

MODELADO BAJO UNA ARQUITECTURA
DE MICROSERVICIOS DE UN PROTOTIPO
DE SOFTWARE, PARA EL COMERCIO DE
TIENDAS DE BARRIO

DIEGO ALFONSO BARRIOS CONTRERAS

GRUPO 2, BOGOTÁ, 2015

Índice de contenidos

I CONTEXTUALIZACIÓN DE LA INVESTIGACIÓN	11
1 ESTUDIO DEL PROBLEMA DE INVESTIGACIÓN	12
1.1 Planteamiento del problema	12
1.2 Formulación del problema	13
1.3 Sistematización del problema	13
2 OBJETIVOS DE LA INVESTIGACIÓN	14
2.1 OBJETIVO GENERAL	14
2.2 Objetivos Específicos:	14
3 JUSTIFICACIÓN DE LA INVESTIGACIÓN	15
4 HIPÓTESIS DE TRABAJO	17
5 MARCO DE REFERENCIA	18
5.1 Marco Teórico	18
5.2 Marco Conceptual	25
5.3 Marco Espacial	26
6 ASPECTOS METODOLÓGICOS	27
6.1 Tipo de estudio	27
6.2 Método de investigación	27
6.3 Fuentes y técnicas para la recolección de la información	27
6.4 Tratamiento de la información	28
7 ALCANCES, LIMITACIONES Y RESULTADOS ESPERADOS	29
7.1 Alcances	29
7.2 Limitaciones	29
7.2.1 Resultados Esperados	30

ÍNDICE DE CONTENIDOS	2
8 Presentación de la organización	31
8.1 Mision	31
8.2 Vision	31
II DESARROLLO DE LA INVESTIGACIÓN	32
9 Diseño del producto	33
9.1 Definición de Roles	33
9.2 Mapeo visual de historias (Visual Story Mapping)	33
9.2.1 Identificación de procesos	34
9.2.2 Identificación de Funcionalidades del Software (Herramientas)	34
9.3 Historias de usuario	35
9.3.1 Producto Mínimo Viable (MVP - Minimal Viable Producto)	37
10 Arquitectura Empresarial	38
10.1 Vistas de Negocio	38
10.1.1 Punto de Vista de Organización	38
10.1.2 Punto de Vista de Actor Cooperación	40
10.1.3 Punto de Vista de Función de Negocio	41
10.1.4 Punto de Vista de Proceso de Negocio	42
10.1.5 Punto de Vista de Cooperación de Proceso de Negocio	44
10.1.6 Punto de Vista de Producto	46
10.1.7 Punto de Vista de Comportamiento de Aplicación . .	47
10.1.8 Punto de Vista de Cooperación de Aplicación . . .	49
10.1.9 Punto de Vista de Estructura de Aplicación	50
10.1.10 Punto de Vista de Uso de Aplicación	50
10.1.11 Punto de Vista de Infraestructura	52
10.1.12 Punto de Vista de Uso de Infraestructura	53
10.1.13 Punto de Vista de Organización e Implementación .	55
10.1.14 Punto de Vista de Estructura de Información . . .	57
10.1.15 Punto de Vista de Realización del Servicio	58
10.1.16 Punto de Vista de Capas	59
10.1.17 Punto de Vista de Interesado	60
10.1.18 Punto de Vista de Realización de Objetivos	61
10.1.19 Punto de Vista de Contribución de Objetivos . . .	63
10.1.20 Punto de Vista de Principios	64
10.1.21 Punto de Vista de Realización de Requerimientos .	66

ÍNDICE DE CONTENIDOS	3
10.1.22 Punto de Vista de Motivación	67
10.1.23 Punto de Vista de Proyecto	68
10.1.24 Punto de Vista de Migración	69
10.1.25 Punto de Vista de Implementación y Migración	70
11 Patrones de diseño	72
11.1 Singleton	72
11.2 Facade	73
11.3 Intercepting filter	74
11.4 Interpreter	75
11.5 Observer	76
12 Modelado de la Solución	78
12.1 Definición de la tecnología y herramientas	78
12.1.1 Back-end	78
12.1.2 Front-end	79
12.1.3 Plataforma	79
12.1.4 Despliegue	79
12.1.5 Almacenamiento de datos.	79
12.2 Definición de Microservicios	80
12.2.1 Gestión de órdenes.	80
12.2.2 Estado del pedido	81
12.2.3 Gestión de Usuarios	82
12.3 Despliegue	84
III CIERRE DE LA INVESTIGACIÓN	86
13 RESULTADOS Y DISCUSIÓN	87
13.1 Descripción del proceso de construcción	87
13.2 Descripción del prototipo	89
14 Conclusiones	95
14.1 Verificación, contraste y evaluación de los objetivos	95
14.1.1 Síntesis del modelo propuesto	97
14.1.2 Aportes originales	99
15 PROSPECTIVA TRABAJO DE GRADO	101
BIBLIOGRAFÍA	102

A	Encuestas	103
B	Descripción conceptual de la Arquitectura	107
B.1	TOGAF	108
B.2	ADM	109
B.3	ArchiMate	110
B.3.1	Conceptos Base	111
B.3.2	Capas	112
B.3.3	Framework ArchiMate	112
B.3.4	Extensión de motivación	113
B.3.5	Extensión de implementación y migración	114
B.3.6	ArchiMate y TOGAF	115
B.3.7	Meta modelo de la capa de negocio	116
B.3.8	Meta modelo de la capa de aplicación	118
B.3.9	Meta modelo de la capa de tecnología	119
B.3.10	Dependencias entre capas	120
B.3.11	Relaciones	121
B.3.12	Extensión de motivación	123
B.3.13	Extensión de implementación y migración	124

Índice de Figuras

5.1	Monolitos y microservicios. Fuente([3])	19
5.2	Arquitectura Hexagonal. Fuente([12])	20
5.3	Arquitectura de microservicios. Fuente([12])	22
5.4	Ley de Conway. Fuente([3])	23
5.5	Organización en base a la lógica del negocio. Fuente([3])	24
10.1	Metamodelo del punto de vista de Organizacion.	39
10.2	Vista Organizacion	39
10.3	Metamodelo Actor cooperación	40
10.4	Actor Cooperación	40
10.5	Metamodelo Punto de Vista de Función de Negocio	41
10.6	Punto De Vista Funcion de Negocio	42
10.7	Metamodelo Punto de Vista de Proceso de Negocio	43
10.8	Vista de Proceso de Negocio	44
10.9	Metamodelo Cooperación de Proceso de Negocio	45
10.10	Vista de Cooperación de Proceso de Negocio	45
10.11	Metamodelo Punto de Vista de Producto	46
10.12	Vista de Producto	47
10.13	Metamodelo Punto de Vista de Comportamiento de Aplicación	48
10.14	Comportamiento de Aplicación	48
10.15	Metamodelo Punto de Vista de Cooperación de Aplicación . .	49
10.16	Vista de Cooperación de Aplicación	49
10.17	Metamodelo Punto de Vista de Estructura de Aplicación . .	50
10.18	Punto de Vista de Estructura de Aplicación	50
10.19	Metamodelo Punto de Vista de Uso de Aplicación	51
10.20	Punto de Vista de Uso de Aplicación	52
10.21	Metamodelo Punto de Vista de Infraestructura	52
10.22	Punto de Vista de Infraestructura	53
10.23	Metamodelo Punto de Vista de Uso de Infraestructura	54
10.24	Punto de Vista de Uso de Infraestructura	54

ÍNDICE DE FIGURAS

6

10.25Metamodelo Punto de Vista de Organización e Implementación	55
10.26Punto de Vista de Organización e Implementación	56
10.27Metamodelo Punto de Vista de Estructura de Información	57
10.28Punto de Vista de Estructura de Información	58
10.29Metamodelo Punto de Vista de Realización del Servicio	58
10.30Punto de Vista de Realización del Servicio	59
10.31Punto de Vista de Capas	60
10.32Metamodelo Punto de Vista de interesado. [6]	61
10.33Punto de Vista de Interesado	61
10.34Metamodelo Punto de Vista de Realización de Objetivos. [6]	62
10.35Punto de Vista de Realización de Objetivos	62
10.36Metamodelo Punto de Vista de Contribución de Objetivos. [6]	63
10.37Punto de Vista de Contribución de Objetivos	64
10.38Metamodelo Punto de Vista de Principios. [6]	64
10.39Punto de Vista de Principios	65
10.40Metamodelo Punto de Vista de Realización de Requerimientos. [6]	66
10.41Punto de Vista de Realización de Requerimientos	67
10.42Metamodelo Punto de Vista de Motivación. [6]	67
10.43Punto de Vista de Motivación	68
10.44Metamodelo Punto de Vista de Proyecto. [6]	69
10.45Punto de Vista de Proyecto. [6]	69
10.46Metamodelo Punto de Vista de Migración. [6]	70
10.47Metamodelo Punto de Vista de Proyecto. [6]	70
10.48Metamodelo Punto de Vista de Implementación y Migración. [6]	71
10.49Punto de Vista de Implementación y Migración. [6]	71
11.1 Patrón Singleton	72
11.2 Patrón Facade	73
11.3 Patrón Interpreter	76
11.4 Patrón Observer	77
12.1 Modelo de Gestión de órdenes	80
12.2 Despliegue Órdenes	81
12.3 Modelo del estado del pedido.	81
12.4 Despliegue Órdenes	82
12.5 Modelo de usuarios.	83
12.6 Despliegue Órdenes	84
12.7 Diagrama de despliegue.	85

