

Acknowledgements

I want to thank a few people.

Preface

This is an example of a thesis setup to use the reed thesis document class (for LaTeX) and the R bookdown package, in general.

Índice general

Introducción	1
Capítulo 1: Planteamiento de la necesidad	3
1.1. Planteamiento del problema	3
1.2. Objetivo General	4
1.2.1. Objetivos específicos	4
1.3. Justificación	4
Capítulo 2: Marco Teórico	7
2.1. Virtualización	7
2.2. Cómputo en la nube	7
2.2.1. Ansible	9
2.3. Microservicios	10
2.4. Diseño Basado en Dominios	12
2.4.1. Aspectos principales de DDD	12
2.4.2. Valores y beneficios de DDD	13
Capítulo 3: Modelacion utilizando diseño basado en dominios	15
3.1. Requisitos	15
3.1.1. Subdominio Inventario	15
Conclusion	17
Apéndice A: The First Appendix	19
Apéndice B: The Second Appendix, for Fun	21
References	23

Índice de cuadros

Índice de figuras

2.1. Aplicación monolítica	10
2.2. Microservicios	11
3.1. Modelado del dominio	16

Abstract

The preface pretty much says it all.

Second paragraph of abstract starts here.

Dedication

You can have a dedication here if you wish.

Introducción

Capítulo 1

Planteamiento de la necesidad

1.1. Planteamiento del problema

Tradicionalmente, las empresas contaban con un centro de datos donde alojaban desde un par de servidores a cientos de estos. Un centro de datos es un local o edificio donde reside el equipo de procesamiento de una o varias empresas. El centro de datos debe constar de al menos una habitación separada con suministro de energía independiente y control clima Corp (2011). El consumo de energía eléctrica dentro de un centro de datos es de alrededor de 45 % por el equipo de tecnologías de la información, en otras palabras, servidores y equipo de telecomunicaciones; el 55 % es consumida por las instalaciones, donde se incluye sistema de distribución, fuentes de alimentación ininterrumpidas, sistemas de enfriamiento Geng (2014). Así mismo, mantener un centro de datos no solo requiere constante monitoreo del equipo de cómputo, sino además, es necesario monitorizar todos los componentes del centro de datos, tales como:

- equipo de enfriamiento
- fuentes de poder
- seguridad
- crecimiento del centro de datos.

Las tecnologías de la información juegan una parte vital dentro de una empresa. Ya qué, estas tecnologías son utilizadas para ejecutar procesos críticos, por lo tanto, es necesario mantener su operación óptima. Pero, si daña un componente sufre una avería y se requiera un reemplazo de un componente físico (disco duro, fuente de poder, memoria RAM, etc). En algunas ocasiones, para realizar este reemplazo es necesario crear una ventana de mantenimiento; puede tardarse días o meses dependiendo del nivel de criticidad de los procesos que se estén ejecutando.

Existen periodos de tiempo donde no se esta utilizando al máximo el equipo de cómputo. Lo cual es un sobre aprovisionamiento de recursos de cómputo. Un estudio realizado en 2012 por *Natural Resources Defense Council*, donde se estimo que en

promedio servidores que ejecutan una sola aplicación tienden a utilizar entre 5 % y 15 % de poder de cómputo Whitney & Kennedy (2012) Por otra parte, se estima que en grandes centros de datos el porcentaje de servidores obsoletos se encuentra entre 20 % y 30 % Delforge (2014).

Por el contrario, si una empresa está teniendo un crecimiento bastante acelerado donde la capacidad de cómputo con la que cuentan actualmente será insuficiente en poco tiempo. La empresa se verá en la necesidad de adquirir nuevo equipo de cómputo. El tiempo de entrega dependerá del lugar de entrega y disponibilidad del equipo en cuestión Dell (2015). Lo cual, puede generar problemas al no tener la infraestructura necesaria para escalar el servicio, lo que conlleva a una posibilidad de pérdida de clientes e inestabilidad de los servicios.

1.2. Objetivo General

Crear un esquema para migración de infraestructura física de TI a la nube, con el fin de agilizar y reducir el proceso de migración que necesita realizar el usuario.

1.2.1. Objetivos específicos

- Crear un modelo del dominio del proceso de migración utilizando diseño basado en dominio (*Domain Driven Design*)
- Desarrollar un esquema en base al modelo conceptual
- Utilizar tecnologías de orquestación, microservicios y contenedores para la implementación del esquema.
- Desarrollar una aplicación la cual implementará el esquema de migración y utilizará las tecnologías antes mencionadas.

