCommerce fue desarrollado por WooThemes desde un fork de Jigoshop. WooCommerce tiene una larga variedad de plugins y themes de WooThemes, como de sitios third party tales como ThemeForest [17] y CodeCanyon [3]. Con cerca de 4.5 millones de descargas desde WordPress.org [19], WooCommerce es una solucion e-Commerce muy popular para WordPress. Para obtener el soporte oficial de WooThemes, es necesario comprar un producto. De otra manera, obtener ayuda desde la comunidad activa del foro.

2.0.10. WP e-Commerce



Figura 2.10: WP e-Commerce website [20].

WP e-Commerce es otra aplicacion popular obtenida desde la conversion del sitio WordPress a una e-Commerce store. WP e-Commerce tiene cerca de 3 millones de descarga

de *plugin* desde *WordPress.org*. Usa el propio HTML y CSS y obtén el control completo sobre la vista y la experiencia de tu *online store*. WP e-Commerce tiene una gran variedad de características estándar, incluyendo *multi-tier pricing* para descuentos por cantidad e integración con redes sociales para *marketing*. Para soporte, hay tutoriales en video y un foro en *WordPress.org*, también como consultantes de características para ayuda profesional.

2.0.11. Jigoshop



Figura 2.11: Jigoshop website [8].

Jigoshop es una solución *e-Commerce free* y *open-source* basado en WordPress. Liberado en 2011, Jigoshop es el predecesor para WooCommerce. Jigoshop tiene más de 30 *themes*, 100 extensiones, y tres *theme frameworks*. Jigoshop es *free*, así como el soporte a *WordPress.org*. Sin embargo, el acceso a la comunidad de *Jigoshop.com* comienza desde \$40 por mes.

Como se observa, existe una gran cantidad de aplicaciones *open-source* las cuales están disponibles para ingresar al mundo del comercio electrónico. Todas estas opciones tienen grandes comunidades que las respaldan, así como muchos usuarios satisfechos.

Considerando esto, ¿habrá alguna razón que justifique el desarrollo de un proyecto de estas características?

2.1. Justificación para el desarrollo del proyecto

2.1.1. Base de datos

base de datos esclavos horizontalmente, SQL escalables verticalmente.

What about transactions?

Many people bring up MongoDB's lack of atomic transactions across collections as evidence that it's not suitable for e-commerce applications. This has not been a significant barrier in our experience so far.

There are other ways to approach data integrity. In systems with low-moderate data contention, optimistic locking is sufficient. We'll share more details about these strategies as things progress.

	traider [26]	ReactionCommerce [22]	NodeShop [27]	Forward [6]
Tecnología	bootstrap, nodejs and mongodb	Nodejs, Meteor, Mongodb, CoffeScript, Bootstrap, Docker		
Mobile				
version			0.06 (21/08/2013)	0.1

Tabla 2.1: Comparación entre diferentes opciones *e-Commerce*

2.2. Tecnologías

2.2.1. Base de datos

Relational databases se encuentran en la mayoría de las organización desde hace muchísimo tiempo, y por buenas razones. *Relational databases* apoya las aplicaciones del pasado que cumplen con las necesidades de negocio actuales; Estas son apoyadas por un extenso ecosistema herramientas; y hay una gran cantidad de mano de obra calificada para implementar y mantener estos sistemas.

Pero las compañías están cada vez mas considerando la opción de alternativas para la infraestructura relacional heredada. En algunos casos la motivación es técnica, tal como la necesidad de escalar o actuar mas allá de las capacidades de sus sistemas actuales. Mientras que en otros casos, las compañías están motivadas por el deseo de identificar alternativas viables a *softwares* propietario de alto costo. una tercera motivación es la agilidad y velocidad del desarrollo, dado que las compañías ansían adaptarse al mercado mas rápido y adoptar metodologías de desarrollo ágil.