13.1 Crear proyecto en Loopback	87
13.2 Creación modelo en loopback	88
13.3 API Explorer	89
13.4 Página inicial.	90
13.5 Acceder.	91
13.6 Registrarme.	91
13.7 Inicio, con usuario registrado.	92
13.8 Realizar pedido.	93
13.9 Modificar el pedido.	93
13.10 Editar producto.	93
13.11 Estado de pedidos.	94
13.12 Administrar Pedidos.	94
14.1 Despliegue de la Solución	98
14.2 Aumento de capacidad de un microservicio	99
A.1 Resultados Compra en tiendas.	104
A.2 Resultados Frecuencia de compra.	104
A.3 Resultados Cercanía	104
A.4 Resultados Frecuencia de compra en el mismo lugar.	105
A.5 Resultados Pedidos por teléfono.	105
A.6 Resultados Horario similar	105
A.7 Resultados Acceso a Internet Diariamente.	106
A.8 Resultados Compras en linea	106
B.1 Estructura TOGAF [7]	109
B.2 ADM [7]	110
B.3 Metamodelos en diferentes niveles de especificidad [6].	111
B.4 Modelo genérico de conceptos base de ArchiMate[6].	112
B.5 Framework Arquitectural [6].	113
B.6 Relación entre los concetos motivacionales y base[6].	114
B.7 Relación ente los conceptos base, motivacionales e implementación y migración [6].	115
B.8 Correspondencia entre TOGAF y Archimate[6].	116
B.9 Meta modelo de la capa de negocio[6].	116
B.10 Meta modelo de la capa de aplicación[6].	118
B.11 Meta modelo de la capa de tecnología[6].	119
B.12 Relaciones entre las capas de ArchiMate [6].	121
B.13 Metamodelo de los conceptos de la extensión de motivación [6].	123

ÍNDICE DE FIGURAS

8

B.14 Relación entre los conceptos base y los de motivación. [6]. . .	124
B.15 Metamodelo de los conceptos de la extensión de implementación y migración[6].	124
B.16 Relación entre los conceptos base de ArchiMate y los de mi- gración e implementación. [6].	125

INTRODUCCIÓN

Iniciar por intentar definir que son los microservicios es bastante complicado pues no existe aún una definición exacta al respecto, sin embargo si se puede afirmar que el término “microservicio” fue utilizado por primera vez sobre el año de 2011 en un taller para arquitectos cerca de la ciudad de Venecia (Según lo afirma Martin Fowler, autor y orador internacional sobre desarrollo de software, especializado en análisis y diseño orientado a objetos, UML, patrones de diseño, y metodologías de desarrollo ágil, incluyendo programación extrema). En el año siguiente el mismo grupo –que estaban en el 2011-, decidió utilizar “microservicios” como el término más apropiado para describir su forma de trabajo.

Y aunque en el momento es un poco complicado de usar, puesto que en no se cuenta con una definición formal -tal como lo asegura Martin Fowler, en su página de internet-, si se puede comentar que el término “arquitectura de microservicios” ha sido usado en los últimos años para describir un estilo particular de diseño de software, en el que se ve un producto completo, como un subconjunto de servicios, que son desplegados independientemente. Y aunque la definición no esté completamente establecida, si se han determinado una serie de características comunes alrededor la organización, lógica de negocio, despliegue automático, inteligencia en los receptores, control descentralizado del lenguaje y de los datos [3].

¿Por qué microservicios? Puede ser la primera pregunta que se venga a la mente del lector, cuando aún parece ser un concepto en desarrollo; pues bien no solo es que parece ser un concepto muy interesante –teniendo en cuenta la opinión personal del autor-, sino que es un diseño que ya cuenta con casos de éxito en su uso por grandes compañías tales como Amazon, eBay y Netflix, han solucionado sus problemas adoptando lo que el día de hoy es conocido como “arquitecturas de microservicios”, en donde en cambio de construir una aplicación monstruosa, dividen su aplicación en un conjunto de servicios más pequeños e interconectados [12].

A lo largo del presente trabajo se ahondara un poco en la definición

de las características principales que se deben tener en cuenta para diseñar una aplicación bajo una arquitectura de microservicios. Adicionalmente se planea aplicar este estilo arquitectónico en un caso de estudio con muchas posibilidades de crecimiento, como lo es el comercio en tiendas de barrio, es decir, se está hablando de las tiendas pequeñas, aquellas que se encuentran a una distancia relativamente corta, cuentan con cierta variedad de productos, puedes ir a comprar al menudeo, sin realizar interminables filas, en fin, los lugares que desde pequeños conocemos, a donde probablemente nuestros padres nos enviaban a comprar lo del desayuno o mucho más. Pero ¿Por qué concentrarse en este tipo de negocios?, pues bien, día a día las grandes compañías (grandes superficies), invierten importantes cantidades de recursos en publicidad, fidelización, entre otros, lo que impide que negocios de menor tamaño (como las tiendas de barrio, que cuentan con una preferencia del 53% por los consumidores, de acuerdo a un estudio realizado en el 2013 por Kantar Worldpanel [1]), puedan competir al mismo nivel, así que con el desarrollo de este trabajo se desarrollara la posibilidad de brindar un elemento tecnológico adicional para mejorar la competitividad, y se centrará en las aplicaciones tecnológicas (web, aplicaciones móviles), que –las tiendas pueden ofrecer a sus usuarios, con las que logren conocerlos y acercarse más a sus preferencias, pero de una manera en que la información esté disponible y asequible y así identificar sus hábitos y ofrecer los productos que realmente necesiten.

PARTE I

**CONTEXUALIZACIÓN DE
LA INVESTIGACIÓN**

Capítulo 1

ESTUDIO DEL PROBLEMA DE INVESTIGACIÓN

1.1 Planteamiento del problema

En la actualidad, las grandes compañías buscan acercarse mucho más a sus clientes, con este objetivo invierten un gran esfuerzo en intentar comprender como se comporta el consumidor, cuáles son sus hábitos, preferencias, gustos, etc. Como consecuencia podemos evidenciar la gran cantidad de campañas publicitarias y diversas estrategias de ventas (elementos que se utilizado durante mucho tiempo), sin embargo dado el crecimiento tecnológico acelerado en el que se ha estado viviendo, las compañías tienen que adaptarse y explotar más a fondo la forma de utilizar estos medios. Las aplicaciones -tanto web como aplicaciones móviles- que permiten comprar mercancía, recibir catálogos, realizar pedidos a domicilio, ofrecer promociones, por tan solo nombrar algunas de sus funcionalidades. Con el uso de ellas, intentan facilitar a sus clientes la manera de realizar sus compras y obtener información de su mercancía y con ello incrementar sus ventas.

Con el afán de explotar estas oportunidades de comercialización, las grandes superficies (o tiendas de cadena) realizan fuertes inversiones con el fin de ser más competitivas, lo que es muy provechoso para el consumidor final pero ocasiona que negocios mucho más pequeños –nos referimos a las tiendas de barrio- sufran el impacto en sus ventas, debido a que estos, no cuentan con los mismos recursos disponibles para invertir y que les permitan competir con las grandes cadenas.

Teniendo en mente esto, contar con una herramienta que les permita a estas tiendas ser más competitivas y acercarse a sus clientes es de vital im-

portancia para mantenerse activo y competitivo en el negocio, pero acercarse no es suficiente, lo clave es ofrecer un valor agregado a sus clientes, un elemento adicional que les facilite la vida, por ejemplo que la herramienta asista en recordar sus compras, ofrecer opciones de compra con base en su historial y su necesidad. Con este objetivo en mente la ingeniería de software debe buscar una forma de satisfacer las diferentes reglas de negocio que existen o puedan surgir.

Analizando esta problemática desde la perspectiva de la arquitectura de software, múltiples modelos se ofrecen para satisfacer esta necesidad. Uno de los que –a perspectiva del autor- se muestra como una opción interesante es la arquitectura de microservicios, la cual cuenta con casos de éxito muy interesantes, de los cuales se puede aprender para enfocarlo en el ámbito colombiano, en un caso específico como el mercadeo de las tiendas de barrio.

1.2 Formulación del problema

¿Una herramienta tecnológica le permitirá a tiendas de barrio facilitar a sus usuarios la adquisición de sus productos, ser más competitivos y mejorar sus ventas?

1.3 Sistematización del problema

¿Es posible aplicar una arquitectura de micro servicios para dar solución al desarrollo del prototipo?

¿Los tenderos lograrán usar esta herramienta en sus procesos de venta?

¿Los usuarios usaran la herramienta, en cambio de utilizar el teléfono para sus domicilios?

¿Mejorarán sus ventas las tiendas de barrio?

Capítulo 2

OBJETIVOS DE LA INVESTIGACIÓN

2.1 OBJETIVO GENERAL

Modelar un prototipo de una aplicación que mejore y facilite los servicios ofertados por parte de las tiendas de barrio a sus clientes, y así mejorar sus ventas. Mediante el uso de un enfoque de microservicios.