1.3. Justificación

El cómputo en la nube ha transformado como las tecnologías de información ofrecen sus servicios y, además, como son consumidas Pinkelman (1993). Las compañías que arriendan su infraestructura o servicios de cómputo son conocidos como proveedores de nube. Actualmente, existen múltiples proveedores de nube alrededor del mundo. Esto les da la oportunidad a las empresas de adquirir los servicios de los proveedores, lo cual podría expandir sus mercados a uno global. Estos servicios son presentados en un catálogo, en donde un usuario u organización puede seleccionar el servicio deseado, utilizarlo y al final solo pagar por el tiempo de consumo, como un pago más de utilidades. Esto les da a los usuarios la posibilidad de acceder a un gran poder de cómputo sin necesidad de una fuerte inversión inicial. Por lo que brinda a las empresas

grandes facilidades, ya que no solo reduce el costo de inversión inicial. Además, permite administrar el equipo de cómputo desde un solo lugar. Un ejemplo claro de esto, es el caso del Hospital Firmley Park donde se necesitaba reducir el tiempo administrativo del equipo de cómputo. La solución a este problema fue virtualizar las computadoras de escritorio de los empleados, con el objetivo de poder administrar todo desde un solo lugar, y se accedía a estas utilizando conexión remota (“Frimley,” 2018).

Gracias al cómputo en la nube es posible tener múltiples respaldos de aplicaciones o computadoras, no solo en diferentes computadoras, sino hasta en diferentes países. Por esta razón, es posible brindar un mejor rendimiento a los clientes, dado que es posible tener instancias de una aplicación lo más cercano posible a él.

La migración de tecnologías de la información a la nube en muchas ocasiones requiere de una gran planeación de la empresa, ya que existen múltiples factores que se deben de considerar. Adicionalmente, si no se tiene el conocimiento ni las herramientas para realizarlo puede resultar en una gran problemática. Un ejemplo de esto es no tomar en consideración los procesos que utilicen equipo *legacy* para algunas aplicaciones cruciales de la empresa Zou & Kontogiannis (2000) . El problema radica en que quizás la aplicación no interactúe correctamente la aplicación con el servidor, y se tenga que adaptar de alguna forma o modificar el código de esta. Lo cual puede generar un retraso o fracaso en la adopción del cómputo en la nube.

Por último, se creará una aplicación que utilice un esquema de detección, replicación y migración a la nube. Esta aplicación realizará lo siguiente: Durante el periodo de detección, buscará el equipo de cómputo dentro de una red, obtendrá información crucial de cada uno de estos (CPU, RAM, disco duro). Esta información se utilizará para crear máquinas virtuales. Durante el periodo de migración, en los equipos de cómputo se iniciará la tarea de creación de imágenes virtuales de estos. Estas imágenes se utilizarán para subirlas a la nube.

Capítulo 2

Marco Teórico

2.1. Virtualización

La virtualización es una forma de abstracción donde los componentes de hardware son presentarlos de una forma lógica Kusnetzky (2011), con los cuales, se pueden crear maquinas virtuales. En otras palabras, la virtualización permite utilizar los recursos físicos de una computadora para crear y alojar N cantidad de máquinas virtuales. Para poder realizar estas tareas es necesario un hipervisor. El hipervisor se encarga de administrar los recursos de cómputo y proveerlos a las máquinas virtuales. Existen dos tipos de hipervisores:

- Tipo 1 o *Bare Metal*. Es conocido como nativo, ya que corre encima del hardware de la máquina, como si fuera un sistema operativo, esto permite un aislamiento verdadero entre cada sistema operativo de cada *VM*.
- Tipo 2 o *Hosted*. Se ejecuta por encima del sistema operativo de la maquina anfitrión, esté se encarga de mostrar los recursos disponibles al hipervisor.

La “máquina virtual” fue desarrollada por *IBM* en los años 60, donde se tenía acceso concurrente e interactivo a una computadora central desde varias terminales (monitores remotos). Cada máquina virtual era una réplica representativa de la computadora central; es decir, daba la impresión de estar físicamente en una computadora real Ali & Meghanathan (2011).