Estas opciones se aplican tanto para aplicaciones analíticas como transaccionales. Las compañías están cambiando *workloads to Hadoop* para sus *offline, analytical workloads*, y están construyendo *online*, aplicaciones operaciones con una nueva clase de tecnología de manejo de datos llamada "[nosql??]", como por ejemplo *MongoDB*

2.2.2. [sql??] and the Relational Model

[sql??] es un lenguaje declarativo de consulta de datos . Un lenguaje declarativo es uno en el cual un programador especifica lo que desea y el sistema lo ejecuta, en lugar de definir proceduralmente como el sistema debería hacerlo. Unos pocos ejemplos incluyen: encontrar el registro del empleado 39, mostrar solo el nombre y el número de teléfono del empleado de la totalidad de su registro, filtrar los registros de los empleados a aquellos que trabajan en contabilidad, contar la cantidad de empleados en cada departamento, unir la información de la tabla de los empleados *employees* con la tabla de *managers*.

En una primera aproximación, [sql??] permite consultar sobre aquellas preguntas sin pensar sobre como la información es expuesta en el disco, cuales índices utilizar para acceder a la información o que algoritmo utilizar para procesar la información. Un componente arquitectural significativo para la mayoría de los *relational databases* is a *query optimizer*, el cual decide cual de las muchas equivalentes lógicas planea ejecutar las respuesta mas rápida a una *query*. Estos optimizadores son usualmente mejores que los promedios de los usuarios de la *database*, pero en algunas ocasiones ellos no tienen la suficiente información o tienen un modelo muy simple del sistema en *order* para generar ejecuciones mas eficientes.

Relational databases, las *databases* mas utilizadas en la practica, siguen el modelo de datos relacional. En este modelo, diferentes entidades del *real-world* son guardadas en diferentes tablas. Por ejemplo, todos los *employees* podrian ser guardados en una tabla *Employees*, y todos los *departments* podrían ser almacenados en la tabla *Departments*. Cada fila de una tabla tiene varias propiedades guardadas en columnas. Por ejemplo, *employees* podrian tener un *employee id*, *salary*, *birth date*, y *first/last names*. Cada una de estas propiedades sera uardada en una columna de la tabla *Employees*

El modelo relacional, va mano a mano con [sql??]. *Queries* simples de [sql??], tales como filtrar, recuperar todos los registros el cual sus campos hagan *match* en algunos test(ejemplo, *employeeid* = 3, o *salary* \$20000). *Constructres mas complejos causan que database haga trabajo extra, tal como joining informacin sob*

Relational data model define entidades altamente estructuradas con relaciones estrictas entre ellos. *Querying* este modelo con [sql??] permite *complex data transversa* sin desarrollar mucho. La complejidad del modelo y *querying*, tienen sus límites, aunque:

Complexity guía a *unpredictability*. La expresividad en [sql??] implica un desafío en relación al costo de cada *query*, y así el costo de *workload*. Mientras, lenguajes de *query* simples pueden complicar la lógica, al mismo tiempo hacen que sea sencillo proveer de

almacenamiento de datos., cuando solo responde a *requests* simples.

Hay muchas maneras modelar un problema. *Relational data model* es estricto: El *schema* asignado para cada tabla especificada la *data* en cada fila. Si se esta almacenando menos *data* estructurada, o filas con mas diferencia en las columnas que se guardan, *relational model* puede ser innecesariamente restrictiva. Similarmente, aplicaciones desarrolladas podrian no encontrar el *relational model* idoneo para modelar cada tipo de *data*. Por ejemplo, una gran cantidad den aplicaciones logicas son escritas en lenguaje *object-oriented* e incluye conceptos *high-level* tales como *lists*, *queues*, y *sets*, y algunos programadores desearan *persistence layer* para modelar esto.

NoSQL

NoSQL encompasses a wide variety of different database technologies and were developed in response to a rise in the volume of data stored about users, objects and products, the frequency in which this data is accessed, and performance and processing needs. Relational databases, on the other hand, were not designed to cope with the scale and agility challenges that face modern applications, nor were they built to take advantage of the cheap storage and processing power available today.

Document Model

Mientras *relational databases store data* en filas y columnas, *document databases store data* en documentos.

Estos documentos tipicamente usas una estructura equivalente a **[json??]**, un formato muy popular entre desarrolladores. Documentos proveen una manera intuitiva y natural para modelar *data* que esta cercanamente alineada con programación *object-oriented*, en el cual cada documento es efectivamente un objeto. Los documentos contienen uno o mas campos, donde cada campo contiene un tipo de valor, tales como *string*, *date*, *binary*, o *array*. En lugar de extenderse un registro entre múltiples tablas y columnas, cada registro y su *data* asociada son tipicamente almacenadas juntas en un solo documento. Esto simplifica el acceso a la *data* y reduce e incluso elimina la necesidad de *joins* y *complex transactions*.