2.2 Objetivos Específicos:

1. Identificar necesidades y oportunidades, mediante mecanismos de recolección de información que facilite entender las necesidades de tenderos y clientes, y que a su vez permita plantear un plan de crecimiento del producto.
2. Diseñar un modelo de la aplicación -de comercio de tiendas- utilizando una arquitectura de microservicios, con el fin de ejemplificar este enfoque en un proyecto para tiendas de barrio.
3. Construir el prototipo de la aplicación siguiendo los lineamientos de la arquitectura para identificar ventajas y desventajas experimentadas durante el proceso.

Capítulo 3

JUSTIFICACIÓN DE LA INVESTIGACIÓN

Las compañías día a día cuentan con más competencia, por lo que es imprescindible para ellas lograr la mayor cantidad de ventajas, en mercados muy competitivos. Por ende, el desarrollo de software debe estar en capacidad de poder explotar estas ventajas –lo antes posible–, ya sea mediante sistemas de administración, sistemas control, contables, o si son los propios productos que utilizan sus usuarios. Teniendo en cuenta esta finalidad, se debe estar siempre preparado para el cambio o para crecer conforme el negocio lo requiera.

Intentando resolver esta problemática, la arquitectura de microservicios plantea una serie de ventajas, las cuales pueden ser aprovechadas por las compañías, pero también es posible que genere desventajas, dependiendo del proyecto o mercado. Con el fin de realizar un análisis más profundo de la aplicabilidad de esta perspectiva en un mercado colombiano específico, se desea diseñar una aplicación bajo una arquitectura de microservicios.

El caso de estudio tomado, es el comercio para las tiendas pequeñas –comúnmente llamadas “tiendas de barrio”–, que al igual que cualquier negocio busca incrementar sus ventas. Para esto se sugiere intentar aprovechar las ventajas que la tecnología les puede ofrecer para acercarse más a sus clientes (gracias a que cada día la tecnología se vuelve más importante en la vida cotidiana); y aunque sus competidores (tiendas de cadena, por ejemplo) no escatiman en recursos para ampliar su participación y acercarse a sus clientes –usando la tecnología–, las tiendas son aun el medio habitual de compra de los consumidores con un 53% (de acuerdo con el LatAm Retail Overview 2014 realizado por Kantar Worldpanel, en 2013), principalmente

explicado por su cobertura que cuenta con más de 307.000 establecimientos de los 577.331 que están establecidos en el país (De acuerdo a Nielsen en el 2013). Con el pasar del tiempo esta participación empieza a disminuir como lo indica el 1% que disminuyo respecto al año 2012 [1], con lo que un cambio es necesario para continuar activo en el mercado (dicho esto como una opinión del autor).

Con base en los análisis hechos se puede ver la necesidad que tienen las tiendas por permitir a los clientes realizar pedidos de manera más sencilla, enviar recomendaciones a sus clientes y ofrecer recordatorios pueden ser facilidades que las pequeñas tiendas pueden ofrecer a sus clientes, con el propósito de acercarse a ellos, darles más valor y con ello mantenerse competitivos con respecto a las grandes compañías.

Desde el punto de vista de la arquitectura de software, existen diferentes enfoques, todos tratando de dar solución a las necesidades de negocio, para este proyecto se plantea el implementar una solución que utilice una arquitectura de microservicios, la cual cuenta con casos de éxito tales como Platzi[5], Amazon o el cada vez más popular Netflix[9], los cuales han logrado implementar una solución utilizando este tipo de arquitectura de manera satisfactoria (analizando este punto tan solo desde un punto de vista del crecimiento que han tenido estas compañías), ayudándoles a convertirse en compañías exitosas. Teniendo como ejemplo estos casos, se desea ver su implementación en un ambiente local –en primera medida- y en un mercado específico que será el comercio en tiendas de barrio, tener un breve abrebotas de cómo se plantea una solución bajo esta línea arquitectural.

Capítulo 4

HIPÓTESIS DE TRABAJO

Los comercios pequeños (tiendas de barrio), pueden ofrecer una herramienta que les permite automatizar la recepción de pedidos e interpretar los hábitos de compra de sus clientes, facilitando la interacción de ellos y ofreciéndoles un valor agregado (como lo intentan las grandes superficies). Lo que puede generar una ventaja competitiva y por ende un aumento en sus ventas.

Capítulo 5

MARCO DE REFERENCIA

5.1 Marco Teórico

Arquitecturas de microservicios

Para dar inicio, intentaremos contextualizar un poco en qué consiste y de donde proviene la arquitectura de microservicios. Lo primero que vale la pena comentar es de donde nace el término microservicio, pues bien, de acuerdo a Martin Fowler el término “microservice” fue discutido en un taller para arquitectos de software cerca a Venecia en mayo de 2011, y se utilizó para describir un estilo arquitectural que muchos habían empezado a explorar recientemente. Luego en mayo de 2012 este mismo grupo decide optar por el nombre de “microservicios” como el término más apropiado [2].

Algo que debemos observar es que aunque el término puede ser relativamente nuevo su idea ha existido por mucho tiempo, sin embargo durante los últimos años ha sido utilizado para describir un modo particular de diseñar software, como un conjunto de servicios desplegables independientemente. Y aunque no existe una definición específica de este estilo arquitectural, si cuenta con una serie de características (se abordarán más adelante) comunes alrededor de la organización, lógica de negocio, despliegue automático, entre otras [3].

Para facilitar la comprensión del estilo de microservicios es útil compararle con el estilo monolítico, el cual radica en construir una aplicación como una unidad. Usualmente las aplicaciones empresariales constan de 3 partes, una parte cliente, una parte de manejo de datos y una parte servidor, la cual contiene la lógica de dominio de la aplicación. En sí, las aplicaciones monolíticas pueden ser exitosas, sin embargo con el paso del tiempo más personas se sienten frustradas, puesto que los cambios constantes de requer-

imientos causan que tenga modificar toda la aplicación, y mantener este tipo de aplicaciones modulares se vuelve más complicado. Y el escalar (tema muy relevante) se convierte en un escalado completo de la aplicación y no del segmento que realmente lo requiere.

Para hacer frente a esta problemática la arquitectura de microservicios dicta el construir conjuntos de servicios. En donde cada uno de ellos es independiente del otro, hasta pueden ser escritos en lenguajes diferentes y mantenidos por equipos diferentes. La Figura 5.1, muestra como se ve una aplicación con estilo monolítico y una con estilo de microservicios [3].

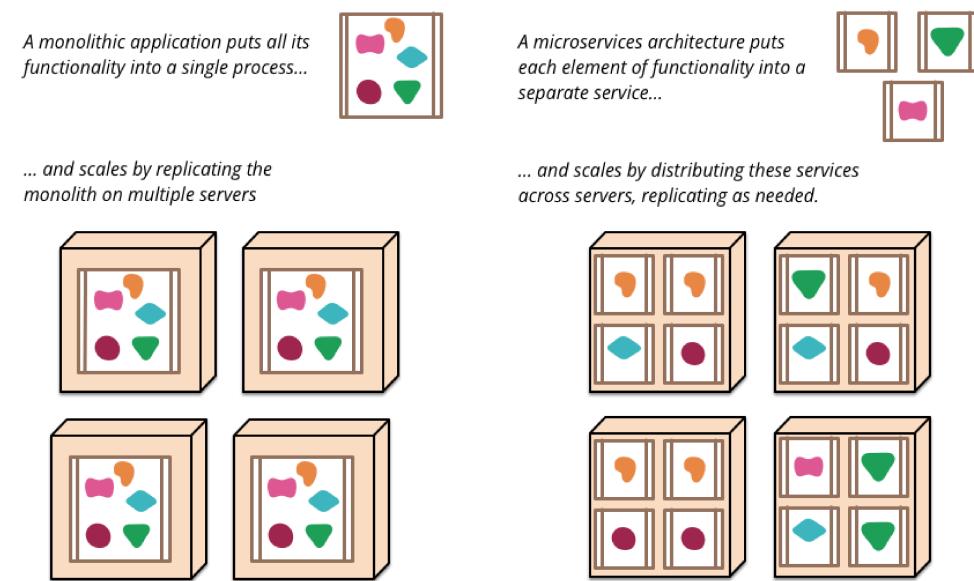


Figure 5.1: Monolitos y microservicios. Fuente([3])

Ejemplo

En el siguiente ejemplo tomado del blog de nginx [9], permite entender como se ve una arquitectura monolítica y una de microservicios. El ejemplo consiste en una aplicación que compita con “Uber” y “Hailo”. Después de un análisis se plantea una solución bajo una arquitectura hexagonal, como se ve en la Figura 5.2.