2.2. Cómputo en la nube

El cómputo en la nube es un modelo para el aprovisionamiento de recursos de cómputo, los cuales son presentados en forma de catálogo. En este catálogo se presentan diversos tipos de servicios, tales como: redes, servidores, almacenamiento, aplicaciones, que pueden ser rápidamente aprovisionados y liberados con un esfuerzo mínimo de

administración o de interacción con el proveedor de servicios. Adicionalmente, el cómputo en la nube está compuesto principalmente por cinco características esenciales Mell, Grance, & others (2011).

Las principales características del cómputo en la nube son:

- Servicio en demanda. El cliente puede adquirir el poder de cómputo necesario, automáticamente sin necesidad de interacción con el proveedor.
- Conectividad. Los servicios son disponibles a través de la red, y pueden ser accedidos a través de distintas plataformas.
- Aprovisionamiento. El poder de cómputo está configurado para servir múltiples clientes en un modelo multi-cliente, con diferentes recursos físicos y virtuales que son asignados y reasignados de acuerdo con la demanda.
- Elasticidad. El aprovisionamiento puede ser elástico y escalable. En muchos casos puede ser automática, que escale rápidamente ya sea creciendo o disminuyendo dependiendo de la demanda.
- Servicios medidos. Los recursos del cómputo en la nube son controlados y optimizados utilizando una capacidad métrica, que se puede medir por almacenamiento, procesamiento, ancho de banda o usuarios activos. La utilización de recursos puede ser monitorizada, controlada y reportada. Proveyendo transparencia tanto para el proveedor como el consumidor de sus servicios.

Adicionalmente existen múltiples formas de despliegue, las cuales pueden tener diversos usos dependiendo en el ambiente en el que se utiliza. Los modelos de despliegue son:

- Nube privada. La infraestructura le pertenece a una empresa. Además, la nube puede estar administrada por la empresa dueña o por terceros. A su vez, puede residir dentro o fuera de la empresa.
- Nube comunitaria. Está constituida por múltiples comunidades u organizaciones las cuales comparten recursos para un fin en común.
- Nube pública. En este modelo la infraestructura de la nube puede ser provista por organizaciones o empresas para el uso del público y está la pertenece a proveedores de nube.
- Nube híbrida. Este modelo está compuesto de dos o más modelos.

Uno de los grandes beneficios del modelo de nube privada, es que da, la posibilidad de brindar una buena calidad de servicios, tiempo de respuesta. Para poder implementar este modelo es necesario utilizar un sistema operativo o plataforma, como los son: *Openstack*, *Cloudstack* Foundation (2017), *VMware vCloud* (“Vcloud suite,” 2018). *Openstack* es un sistema operativo que controla una gran cantidad de recursos de cómputo, almacenamiento y red *Openstack* (2015). Los recursos son administrados por medio de un panel de control, el cual, puede ser accedido por un navegador web, donde solo tienen acceso los administradores y los. Por último, *Openstack* es una

plataforma de software libre, la cual es la más utilizada por la comunidad de software libre, además, está apoyada por múltiples empresas, entre ellas, *Red Hat*, *HP*, *Google*, etc.

El computo en la nube brinda diferentes niveles de servicios, en donde dependiendo del nivel seleccionado se tendrá más capacidad de personalización. Los principales niveles de servicios IBM (2018) :

- Software as a Service (SaaS). Las aplicaciones basadas en la nube, las cuales son ejecutadas en computadoras distantes “en la nube” que pertenecen y son operadas por otros, las cuales conectan las computadoras de los usuarios vía internet y, usualmente, navegador web.
- Platform as a service (PaaS). Provee un ambiente donde todo lo que se requiere para soportar un ciclo de desarrollo e implementación de aplicaciones web, sin la necesidad de comprar o manejar hardware, software.
- Infrastructure as a service (IaaS). Provee a las compañías con recursos computacionales incluyendo servidores, redes, almacenamiento, y espacio dentro de un centro de datos con pago por uso.

Los niveles de servicios que ofrece la nube han incrementado la complejidad de los sistemas actuales, lo cual ha aumentado el número de actividades de los administradores de sistemas. Estas actividades suelen ser repetitivas como: crear máquinas virtuales, instalación de actualizaciones, software o dependencias de este, aunque muchas de estas actividades se pueden ser automatizadas utilizando *scripts*. El problema recae en que muchos *scripts* son creados para realizar tareas en un ambiente específico, además, de que en muchas ocasiones no son documentados apropiadamente, y en algunas ocasiones es necesario tener un cierto nivel de conocimiento en programación, por otro lado, para ejecutar *scripts*, es necesario obtener acceso al servidor donde serán ejecutarlos. Como resultado, ha sido necesario encontrar nuevas formas de realizar estas tareas repetitivas.