En un *document database*, la nocion de esquema es dinamica: cada documento puede contener diferentes campos. Esta flaxibilidad puede ser particularmente util para modelar *data* sin estructura y *polymorphic*. Esto tambien hace posible la evoluición de una aplicación

durante su desarrollo, simplemente agregando nuevos campos. Adicionalmente, *document databases* generalmente proveen consultas robustas que los desarrolladores esperan en *relational databases*.

Aplicaciones: *Document databases* son de propósito general y útiles para una amplia variedad de aplicaciones, debido a su flexibilidad del *data model*, la habilidad para consultar sobre cualquier campo y el *natural mapping* del *document data model* a objetos en programación de lenguajes modernos.

Ejemplos: MongoDB y CouchDB.

Graph Model

Basado en Teoría de Grafos, usa estructuras de grafos con nodos, bordes y propiedades para representar la *data*. -en esencia, la *data* es modelada como una red de relaciones entre elementos específicos. Si bien es cierto, *Model Graph*, puede ser contra intuitivo y tomar tiempo para entenderlo, puede ser utilizado extensamente para numerosas aplicaciones. Su principal característica es que modela fácilmente las relaciones entre entidades en una aplicación.

Aplicaciones: *Graph databases* son útiles en escenarios donde las relaciones son el *core* de la aplicación, como redes sociales. Ejemplos: Neo4j y HyperGraphDB.

Key-Value

From a data model perspective, key-value stores are the most basic type of NoSQL database. Every item in the database is stored as an attribute name, or key, together with its value. The value, however, is entirely opaque to the system; data can only be queried by the key. This model can be useful for representing polymorphic and unstructured data, as the database does not enforce a set schema across key-value pairs.

Wide Column Models

Wide column stores, or column family stores, use a sparse, distributed multi-dimensional sorted map to store data. Each record can vary in the number of columns that are stored, and columns can be nested inside other columns called super columns. Columns can be grouped

together for access in column families, or columns can be spread across multiple column families. Data is retrieved by primary key per column family.

Applications: Key value stores and wide column stores are useful for a narrow set of applications that only query data by a single key value. The appeal of these systems is their performance and scalability, which can be highly optimized due to the simplicity of the data access patterns.

2Examples: Riak and Redis (Key-Value); HBase and Cassandra (Wide Column).

	SQL Databases	NoSQL Databases
Types	One type (SQL database) with minor variations	Many different types including key-value stores, document databases, wide-column stores, and graph databases
Development History	Developed in 1970s to deal with first wave of data storage applications	Developed in 2000s to deal with limitations of SQL databases, particularly concerning scale, replication and unstructured data storage
Examples	MySQL, Postgres, Oracle Database	MongoDB, Cassandra, HBase, Neo4j
Data Storage Model	Individual records (e.g., "employees") are stored as rows in tables, with each column storing a specific piece of data about that record (e.g., "manager,date hired,etc.), much like a spreadsheet. Separate data types are stored in separate tables, and then joined together when more complex queries are executed. For example, "offices" might be stored in one table, and "employees" in another. When a user wants to find the work address of an employee, the database engine joins the "employee" and "office" tables together to get all the information necessary.	Varies based on database type. For example, key-value stores function similarly to SQL databases, but have only two columns ("key" and "value"), with more complex information sometimes stored within the "value" columns. Document databases do away with the table-and-row model altogether, storing all relevant data together in single "document" in JSON, XML, or another format, which can nest values hierarchically.
Schemas	Structure and data types are fixed in advance. To store information about a new data item, the entire database must be altered, during which time the database must be taken offline.	Typically dynamic. Records can add new information on the fly, and unlike SQL table rows, dissimilar data can be stored together as necessary. For some databases (e.g., wide-column stores), it is somewhat more challenging to add new fields dynamically.
Scaling	Vertically, meaning a single server must be made increasingly powerful in order to deal with increased demand. It is possible to spread SQL databases over many servers, but significant additional engineering is generally required.	Horizontally, meaning that to add capacity, a database administrator can simply add more commodity servers or cloud instances. The database automatically spreads data across servers as necessary.
Development Model	Mix of open-source (e.g., Postgres, MySQL) and closed source (e.g., Oracle Database)	Open-source
Supports Transactions	Yes, updates can be configured to complete entirely or not at all	In certain circumstances and at certain levels (e.g., document level vs. database level)
Data Manipulation	Specific language using Select, Insert, and Update statements, e.g. SELECT fields FROM table WHERE...	Through object-oriented APIs
Consistency	Can be configured for strong consistency	Depends on product. Some provide strong consistency (e.g., MongoDB) whereas others offer eventual consistency (e.g., Cassandra)