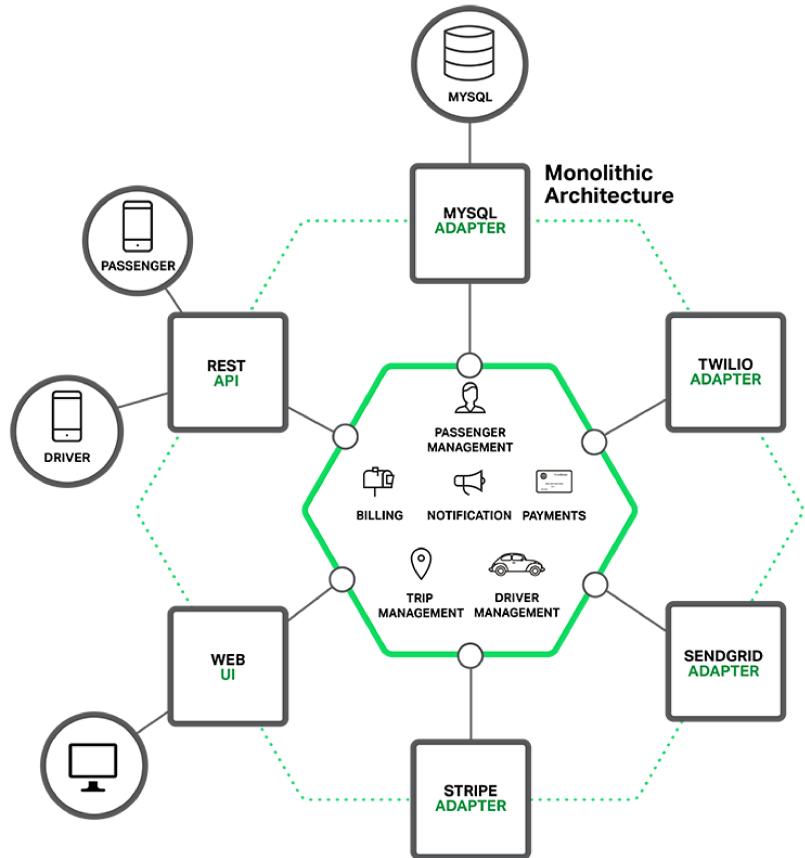


Figure 5.2: Arquitectura Hexagonal. Fuente([12])

En el core de la aplicación se encuentra la lógica de negocio, que esta implementada por diferentes módulos. Alrededor del core se encuentran los adaptadores que se encargan de comunicarse con el exterior (por ejemplo componentes de accesos a base de datos, componentes de manejo de mensajes, web, entre otros).

Aunque la aplicación es modular, su despliegue es como un monolito, y depende mucho del empaquetado que el lenguaje en el que estén construidas (por ejemplo java genera un WAR).

Estas aplicaciones tienen múltiples ventajas como el hecho que son simples de desarrollar pues cuentan con IDEs especializados para esto. También son simples para su prueba, y también para su despliegue, pues básicamente consiste en un copiado del archivo empaquetado al servidor.

Desafortunadamente este enfoque simple tiene grandes limitaciones. Aplicaciones exitosas tienden a crecer y con el tiempo se convierten en inmensas aplicaciones. Lo que ocasiona que con cada iteración se adicionen más y más funcionalidades, lo que conlleva a que se conviertan en monolitos monstrosos, no solo en tamaño sino en complejidad. Adicionalmente también afectan el proceso del despliegue, puesto que con aplicaciones más grandes se requiere de más tiempos para iniciar, y el equipo no la entiende completamente (salvo unos pocos).

Como respuesta a esto muchas organizaciones como Amazon, eBay y Netflix, han solucionado este problema adoptando lo que es hoy conocido como arquitectura de microservicios.

Para continuar con el ejemplo, cada servicio que se genera tendrá su propia arquitectura hexagonal y se interconectaran como se ve en la Figura 5.3.

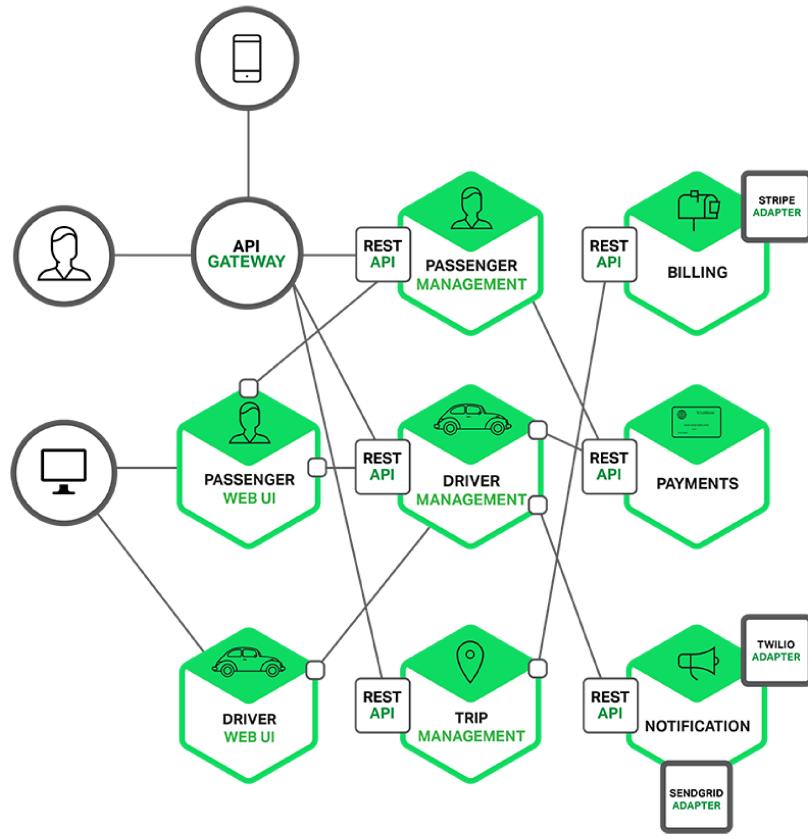


Figure 5.3: Arquitectura de microservicios. Fuente([12])

Cada área funcional es implementada ahora por su propio microservicio. Más aun la web es dividida en un conjunto de aplicaciones web mas simples [12].

Características

Pequeño y enfocado en hacer una sola cosa bien [10]

El código fuente crece frecuentemente, con cada adición de una nueva funcionalidad, lo que con el tiempo se convierte en un problema pues cada vez es más difícil saber donde se debe hacer un cambio.

Los microservicios intentan aislar la especialidad de una lógica de negocio específica en un solo lugar, lo que permite disminuir la tentación de hacer crecer mucho nuestro componente.

La pregunta recurrente es ¿Qué tan pequeño es pequeño? La cantidad de líneas de código es problemático, pues cada lenguaje se expresa de manera diferente. Jon Evans en RealEstate.com.au en Australia caracteriza los microservicios como algo que pueda ser reescrito en dos semanas.

Un factor adicional para definir para responder que tan pequeño, es que tan bien se alinea a las estructuras del equipo. Si el código base es muy grande para ser manejado por un equipo pequeño, es un indicador de que puede ser dividido.

Organizado en torno a la lógica de negocio [3]

Cuando se busca dividir una aplicación en partes, a menudo la administración se enfoca en la capa tecnológica, separando los equipos de UI, servidor y base de datos. Cuando los equipos están separados, necesitan seguir un protocolo para tiempos y presupuesto de otros equipos. Lo que hace que el equipo intente separar su trabajo del de los otros. Con lo que es un ejemplo de la ley de Conway (Cualquier organización que diseña sistemas producirá un diseño cuya estructura es una copia de la estructura de comunicación de la organización. Figura 5.4).

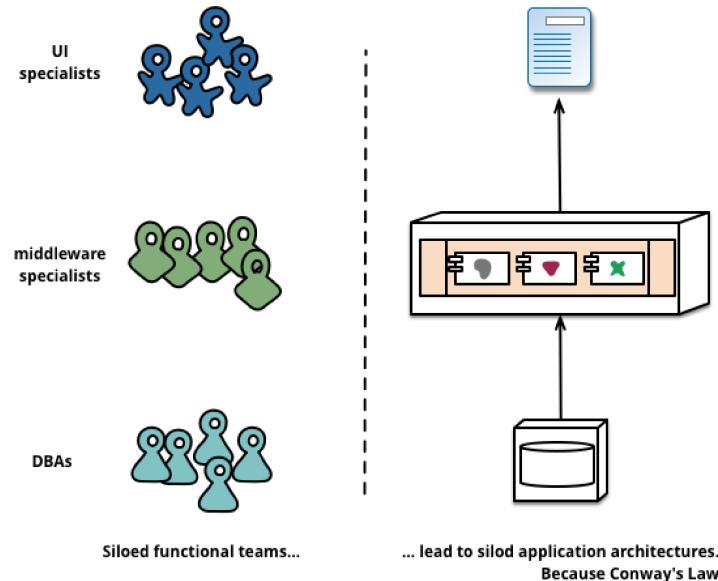


Figure 5.4: Ley de Conway. Fuente([3])

El enfoque de microservicios es diferente pues divide las organizaciones alrededor de la lógica de negocio. Esta organización es interdisciplinaria pues

mescla UI, lógica de negocio y base de datos, lo que conlleva a que el equipo tenga diferentes habilidades, la Figura 5.5 nos permite ver como se aplicaría la ley de Conway a esta estructura.