2.2.1. Ansible

En el mercado actual de las tecnologías de la información se ha incrementado la velocidad de desarrollo y mantenimiento. La unión de las áreas de desarrollo de software y operaciones ha creado una nueva área llamada DevOps Geerling (2015). DevOps utiliza prácticas del desarrollo de software en la administración de infraestructura como código (*Infrastructure-as-code* IaC). *IaC* es un algoritmo que se encarga de instalar dependencias, controladores necesarios por un programa en específico en un servidor Wittig & Wittig (2016). Por último, DevOps promueve el uso de conjuntos de *scripts*, modelos para automatización y configuración. Esto con el propósito de reutilizar código y mejorar los tiempos de desarrollo e implementación. Existen múltiples herramientas para DevOps tales como:

- *Puppet* (2017a)

- *Chef* (2018)
- *Ansible* (2017b)

La funcionalidad de estos programas es la administración y orquestación de infraestructura.

Ansible solo necesita un nodo de administración, el cual cuenta con un inventario y múltiples *playbook*, estos últimos son los archivos de configuración, implementación y orquestación. Dentro del inventario se crean grupos en, los cuales se agregan los nombres de los servidores o direcciones IPs. Dentro de *playbook* se configuran las tareas a realizar; estos archivos son creados utilizando el formato *YAML*, por otro lado, los módulos utilizan *JSON*. Se utiliza *SSH* para la conectividad remota y no requiere abrir puertos extras. Adicionalmente, solo requiere tener instalado* *Python*.

2.3. Microservicios

Tradicionalmente, las aplicaciones son programadas en una sola instancia donde todas las actividades residen en una misma aplicación, también conocido como arquitectura monolítica. Lo cual, genera una problemática al tratar de actualizar el código de la aplicación, ya que muchas ocasiones puede contar con cientos si no miles de líneas de código. Además, son difíciles de comprender y mantener. Por otro lado, cuando se requiere escalar una porción de la aplicación es necesario escalar toda la aplicación, lo cual genera un mayor costo. La figura 2.1 se puede ver varios grupos de desarrolladores trabajando en la misma aplicación sin tener definido que partes de la aplicación le pertenecen. A diferencia de la arquitectura monolítica, la arquitectura orientada a

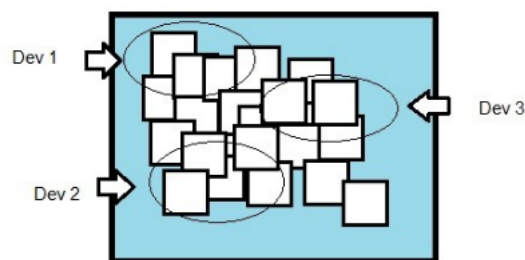


Figura 2.1: Aplicación monolítica

servicios, esta constituida por múltiples servicios, los cuales trabajan en conjunto para realizar una tarea. El desarrollo orientado a dominio tiene como objetivo desarrollar una aplicación, la cual debe expresar el objetivo de un negocio. Adicionalmente, las características. La entrega continua, virtualización, bajo demanda, automatización de infraestructura, sistemas en escala. Estas son las características que ayudan a implementar microservicios Newman (2015). Microservicios es un conjunto de servicios

autónomos que trabajan para alcanzar una meta en común. Como se habló con anterioridad las aplicaciones monolíticas limitan la forma de actualizar las aplicaciones ya que se requiere un cierto periodo de tiempo para realizar mantenimiento en el cual se realizan actualizaciones, además, de que limita las actualizaciones de esquemas y formas de manejo de datos. El alcance de cada servicio se enfoca en los alcances del negocio, esto permite reconocer con mayor facilidad el alcance o dominio de cada servicio. Uno de los beneficios que ofrece esta arquitectura es poder utilizar diversas tecnologías ya sea lenguajes de programación, base de datos ya que cada servicio es independiente de otro. Las características claves de los microservicios son:

- Diseño orientado a dómimo. Es un enfoque para el desarrollo de software con necesidades complejas mediante una profunda conexión entre la implementación y los conceptos del modelo y núcleo del negocio.
- Principio de responsabilidad simple. De acuerdo con la filosofía de Unix cada servicio es responsable de una parte única de la funcionalidad y lo hace bien Raymond (2003).
- Interfaz explícita
- *DURS independiente (Deploy, Update, Replace, Scale)* Cada servicio se puede implementar, actualizar, reemplazar y escalar de forma independiente.
- *Endpoints/ pipes* Cada microservicio posee su lógica de dominio y se comunica con otros a través de protocolos simples, como REST, el cual provee conectividad.