Tabla 2.2: Resumen NoSQL vs. SQL

2.2.3. MongoDB

MongoDB (from "humongous") is an open-source document database, and the leading NoSQL database. Written in C++ [10].

2.2.4. Angularjs

[24]

2.2.5. Bootstrap

Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web [1].

2.2.6. Nodejs

Node.js® is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices [11].

2.2.7. CoffeeScript

CoffeeScript is a little language that compiles into JavaScript. Underneath that awkward Java-esque patina, JavaScript has always had a gorgeous heart. CoffeeScript is an attempt to expose the good parts of JavaScript in a simple way.

The golden rule of CoffeeScript is: "It's just JavaScript". The code compiles one-to-one into the equivalent JS, and there is no interpretation at runtime. You can use any existing JavaScript library seamlessly from CoffeeScript (and vice-versa). The compiled output is readable and pretty-printed, will work in every JavaScript runtime, and tends to run as fast or faster than the equivalent handwritten JavaScript [4].

2.2.8. Grunt

[7]

2.2.9. Docker

Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications. Consisting of Docker Engine, a portable, lightweight runtime and packaging tool, and Docker Hub, a cloud service for sharing applications and automating workflows, Docker enables apps to be quickly assembled from components and eliminates

the friction between development, QA, and production environments. As a result, IT can ship faster and run the same app, unchanged, on laptops, data center VMs, and any cloud [23].

<i>E-Commerce</i>	SQL Databases	NoSQL Databases
El <i>schema</i> de los productos varia	<i>Schema</i> rígido	<i>Schema</i> flexible
Necesita escalar		
Transacciones		

Tabla 2.3: [nosql??] vs. [sql??] en relación a [ecommerce??]

2.3. MongoDB y E-Commerce [25]

Demostración de *next-generation data stores* típicamente giran en torno a *social network*: Twitter, Facebook, Foursquare, etc. Desafortunadamente, tales aplicaciones tienden a tener complejos *data models*. *E-Commerce* por ejemplo, tiene la ventaja de incluir un largo número de patrones familiares para *data modeling*. Además no es complicado imaginar como *products*, *categories*, *product reviews* y *orders* son típicamente modeladas en [rdbms??].

E-Commerce han sido usualmente un dominio exclusivo de [rdbms??]s, y esto es así por un par de razones. La primera, es que [ecommerce??] *sites* generalmente requieren *transactions*, y *transactions* son una operación básica en [rdbms??]. Lo segundo es que, hasta hace poco, dominios que requieren de un *rich data model* y sofisticadas *queries* ha sido presupuesto que se ajustan mejor en [rdbms??]. En las siguientes secciones se cuestionara la segunda afirmación.

2.3.1. Catalog Management

Si es necesaria información sobre como el *catalog management* se maneja con *relational databases*, es necesario dar una mirada rápida a los *schemas* del popular Magento *e-Commerce framework* C.1 o OfBiz de Apache C.2. Lo que se observa es un conjunto de tablas trabajando a la par para proveer un *schema* flexible sobre un fundamentalmente inflexible estilo de *database system*.

Esto significa que la *data* de cualquier producto se extiende a través de una docena de tablas. Esto incrementa la complejidad del código requerido para la persistencia de consultas de productos individuales y hacer una consulta *shell-based* es casi imposible. Simplemente considere escribir un [sql??] JOIN para reunir el modelo de un producto como esto:


```

25   ObjectID("4bd87bd8277d094c458d2a43"),
26   ObjectID("4bd87bd8277d094c458d2b44"),
27   ObjectID("4bd87bd8277d094c458d29a1")
28 ]
29 }

```

Listing 2.1: Búsqueda en MongoDB

Claramente no es una representación completa de un producto, pero esto demuestra cuantas de estas tablas triviales que existen en una *relational representation* pueden prescindir en un *document representation*.