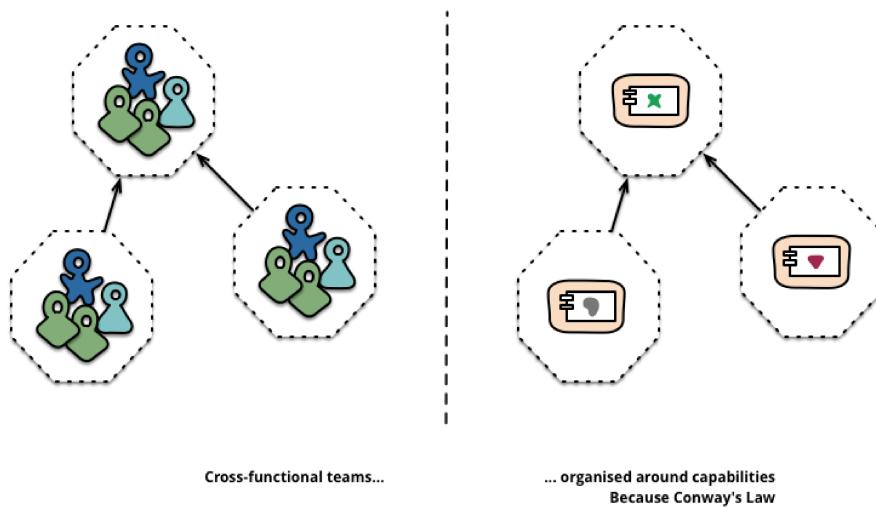


Figure 5.5: Organización en base a la lógica del negocio. Fuente([3])

Autonomía [10]

Los microservicios son una entidad separada. Deben ser desplegados como un servicio aislado en infraestructuras que se enfocan en ofrecer plataformas como servicio (PAAS por sus siglas en inglés), o deben estar en su propio sistema.

Toda la comunicación entre servicios son llamados de red, para reforzar la separación entre servicios y evitar los peligros del estrecho acoplamiento.

Productos no proyectos [3]

Los esfuerzos de la mayoría de los desarrollos usan los modelos de proyectos, en los cuales el objetivo es desarrollar piezas de software que una vez terminadas son consideradas completas. Una vez terminado el software es manejado por el encargado a una empresa –o área- de mantenimientos y el equipo se disuelve.

Los promotores de microservicios intentan evitar este modelo con la idea que un equipo debería manejar un producto a lo largo del ciclo de vida de este. Como inspiración se toma la idea de Amazon que dice: “lo construyes, lo ejecutas”, en donde se empodera al equipo a que toma la completa responsabilidad sobre el software en producción.

Lo anterior no quiere decir que en un enfoque monolítico no pueda ser aplicada esta ideología, solo que con servicios más pequeños es más sencillo lograr una familiaridad entre el equipo desarrollador y el producto.

Diversidad de tecnología [2]

Dado que cada microservicio es una unidad de despliegue independiente, se cuenta con una considerable libertad en las elecciones de tecnología que se puedan tomar. Los microservicios pueden usar diferentes lenguajes, librerías o almacenamiento de datos, con lo que la elección se basa en cuál es el problema a resolver, y con base en ello escoger la mejor herramienta, o a un nivel más granular la librería correcta, pues usualmente en una aplicación monolítica, todos los módulos deben usar la misma versión.

No hay que olvidar que no se debe sobreponer el conocimiento que puede manejar la organización, pues puede ocurrir que al existir demasiada diversidad tecnológica esto se convierta en un problema más que en una solución. Por ello se recomienda tener límites en cuanto a las opciones que se puedan tener.

Otro punto muy importante es el poder experimentar. Es muy importante pues con aplicaciones monolíticas hacer un cambio de tecnología resulta casi imposible, mientras que si esto se desea hacer con un microservicio es mucho más sencillo dado su tamaño y control.

5.2 Marco Conceptual

Microservicio: No existe una definición específica al respecto, pero para fines prácticos podemos definirlos como una serie de pequeños servicios, autónomos y que trabajan conjuntamente.

Tienda de barrio: Hace referencia a aquellos negocios que están enfocados a la venta al detal, su tamaño es pequeño y son utilizados en su mayoría para consumos menores.

Ley de Conway: Las organizaciones que diseñan sistemas están limitadas a producir diseños que son copias de las estructuras de comunicación de estas organizaciones. En pocas palabras, el software desarrollado, será una imagen de la estructura organizacional.

Grandes superficies: se llama así a aquellas tiendas y mercados de gran tamaño que ofrecen una gran cantidad de diferentes productos.

Arquitectura monolítica: determina un estilo arquitectural que en la cada una de las funcionalidades discernibles, no son componentes arquitecturales separados, por ejemplo un archivo WAR que contiene la interfaz, lógica y manejo de datos.

5.3 Marco Espacial

El marco planteado para realizar el proyecto es Colombia, y por facilidad del autor la ciudad escogida es Bogotá. En una primera fase se plantea iniciar por realizar la recolección de información (encuestas, entrevistas, observación, etc.), en un sector empresarial de la ciudad de Bogotá en la localidad “Santa Fe”, alrededor del Parque Central Bavaria, ubicado en la carrera 13 con calle 28.

Capítulo 6

ASPECTOS METODOLÓGICOS

6.1 Tipo de estudio

El tipo de estudio seleccionado para el presente trabajo es explicativo, puesto que se busca identificar el impacto que una herramienta tecnológica puede tener en el modelo de negocio de las tiendas de barrio, para ello se requiere ahondar en el comportamiento y necesidades tanto de los clientes como de los dueños de este tipo de negocios. Con el entendimiento –parcial e incremental– de las necesidades, se quiere lograr que la herramienta genere un valor adicional para ambas partes.

6.2 Método de investigación

Se plantea utilizar un método sistemático, que permita evaluar los usuarios que conformarían el modelo (componentes), y las necesidades que requieren del otro, para así lograr sus respectivos objetivos (relaciones), por ejemplo para los tenderos es incrementar sus ventas mientras que para los usuarios es tener una mejor experiencia y facilitar sus métodos de compra.

6.3 Fuentes y técnicas para la recolección de la información

Con el fin de comprender necesidades de las partes implicadas en el modelo, se plantea principalmente el uso de fuentes primarias como: Observación

(desde la experiencia personal del autor), el uso de encuestas a posibles consumidores en primera medida y posteriormente entrevistas enfocadas más hacia tenderos, con el que se pueda discutir las necesidades de sus clientes.

6.4 Tratamiento de la información

Con la información recolectada se plantea presentar un resumen cualitativo de los resultados encontrados, desde las dos perspectivas a estudiar, es decir, desde el dueño del negocio y el comprador. Partiendo de este análisis se refinara el back log, que brindara una guía un poco más completa de hacia dónde se podría dirigir el proyecto en una próxima fase.

Capítulo 7

ALCANCES, LIMITACIONES Y RESULTADOS ESPERADOS

7.1 Alcances

El proyecto pretende realizar un análisis, diseño y desarrollo de una herramienta que aplique los lineamientos de una arquitectura de microservicios, con el fin de exemplificar el uso de este tipo de arquitecturas, así que dentro del alcance del proyecto está el integrar al menos dos microservicios, y con ello realizar un corto primer análisis de que ventajas y desventajas se han identificado a lo largo del proyecto.

Adicionalmente es importante mencionar que el prototipo busca ser completamente funcional, y contara con una fase de incepción que brinde un punto de partida para que el proyecto vaya madurando con cada iteración, con esto en mente el prototipo podría ser usado en el modelo de negocio de tiendas de barrio.

7.2 Limitaciones

Teniendo en cuenta las restricciones –principalmente de tiempo y presupuesto– del proyecto se han identificado ciertas limitaciones, sin duda la principal radica en que aunque el prototipo es funcional, el tiempo de implementación y capacitación no está contemplado en la ejecución de esta fase del proyecto, por lo que los resultados serán analizados en un ambiente no productivo.

CAPITULO 7. ALCANCES, LIMITACIONES Y RESULTADOS ESPERADOS30

La recolección de información –tal como encuestas u otras-, se realizará progresivamente, dado lo que el proyecto se plantea para ser incremental, con lo que el primer ciclo de recolección de información (pensado para no ser muy largo) no se pretende abarcar todos los posibles requerimientos que puedan surgir en este modelo de negocio.

7.2.1 Resultados Esperados

Con la finalización de la esta primera fase del proyecto se espera contar con un sistema estable capaz de cumplir con los objetivos seleccionados. Teniendo en cuenta esto, se pretende obtener:

- Documentación básica del proyecto, que esté enfocada a la especificación de la arquitectura a usar (dado que al usar scrum como metodologías de trabajo, se da más importancia al desarrollo de software funcional que tener una extensa documentación).
- Prototipo funcional del proyecto, que cuente con al menos dos microservicios integrados.
- Una descripción de las ventajas y desventajas evidenciadas durante el proyecto, con respecto a la aplicación de una arquitectura de microservicios.

Capítulo 8

Presentación de la organización

8.1 Mision

Apoyar a los dueños de pequeñas tiendas de barrios(tenderos), mediante el uso de herramientas tecnológicas, que les permita acercarse a sus clientes y abrir nuevas posibilidades a sus negocios, creando una ventaja competitiva y permitirles competir con las grandes superficies.