La forma en que se comunican entre sí los servicios es utilizando llamadas a través de la red. Esto puede ser utilizando RPC (Remote Procedure Calls) o REST (REpresentation State Transfer).

Un microservicio puede ser implementado en múltiples ambientes tales como:

- Máquinas virtuales
- Contenedores
- Plataforma como servicio (PaaS)

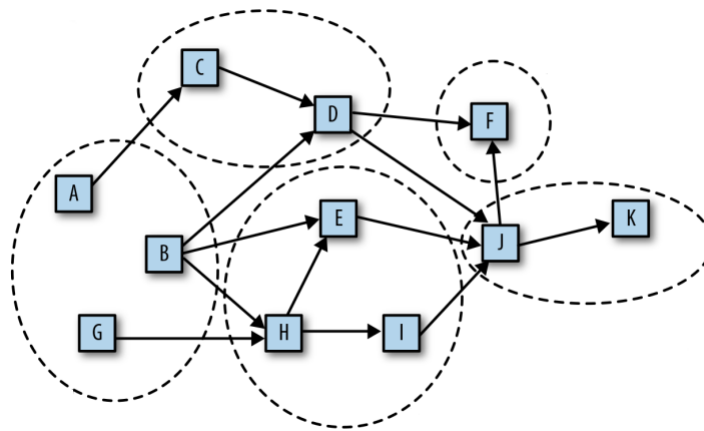


Figura 2.2: Microservicios

Múltiples microservicios pueden ser implementados en el mismo ambiente, aunque no es recomendado, ya que reduce los puntos de falla. En la figura 4 se muestra un ejemplo de una aplicación utilizando la arquitectura de microservicios, en donde se tiene bien definido los dominios de cada servicio. Como se muestra en la 2.2, se tienen multiples servicios en dominios bien definidos, ademas, de tener bien definido como se comunican entre ellos.

2.4. Diseño Basado en Dominios

Los fundamentos principales de DDD están basados en la discusión, escuchar, entendimiento, descubrimiento, y valores de negocio, todo esto para poder centralizar el conocimiento. Si eres capaz de entender el negocio en el cual se basará, por lo menos podrá participar en el modelado del software y podrá participar en el proceso de crear el lenguaje ubicuo.

Es un modelo basado en software, el cual esta basado en un dominio de negocio. También considerado como modelo objeto, donde existen objetos, los cuales tienen datos y comportamientos en base al negocio. Crear un modelo del domino es esencial para poder utilizar DDD. Utilizando DDD los modelos del dominio tienden a ser pequeños y enfocados. Permitir que los expertos del dominio y desarrolladores trabajen en conjunto, lo cual producirá un software que este basado en el negocio. Centralizar el conocimiento es clave, porque con esto la empresa es capaz de garantizar la comprensión del software.

DDD provee técnicas de desarrollo de software, las cuales se encargan del diseño estratégico y táctico. Diseño estratégico ayuda a entender cuales son las inversiones que se tiene realizar con el software, que tipo de software existe para poder obtener un software rápido y seguro. Diseño táctico ayuda a desarrollar un solo modelo de la solución Newman (2015).

2.4.1. Aspectos principales de DDD

- Acerca a los expertos del dominio y desarrolladores para trabajar en conjunto para reflejar el modelo mental del experto. Al trabajar juntos los expertos del dominio y desarrolladores su principal objetivo es crear un lenguaje ubicuo. Este lenguaje permitirá tener una mejor comunicación y un mayor entendimiento sobre el dominio del negocio.
- DDD aborda las iniciativas estratégicas de la empresa. Aunque DDD incluya técnicas de análisis, esta mas enfocado con la estrategia de dirección de la empresa. Los aspectos técnicos de la estrategia del diseño tiene como objetivo crear bounding system y preocupaciones de negocios.

- Tácticas de diseño permiten a los desarrolladores producir un software que esta correctamente codificado en base a los conocimientos de los expertos del dominio, es escalable, y permite cómputo distribuido.