Para *object-oriented data*, los documentos tienen mayor sentido, tanto en concepto como rendimiento. Una representación *document-oriented* de una *data* de un producto se traduce a unas pocas entidades (un puñado de *collections* vs. una docena de tablas), mejor *performance* en consultas (sin *sever-side joins*), y estructuras que corresponden precisamente al producto. Ya no existe la necesidad de diseñar un *master schema* que pueda considerar a cada tipo de producto concebible.

Catalog management es esencialmente *content management*, un campo en donde MongoDB sobresale.

2.3.2. Shopping Carts and Orders

Permitir que un *shopping cart* sea simplemente una orden en un estado del *cart*, el modelo de *shopping carts* y las ordenes en MongoDB se tornan muy sencillas:

```

1
2 { '_id': ObjectId('4b980a6dea2c3f4579da141e'),
3   'user_id': ObjectId('4b980a6dea2c3f4579a4f54'),
4   'state': 'cart',
5
6   'line_items': [
7     { 'sku': 'jc-432',
8       'name': 'John Coltrane: A Love Supreme',
9       'retail_price': 1099
10    },
11
12    { 'sku': 'ly-211',
13      'name': 'Larry Young: Unity',
14      'retail_price': 1199
15    },

```

```

16 ],
17
18 'shipping_address': {
19   'street': '3333 Greene Ave.',
20   'city': 'Brooklyn',
21   'state': 'NY',
22   'zip': '11216'
23 },
24
25 'subtotal': 2199
26 }

```

Listing 2.2: Estructura de una orden.

Notar que es posible presentar los pedidos como un *array* de productos. Como es usual con documentos, esto hace el despliegue del *shopping cart* mas sencillo, dado que no hay *joins* envueltos. Pero esto también resuelve el problema del versionamiento de productos. Usualmente es necesario tener el estado de un producto cuando este es comprado. Esto puede ser logrado en una **[rdbms??]** estableciendo un vinculo a versiones particulares de un producto. Aquí, sin embargo, simplemente se almaceno el estado de un producto dentro de la misma orden.

2.3.3. Querying Orders

Dado que MongoDB soporta consultas dinámicas y *secondary indexes*, las consultas para las ordenes son automáticas. Es posible, por ejemplo, definir un *index* en un *product* **[sku??]**, lo que permite consultas eficientes en todas las ordenes para un producto dado:

```

1 db.orders.ensureIndex({'line_items.sku': 1});
2 db.orders.find({'line_items.sku' => 'jc-431'});

```

Listing 2.3: Consulta eficiente con *secondary indexes*.

Con MongoDB, es posible realizar consultas en atributos arbitrarios, de esa manera cualquier *query*, en *orders collection* es posible. Y para *queries* comunes, es posible definir *indexes* para una mejor eficiencia.

2.3.4. Aggregation

Claramente, *aggregation* también es necesario. Se desea reportar ordenes de diferentes maneras, y para ese propósito, *map-reduce* esta disponible. Como ejemplo, el comando *map-*

reduce que *agregtes* el total de ordenes por *zip code*:

```
1 map = "  
2   function() {  
3     emit(this['shipping_address']['zip'], {total: this.total})  
4   }"  
5  
6 reduce = "  
7   function(key, values) {  
8     var sum = 0;  
9     values.forEach(function(doc) {  
10      sum += doc.total;  
11    }  
12  
13    return {total: sum};  
14  }"  
15  
16  
17 db.orders.mapReduce(map, reduce, {out: 'order_totals_by_zip'});
```

Listing 2.4: Ejemplo de commando *map-reduce*.

2.3.5. Updating Orders

Incrementing Quantity

Una manera de ajustar la cantidad es usando un *position operator*, el cual permite aplicar *atomic operations*, a un único objeto dentro de un *array*. A continuación se muestra como cambiar el numero de álbumes que se están ordenando:

```
1 db.orders.update({'_id': order_id, 'line_items.sku': 'jc-431'},  
2   {'$set': {'line_items.$.quantity': 2}});
```

Listing 2.5: Ejemplo de *atomic operation*.