8.2 Vision

Proveer un sistema de administración y ventas para los tenderos, de manera económica, que sea conocida y usada por ellos en menos de 3 años, y que a su vez les permita seguir competitivos.

PARTE II

**DESARROLLO DE LA
INVESTIGACIÓN**

Capítulo 9

Diseño del producto

En el presente capítulo se describirán los roles y la definición de las historias de usuario iniciales que son planteadas. A partir de estas se seleccionarán las más relevantes para crear un producto mínimo viable (MVP). Adicionalmente, se realizó un análisis sobre los resultados de una encuesta (Ver Anexos) que ayudó a plantear el desarrollo del producto.

9.1 Definición de Roles

- Dueño Tienda: Persona a la que pertenece la tienda.
- Tendero: Persona que se encarga de despachar y organizar los pedidos. Usualmente suele ser el mismo dueño de la tienda.
- Cliente: Persona que compra los productos a la tienda.
- Domiciliario: Es la persona que se encarga de llevar los pedidos solicitados de la tienda a donde el cliente especifica.

9.2 Mapeo visual de historias (Visual Story Mapping)

Jeff Patton plantea una técnica de Análisis Ágil llamada Mapeo visual de historias o Visual Story Mapping[11]. En esta técnica se busca identificar objetivos, actividades y herramientas que se requieren para generar un mínimo producto comercializable. Para nuestro caso de estudio lo primero que se va a identificar son los procesos (o actividades) que cumplen nuestro objetivo de aumentar las ventas y fidelizar los clientes. Posteriormente se identificarán

a grandes rasgos las funciones (o herramientas) que serán ofrecidas por el sistema.

9.2.1 Identificación de procesos

Los procesos identificados para el sistema son:

- Venta de productos
- Gestión de pedidos
- Preparación de pedidos
- Despacho de pedidos.
- Mercadeo.

9.2.2 Identificación de Funcionalidades del Software (Herramientas)

- Venta de productos.
 - Crear pedido de compra
 - Pago del envío con tarjeta crédito.
 - Pago del envío con tarjeta débito.
 - Pago en efectivo.
 - Devolución de pago en caso de eventualidad.
 - Confirmación de pago.
 - Identificar el cliente
 - Visualizar el histórico de las compras.
- Gestión de pedidos
 - Atender un pedido de compra.
 - Validar que el pedido haya sido atendido.
 - Asegurar que en el inventario cuente con los productos especificados.
 - Permitir realizar un pedido personalizado.
 - Comunicación con cliente sobre aclaraciones del pedido.

- Agendar entrega de pedido.
- Visualizar los pedidos realizados.
- Preparación de pedidos
 - Reducir del inventario los productos.
 - Confirmación del pedido alistado.
 - Identificar lugar de entrega.
 - Generar factura de compra.
- Despacho de pedidos.
 - Buscar un domiciliario disponible.
 - Confirmar el domiciliario al cliente.
 - Visualizar la ubicación de mi pedido.
 - Seleccionar la ruta optima de entrega de los pedidos.
- Mercadeo
 - Permitir identificar tendencias de compra.
 - Sugerir promociones con base en productos a perecer.
 - Identificar los productos menos y más vendidos.
 - Permitir crear promociones personalizadas a clientes.

9.3 Historias de usuario

Con base en el desarrollo del VSM (Visual Story Mapping), se plantean algunas historias de usuario que nos darán el punto de partida para el desarrollo del producto. Es importante aclarar que dada la naturaleza iterativa que se está utilizando, las historias serán adicionadas y/o modificadas según la evolución del producto, por lo que entrar a detallar todo en este punto no generaría mucho valor. A continuación se listan las historias de usuario (en ningún orden específico) que conforman nuestro backlog.

1. Como comprador deseo solicitar a mi tendero productos desde mi computador para facilitar mis compras.
2. Como comprador deseo solicitar a mi tendero productos desde mi celular para facilitar mis compras.

3. Como tendero deseo tener una lista ordenada de los pedidos recibidos para poder despacharlos y no perder ninguno.
4. Como tendero deseo poder administrar la lista de pedidos recibidos para poder organizar mi negocio.
5. Como comprador deseo programar la hora de recepción de mis pedidos para poder planear mejor mis actividades.
6. Como cliente deseo seleccionar productos y agregarlos a mi pedido de forma gráfica para facilitar mi forma de compra.
7. Como tendero deseo que los productos ofertados estén enlazados con mi inventario para poder ofrecerles productos en existencia a sus compradores.
8. Como tendero deseo tener control de mi inventario para saber qué productos necesito conseguir para tener un inventario atractivo a mis compradores.
9. Como tendero deseo conocer por medio de avisos los productos que están por terminarse de mi inventario para tener un inventario atractivo a mis compradores.
10. Como tendero deseo tener reportes que me permitan ver los productos que tienen mayor solicitud y cuales no para poder abastecer mejor mi tienda.
11. Como comprador deseo tener un recordatorio de mis pedidos más frecuentes según mis preferencias para ayudarme a organizar mis compras.
12. Como comprador deseo ver un historial de mis compras para conocer mis gastos.
13. Como comprador deseo poder solicitar productos a diferentes tenderos para tener más opciones y variedad.
14. Como comprador deseo realizar pedidos seleccionando el producto y que el pedido se distribuya a diferentes tenderos según las mejores ofertas.
15. Como tendero deseo crear campañas promocionales, e informar a mis clientes para mejorar mi negocio.

16. Como comprador deseo buscar promociones según mis preferencias para obtener mejores beneficios.
17. Como comprador deseo que mi ubicación sea obtenida automáticamente para facilitar el uso de la aplicación.
18. Como tendero deseo que la orden lista genere una factura sea impresa para entregársela al domiciliario y este al comprador.
19. Como domiciliario deseo agrupar mis entregas para hacer más efectivo mi trabajo.

9.3.1 Producto Mínimo Viable (MVP - Minimal Viable Producto)

Con el objetivo de tener un producto mínimo viable (MVP) se seleccionaron las historias de usuario mas importantes, en las cuales se enfocara esta primera entrega. Estas son:

- Como comprador deseo solicitar a mi tendero productos desde mi computador para facilitar mis compras.
- Como tendero deseo tener una lista ordenada de los pedidos recibidos para poder despacharlos y no perder ninguno.
- Como tendero deseo poder administrar la lista de pedidos recibidos para poder organizar mi negocio.
- Como comprador deseo que mi ubicación sea obtenida automáticamente para facilitar el uso de la aplicación.

Capítulo 10

Arquitectura Empresarial

El presente capítulo se van a representar los diferentes puntos de vista requeridos para especificar la arquitectura del presente proyecto. Cada Punto de vista está compuesto por el metamodelo de la vista con una breve explicación de este, seguido por la realización del metamodelo aplicado al caso de negocio de compra de productos a través de un portal web. Cabe resaltar que aunque la solución puede ser utilizada por cualquier tipo de negocio de características similares, los modelos se plantean teniendo como base la aplicación de la solución para una sola empresa.

10.1 Vistas de Negocio

10.1.1 Punto de Vista de Organización

Se enfoca en representar la organización –interna- de la empresa. Es útil para identificar competencias, autoridad y responsabilidades. El metamodelo puede ser apreciado en la Figura 10.1.

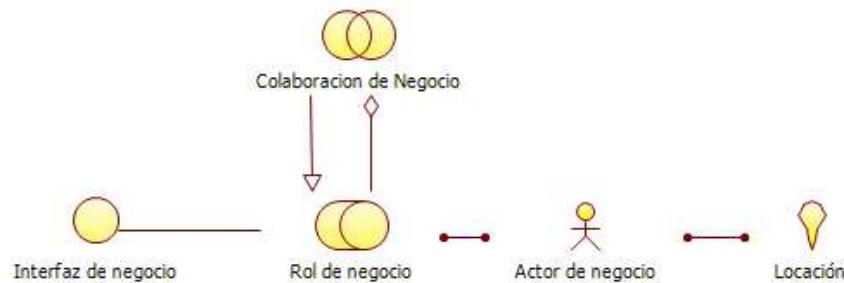


Figure 10.1: Metamodelo del punto de vista de Organizacion.

La Figura 10.2 representa la organización de la tienda de barrio, con sus principales roles el tendero y el domiciliario que trabajan conjuntamente para despachar mercancía, que es ofrecida por internet y físicamente, en la tienda.