2.4.2. Valores y beneficios de DDD

1. Organizaciones ganan un modelo útil de su dominio. El objetivo de DDD es invertir todos los esfuerzos en lo que importa más del negocio. Se enfoca en el dominio central (Core domain). Otros modelos existen para dar soporte al dominio central.
2. Una definición refinada y precisa del negocio es desarrollado. Cuando el modelo es refinado una y otra vez, se desarrolla un mejor entendimiento el cual se puede utilizar como herramienta de análisis.
3. Expertos del dominio contribuyen al diseño del software. Cuando los expertos comparten sus conocimientos entre ellos, permite crear un mejor entendimiento del negocio. Esto, ayuda a el crecimiento de la empresa. Adicionalmente, los desarrolladores comparten un lenguaje ubicuo con los expertos.
4. Mejor experiencia de usuario. Usualmente, la retroalimentación del usuario puede transformarse en un mejor reflejo del modelo del dominio. Cuando el software deja mucho al entendimiento del usuario, los usuarios necesitan ser entrenados para poder utilizarlo. En esencia, el usuario solo transfiere su entendimiento del software a datos, los cuales son introducidos al software. Estos datos son guardados. Sí el usuario no entiende exactamente que es lo que necesita introducir, entonces, los resultados no son los correctos.
5. Los límites son claros, los cuales son planteados alrededor de los modelos. Los desarrolladores son orientados a utilizar un enfoque de negocio.
6. La arquitectura empresarial es mejor organizada. Cuando los límites del contexto son bien definidos y cuidadosamente particionados, todos los equipos tienen un claro entendimiento de donde y porqué las integraciones son necesarias. Los límites son explícitos, y las relaciones entre ellos también.
7. Ágil, iterativo, modelado continuo es usado. El objetivo de DDD es refinar el modelo mental de los expertos del dominio a un modelo útil para el negocio.
8. Nuevas herramientas, estratégica y tácticas, son utilizadas. El límite contextual da al equipo límites de modelado en donde se crea una solución para un problema específico en el dominio. Dentro de un límite contextual un lenguaje ubicuo es creado. Este, es utilizado por el equipo y en el modelo del software. Dentro de un límite de modelado pueden utilizar tácticas: Aggregates, Entidades, Objeto Valor, Servicios, Eventos del Dominio, entre otros.

Capítulo 3

Modelacion utilizando diseño basado en dominios

3.1. Requisitos

- Permita reconocer el equipo de computo en la red.
- Crear un inventario con el equipo de computo del usuario.
 - Permita seleccionar cuales serán las computadoras para la migración.
- Facilitar la creación de respaldos de disco duro
- Migración de los respaldos a la nube

Las computadoras del usuario deberán tener lo siguiente:

- *Windows* 7, 8.1 o 10
- *Powershell* 3.0 en adelante
- *NET* 4.0
- *WinRM* deberá ser creado y activado.

El modelo siempre debe ser construido teniendo en cuenta las consideraciones de diseño y software. Esto, con el propósito de diseñar un modelo, el cual pueda ser expresado apropiadamente en software.

Una vez analizado los requisitos se encontraron tres subdominios:

- Inventario
- Migración
- Orquestación Vernon (2013)

figura pagina 55 de implementacion Cada uno de estos se explicará a continuación.