Adding and Removing Items

Igualmente, *atomic operators* resuelven el problema de agregar y remover productos desde el carro. Por ejemplo, se puede utilizar *\$push atomic operator* para agregar un ítem al *cart*;

```
1 db.orders.update({'_id': order_id},
```

```

2      {'$push': {'line_items':
3        {'sku': 'md-12', 'price': 2500, 'title': 'Basketball'}}
4      '$inc': {'subtotal': 2500}});

```

Listing 2.6: Ejemplo de *atomic operation*.

Al ajustar la cantidad y el cambio de los mismos *items* en el *cart*, es necesario actualizar el total de la orden. Notar el uso de *\$inc operator* para manejar esto.

2.3.6. Inventory

No todos los sitios *e-Commerce* necesitan manejar el inventario. Pero para aquellos que si lo hacen, MongoDB funciona a la altura de las circunstancias.

Una manera para manejar el inventario, es *store* un documento separado por cada *physical item* en la bodega. Así, por ejemplo, si la bodega tiene veinte copias del álbum Coltrane, se traduce en veinte documentos distintos en *inventory collection*. Cada documento tiene una estructura como la siguiente:

```

1  {'_id': ObjectId('4b980a6dea2c3f4579da432a'),
2   'sku': 'jc-431',
3   'state': 'available',
4   'expires': null,
5   'order_id': null
6  }

```

Listing 2.7: Ejemplo de *atomic operation*.

Cuando un usuario intenta agregar un *item* al *cart*, un *findAndModify command* puede ser facilitado para *atomically mark* el *item* en *in-cart*, asociando el *item* con una orden dada, y estableciendo un tiempo de expiración:

```

1  query = {'sku': 'jc-431', 'state': 'available'};
2
3  update = {'$set':
4    {'state': 'cart',
5     'order_id': order_id,
6     'expires': Date.now() + 15 * 60}};
7
8  item = db.inventory.findAndModify(query: query, update: update);

```

Listing 2.8: Ejemplo de *atomic operation*.

Si se obtiene un *item back* desde el *findAndModify operation*, se sabe que tenemos un único *lock* en el *item*, y es posible *store it* en el *cart*. Cuando el usuario desea *check out*, el estado del *item* puede cambiar a "*purchased*", o cualquiera sea el caso de la llamada.

Mientras, se pueda ejecutar un *script in the background* que libere el inventario del *cart* que no ha sido *purchased* en la ventana de quince minutos. La actualización es trivial:

```
1 db.inventory.update({'state': 'cart', 'expires': {'$lt': Date.now()}}),
2 {'$set': {'state': 'available', 'expires': null, 'order_id': null}},
3 {multi: true});
```

Listing 2.9: Ejemplo de *atomic operation*.

2.3.7. Transactions, Consistency y Durability

Muchos argumentos impuestos contra NoSQL en *e-Commerce* se centran en *transactions*, *consistency*, y *durability*. En relación a esto se mencionan algunos puntos.

En relación a *transactions*, ciertamente MongoDB no soporta el tipo *multi-object*; sin embargo, soporta *atomic operations* sobre documentos individuales. y esto combinado con *documento-oriented modeling* recién descrito y creatividad, es suficiente para muchos problemas *e-Commerce*. Ciertamente, si se necesita *debit one account* y *credit another* en la misma operación, o si se desea *rollback*, sera necesario *full-fledged transactions*. No obstante, *transactionality* provista por MongoDB debería ser suficiente en la mayoría de los casos, si no en todos, para *e-Commerce operations*.

Si la preocupación esta sobre *consistency* y *durability*, operaciones escritas en MongoDB pueden ser realizadas *consistency* sobre conexiones. Además, MongoDB 1.5 soporta *near-real-time replications*, así que es posible asegurarse que una operación ha sido *replicated* antes de retornar.

2.3.8. Scalability

La manera mas sencilla para *scale* la mayoría de las *databases*, es *upgrading* el *hardware*. Si la aplicación esta corriendo en un único nodo, es usualmente posible agregar una combinación de *disk IOPS*, *memory*, y CPU para eliminar los cuello de botella de la *database*. La técnica de mejorar el *hardware* de un solo node para escalar se conoce como *vertical scaling* o *scaling up*. *Vertical scaling* tiene la ventaja de ser simple, seguro, y *cost-effective* hasta un cierto punto. Si se esta ejecutando sobre un *virtualized hardware*(tal como

Amazon's EC2), entonces puedes encontrar que una instancia lo suficientemente larga no esta disponible. Si estas ejecutando sobre *physical hardware*, habrá un punto donde el costo de un servidor mas poderoso se vuelve prohibitivo.