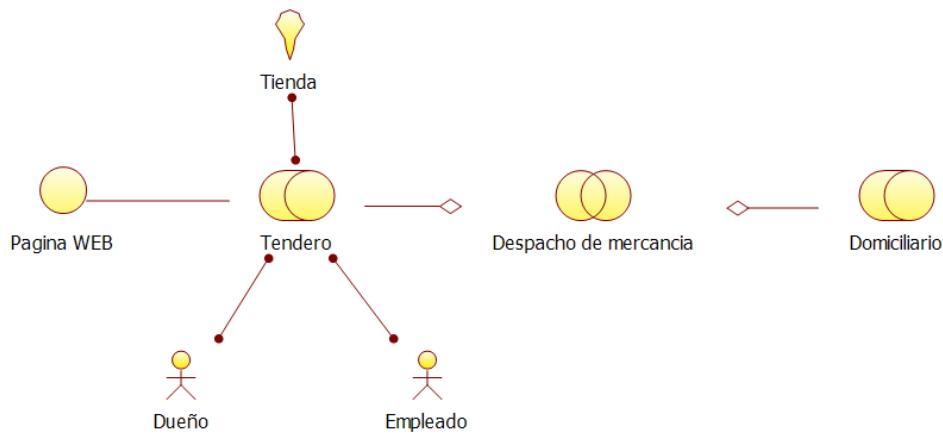


Figure 10.2: Vista Organizacion

10.1.2 Punto de Vista de Actor Cooperación

Se enfoca en identificar las relaciones de los actores con el entorno. Aunque otro importante uso es el mostrar como un grupo de actores trabajan juntos para realizar un proceso de negocio. El metamodelo de la vista puede ser apreciado en la Figura 10.3.

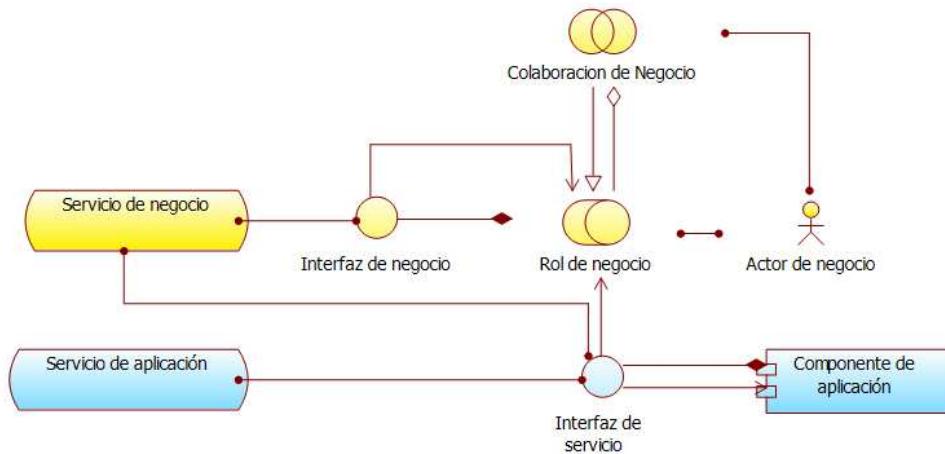


Figure 10.3: Metamodelo Actor cooperación

En la Figura 10.4, se puede apreciar que la colaboración entre roles “despacho de mercancía”, ayuda a soportar el servicio ofrecido al consumidor de “Solicitar pedido”.

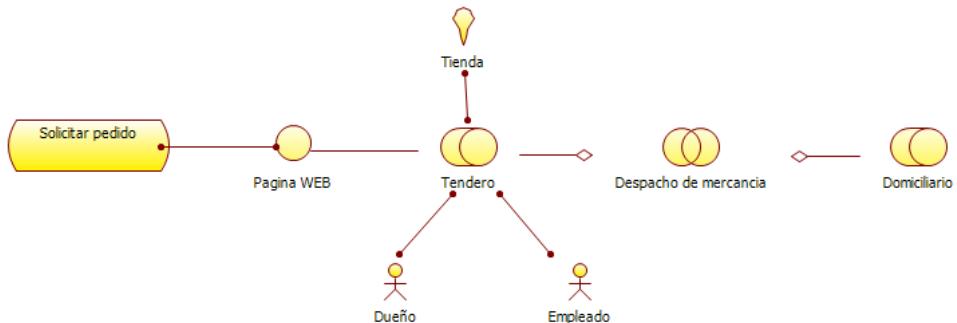


Figure 10.4: Actor Cooperación

10.1.3 Punto de Vista de Función de Negocio

Muestra las funciones principales de negocio para la organización y su relación en términos de flujo de información, valor, o bienes entre ellas. Son usados para representar los aspectos más estables de una compañía en términos de sus actividades primarias, independientemente de sus cambios organizacionales o de tecnología. La Figura 10.5 representa el metamodelo de este punto de vista.

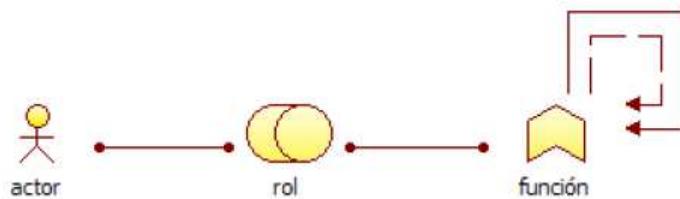


Figure 10.5: Metamodelo Punto de Vista de Función de Negocio

Para el caso de estudio se identifican las funciones principales asociadas al caso de negocio de venta a través de canales virtuales. Estas funciones están siendo soportadas por los roles principales de la empresa, el tendero, el cual se encarga de ofrecer gestión de inventarios, clientes y despachar un pedido, mientras que el domiciliario se encarga de entregar el pedido y recibir el pago de este. El modelo puede ser visto en la Figura 10.6.

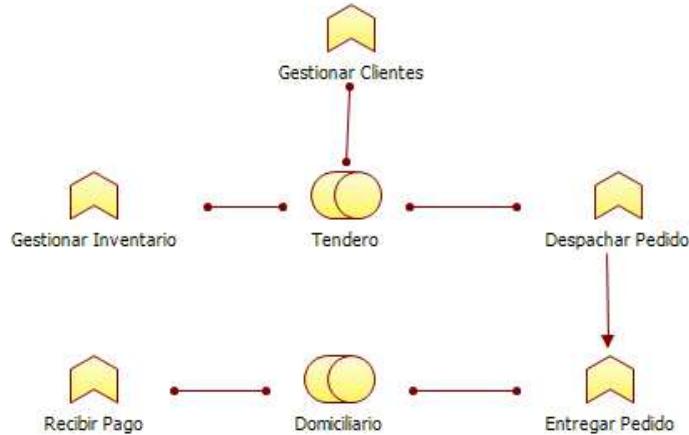


Figure 10.6: Punto De Vista Función de Negocio

10.1.4 Punto de Vista de Proceso de Negocio

Es usado para mostrar la estructura y composición –a alto nivel- de uno o más procesos de negocio. Debe contener otros conceptos como:

- Los servicios que un proceso de negocio ofrece al exterior, mostrando como los procesos contribuyen a la realización de los productos de la compañía.
- Asignación de los procesos de negocio a los roles.
- La información usada por el proceso de negocio.

El metamodelo del punto vista se observa en la Figura 10.7.

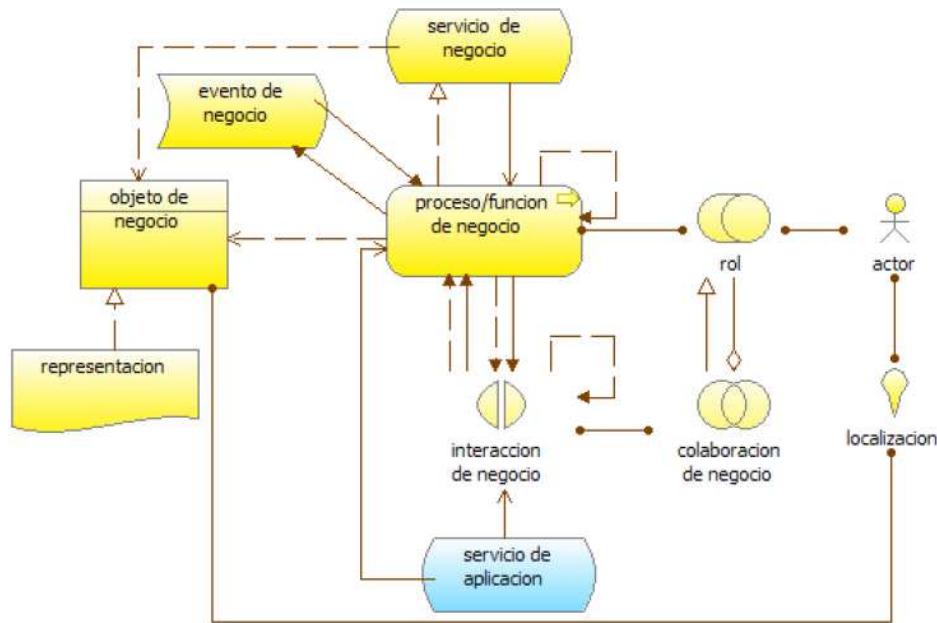


Figure 10.7: Metamodelo Punto de Vista de Proceso de Negocio

En la Figura 10.8, se observa el servicio al cual se enfoca el negocio “Vender productos de calidad sin intermediarios” el cual esta soportado por el proceso de negocio de “Venta de productos”, el cual a su vez se implementa por otros 3 procesos, los cuales son disparados por una solicitud de pedido para producir una venta.