3.1.1. Subdominio Inventario

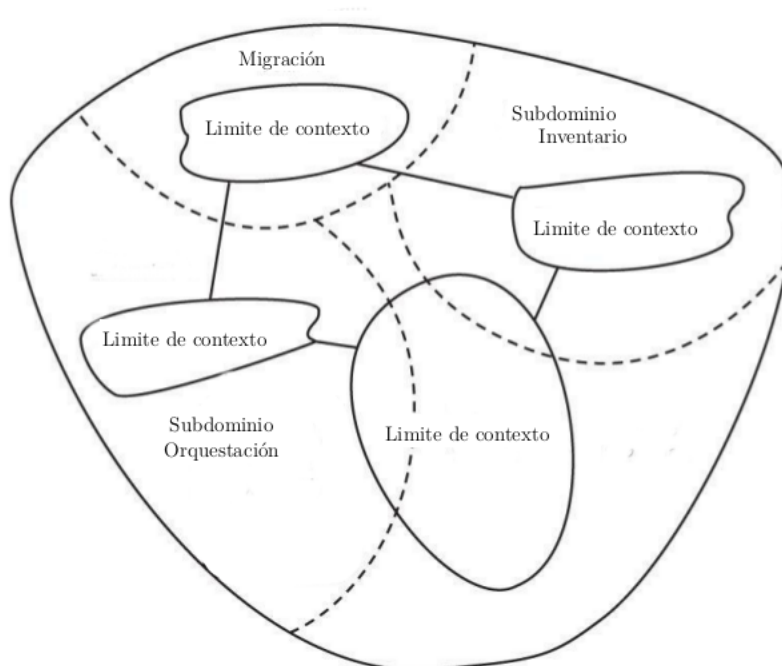


Figura 3.1: Modelado del dominio

Conclusion

Apéndice A

The First Appendix

This first appendix includes all of the R chunks of code that were hidden throughout the document (using the `include = FALSE` chunk tag) to help with readability and/or setup.

In the main Rmd file

```
# This chunk ensures that the thesishdown package is  
# installed and loaded. This thesishdown package includes  
# the template files for the thesis.  
if(!require(devtools))  
  install.packages("devtools", repos = "http://cran.rstudio.com")  
if(!require(thesishdown))  
  devtools::install_github("ismayc/thesishdown")  
library(thesishdown)
```

In Chapter ??:

Apéndice B

The Second Appendix, for Fun

References

- Ali, I., & Meghanathan, N. (2011). Virtual machines and networks-installation, performance study, advantages and virtualization options. *arXiv Preprint arXiv:1105.0061*.
- Corp, R. (2011). *R&M data center handbook*. Reichle & De-Massari AG.
- Delforge, P. (2014). America's data centers are wasting huge amounts of energy. WSP Environment & Energy, LLC Natural Resources Defense Council. Retrieved from <https://www.nrdc.org/sites/default/files/data-center-efficiency-assessment-IB.pdf>.
- Dell. (2015). Shipping and delivery. Retrieved from <http://www.dell.com/support/Contents/mx/en/mxdhs1/article/eSupport-Order-Support/shipping-and-delivery?~ck=mn>
- Foundation, T. A. S. (2017). Apache cloudstack. Retrieved from <https://cloudstack.apache.org/>
- Frimley. (2018). *Vmware.com*. Retrieved from <https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/casestudy/customers/vmware-frimley-park-hospital-13q2-en-case-study.pdf>
- Geerling, J. (2015). *Ansible for devops: Server and configuration management for humans*. LeanPub.
- Geng, H. (2014). *Data center handbook*. John Wiley & Sons.
- IBM. (2018). Learn what is cloud computing? Retrieved from <https://www.ibm.com/cloud/learn/what-is-cloud-computing>
- Kusnetzky, D. (2011). *Virtualization: A manager's guide*. .O'Reilly Media, Inc."
- Mell, P., Grance, T., & others. (2011). The nist definition of cloud computing.
- Newman, S. (2015). *Building microservices: Designing fine-grained systems*. .O'Reilly Media, Inc."
- Openstack. (2015). Software. Retrieved from <https://www.openstack.org/software/>

- Pinkelman, J. (1993). Computing changes: An industry perspective. *ACM Inroads*, 4(4), 39–42.
- Raymond, E. S. (2003). *The art of unix programming*. Addison-Wesley Professional.
- Vcloud suite. (2018). *VMWare*. Retrieved from <https://www.vmware.com/products/vcloud-suite.html>
- Vernon, V. (2013). *Implementing domain-driven design* (p. 55). Addison-Wesley.
- Whitney, J., & Kennedy, J. (2012). The carbon emissions of server computing for small-to medium-sized organization. WSP Environment & Energy, LLC Natural Resources Defense Council. Retrieved from http://www.wspenvironmental.com/media/docs/ourlocations/usa/NRDC-WSP_Cloud_Computing.pdf.
- Wittig, M., & Wittig, A. (2016). *Amazon web services in action*. Manning.
- Zou, Y., & Kontogiannis, K. (2000). Web-based specification and integration of legacy services. In *Proceedings of the 2000 conference of the centre for advanced studies on collaborative research* (p. 17). IBM Press.
- (2017a). *Puppetlabs*. Retrieved from <https://www.puppetlabs.com>
- (2017b). *Red Hat Ansible DevOps made simple*. Retrieved from <https://www.ansible.com>
- (2018). *Chef*. Retrieved from <https://www.chef.io/chef/>