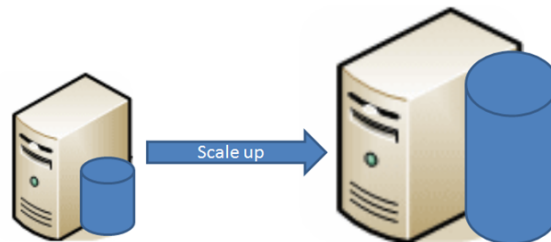


Figura 2.13: *Vertical scaling* o *Scaling up*

Entonces tiene sentido considerar *scaling horizontally* o *scaling out*. En lugar de reforzar un único nodo, *scaling horizontally* significa distribuir de *database* sobre múltiples maquinas. Dado que *horizontally scaled architecture* puede utilizar *comodity hardware*, el costo de *hosting* el total de la *data* puede ser reducido significativamente. Incluso, la distribución de *data* sobre maquinas mitiga las consecuencias de fallo. Las maquinas inevitablemente fallaran de algún momento a otro. Si se *scaled vertically* , y la maquina falla, entonces es necesario tratar con una falla en un maquina de la cual la mayoría del sistema depende. Podría no considerarse un tema si una copia de la *data* existe en un *replicated slave*, pero aun esta el caso e que solo un único *sever* es necesario para bajar el sistema completo. En contraste con el fallo dentro de un *horizontally scaled architecture*. Esto podría ser menos catastrófico dado que una sola máquina representa un porcentaje menor del sistema completo.

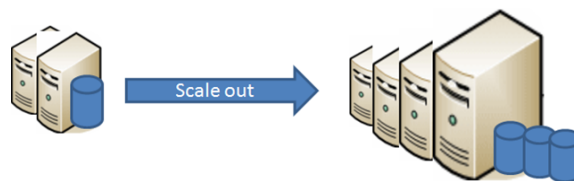


Figura 2.14: *Horizontally scaling* o *Scaling out*

MongoDB es un *database management system* diseñado para hacer *horizontal scaling* manejable, dado que fue construido para aplicaciones *web* e infraestructuras de Internet.

2.3.9. Conclusion

Es cierto que la mayoría de *NoSQL databases* no fueron construidas considerando *e-Commerce*. *Databases* que carecen *data models* enriquecidos, *dynamic queries*, y la noción

de *transactionality* no puede esperarse que compitan en el espacio de *e-Commerce*, entonces no es comprensible que no se considere MongoDB tampoco.

Pero para las partes en donde el sitio **[eCommerce??]** comprende el manejo de contenido, MongoDB es claro vencedor. E incluso para mas *transactional components* del sistema, MongoDB tiene características que hacen de la posibilidad de correr un sistema completo de *e-Commerce* una realidad.

Capítulo 3

Conclusiones

3.1. Trabajos Futuros

Glosario

D

DBMS Database management system.

N

NoSQL Not Only SQL.

R

RDBMS Relational database management system.

S

SKU Stock-keeping unit.

SQL Structured Query Language.

Bibliografía

- [1] *Bootstrap*, Noviembre 2014. <http://getbootstrap.com/>.
- [2] *Chipotle Website*, Noviembre 2014. <http://www.chipotle.com/>.
- [3] *CodeCanyon Website*, Noviembre 2014. <http://codecanyon.net/>.
- [4] *CoffeeScript*, Noviembre 2014. <http://coffeescript.org/>.
- [5] *Drupal Commerce Website*, Noviembre 2014. <https://drupalcommerce.org/>.
- [6] *Forward*, Noviembre 2014. <http://getfwd.com/>.
- [7] *Grunt*, Noviembre 2014. <http://gruntjs.com/>.
- [8] *Jigoshop Website*, Noviembre 2014. <https://www.jigoshop.com/>.
- [9] *Magento Community Edition Website*, Noviembre 2014.
<http://www.magentocommerce.com/download>.
- [10] *MongoDB*, Noviembre 2014. <http://www.mongodb.org/>.
- [11] *Nodejs*, Noviembre 2014. <http://nodejs.org/>.
- [12] *OpenCart Website*, Noviembre 2014. <http://www.opencart.com/>.
- [13] *osCommerce Website*, Noviembre 2014. <http://www.oscommerce.com/>.
- [14] *PrestaShop Website*, Noviembre 2014. <http://www.prestashop.com/>.
- [15] *simpleCart Website*, Noviembre 2014. <http://simplecartjs.org/>.
- [16] *Spree Commerce Website*, Noviembre 2014. <http://spreecommerce.com/>.
- [17] *ThemeForest Website*, Noviembre 2014. <http://themeforest.net/>.

- [18] *WooCommerce Website*, Noviembre 2014.
<http://www.woothemes.com/woocommerce/>.
- [19] *WordPress.org Website*, Noviembre 2014. <https://wordpress.org/>.
- [20] *WP e-Commerce Website*, Noviembre 2014. <http://getshopped.org/>.
- [21] *Zen Cart Website*, Noviembre 2014. <http://www.zen-cart.com/>.
- [22] Dobbertin, Eric: *Reaction Commerce*, 2014. <https://reactioncommerce.com/>.
- [23] Docker, Inc: *Docker*, Noviembre 2014. <https://www.docker.com/>.
- [24] Google: *AngularJs*, Noviembre 2014. <https://angularjs.org/>.
- [25] Kyle: *MongoDB and E-commerce*, Abril 2010. <http://web.archive.org/web/20110713174947/http://kylebanker.com/blog/2010/04/30/mongodb-and-ecommerce/>.
- [26] Software, Eastpoint: *traider*, Noviembre 2014. <http://traider.io/>.
- [27] Topley, Billy: *NodeShop*, Noviembre 2014. <http://nodeshop.org/>.
- [28] Trudeau, Richard J.: *Introduction to Graph Theory*.

Apéndice A

Apendice A

Acá van los apéndices, aprovechando...

Nota: los archivos memoria.algo (con .algo"distinto de "tex") se generan automáticamente al compilar, algunos errores con la bibliografía pueden solucionarse borrándolos y re-compilando.

Apéndice B

Teoria de Grafos

inicio de mi grafo [28]

Apéndice C

Catalog management handled with relational databases

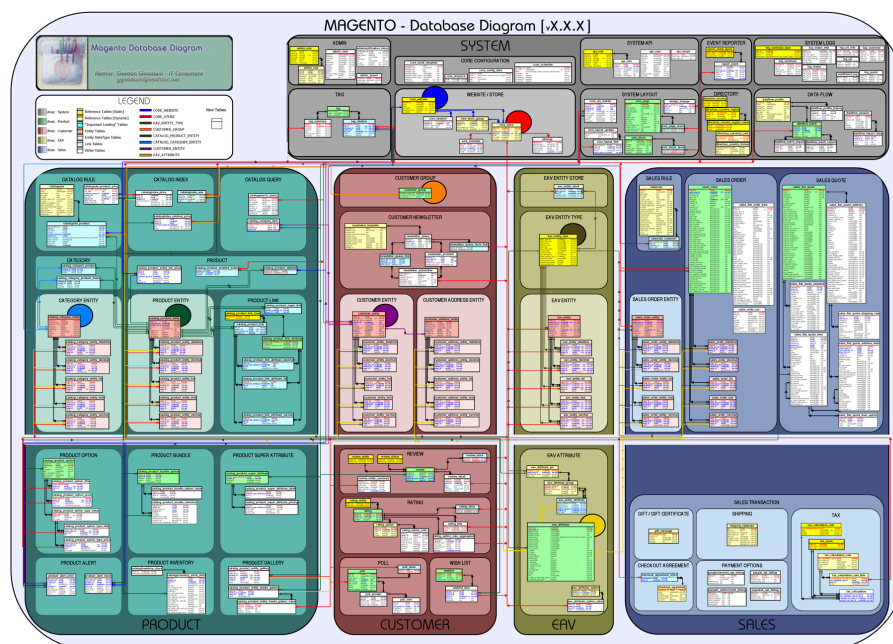



Figura C.1: schemas of Magento e-commerce framework.



`figuras/apendice/openCartWebsite.jpg`

Figura C.2: schemas of the Apache's OfBiz.