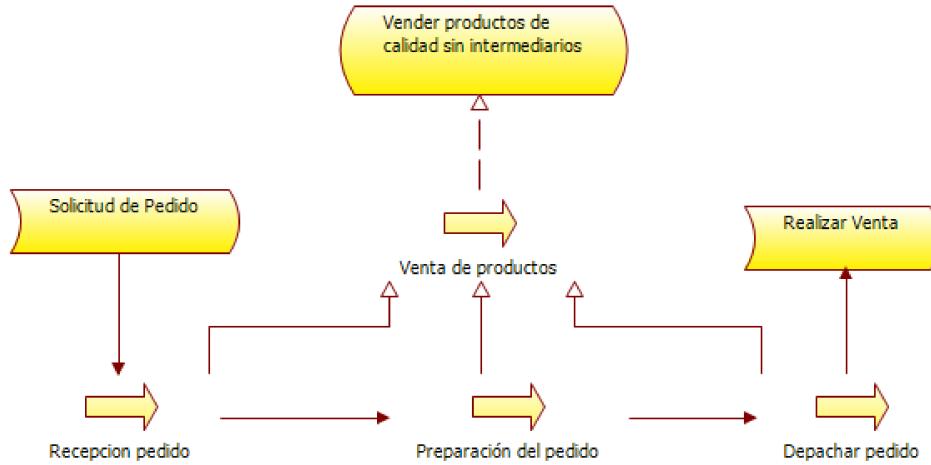


Figure 10.8: Vista de Proceso de Negocio

10.1.5 Punto de Vista de Cooperación de Proceso de Negocio

Es usado para amostrar las relaciones que existen entre los procesos de negocio y con su entorno. Entre los aspectos importantes del punto de vista se encuentran:

- Relación causal entre los procesos principales de negocio de la compañía.
- Mapeo de los procesos de negocio a funciones de negocio.
- Realización de servicios por procesos de negocio.
- El uso de datos compartidos.

El metamodelo puede ser apreciado en la Figura 10.9.

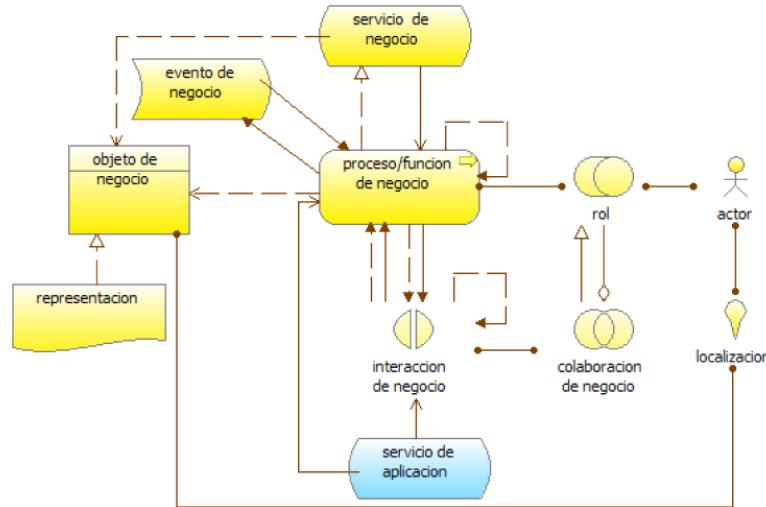


Figure 10.9: Metamodelo Cooperación de Proceso de Negocio

Para el caso de ventas en las tiendas de barrio se aprecia la relación que tienen los roles de “Tendero” y “Domiciliario” con los procesos de “venta de productos” y “Entrega de pedido”. A su vez se identifican los eventos principales que son “Solicitud de pedido”, “Pedido preparado” y finalmente la “Venta de mercancía”. Todo esto enfocado en ofrecer el servicio de “Vender productos de calidad sin intermediarios”. Lo anterior se ve representado en la Figura 10.10.

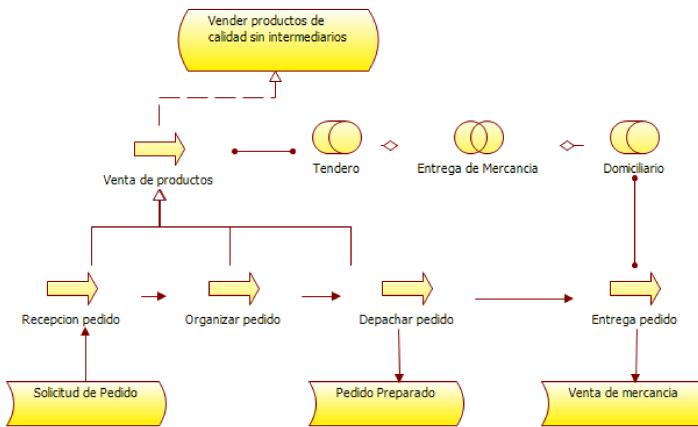


Figure 10.10: Vista de Cooperación de Proceso de Negocio

10.1.6 Punto de Vista de Producto

Representa el valor de ese producto al consumidor o a otras partes involucradas y adicionalmente muestra la composición del producto en términos de servicios, contratos asociados u otros acuerdos. Es típicamente usado para el desarrollo de productos a partir de la composición de servicios existentes o para identificar los servicios que deben ser creados para un producto particular. El metamodelo se aprecia en la Figura 10.11.

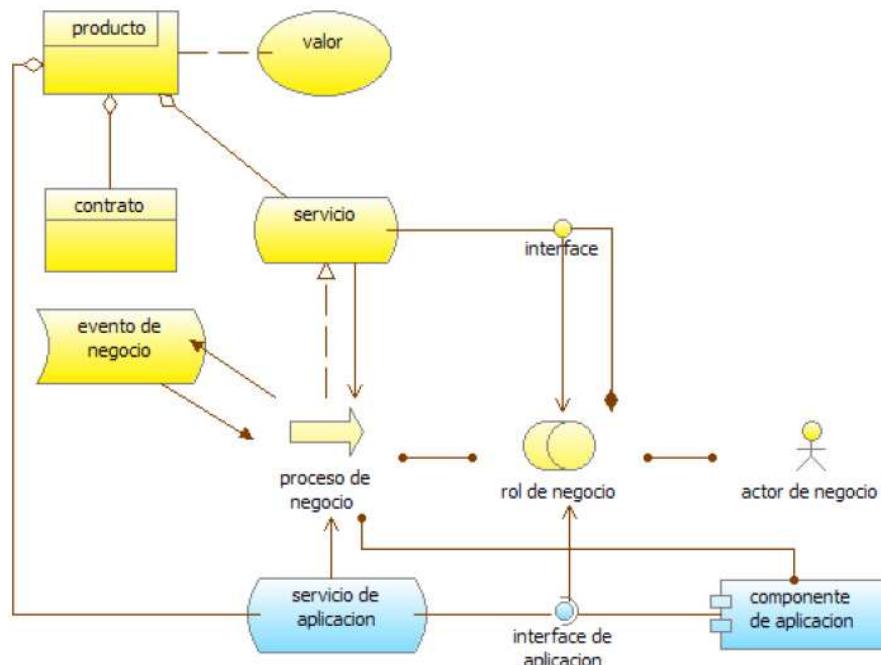


Figure 10.11: Metamodelo Punto de Vista de Producto

La Figura 10.12 muestra que el producto a ofrecer es la página web de la tienda la cual se basa en un contrato de “acceso a la tienda”, y tiene valor para el cliente en mejorar la forma de comunicarme y llevar registro de mis pedidos. A su vez este esta soportado por el servicio de negocio de “Solicitar productos de calidad sin intermediarios”.

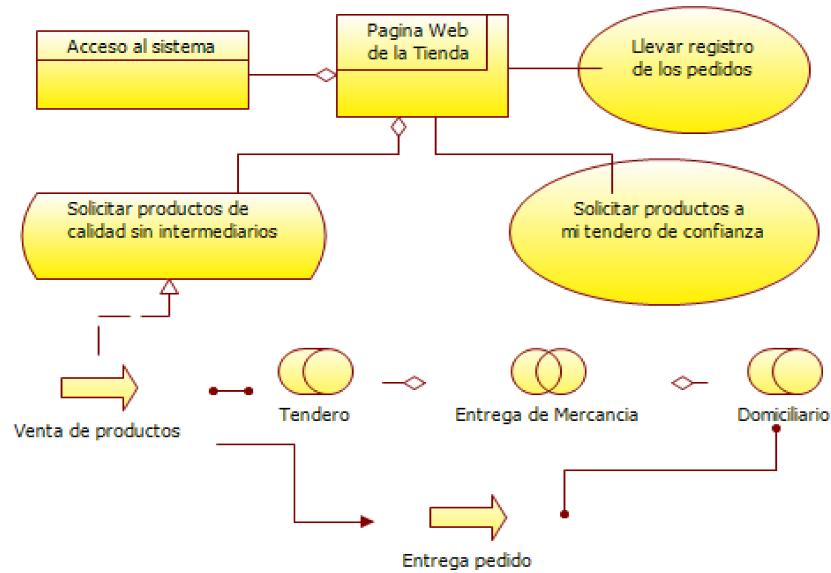


Figure 10.12: Vista de Producto

10.1.7 Punto de Vista de Comportamiento de Aplicación

Describe el comportamiento interno de una aplicación. Este punto de vista es útil en el diseño del comportamiento general de la aplicación o identificando superposición funcional de las diferentes aplicaciones. La Figura 10.13 muestra el metamodelo de del punto de vista.

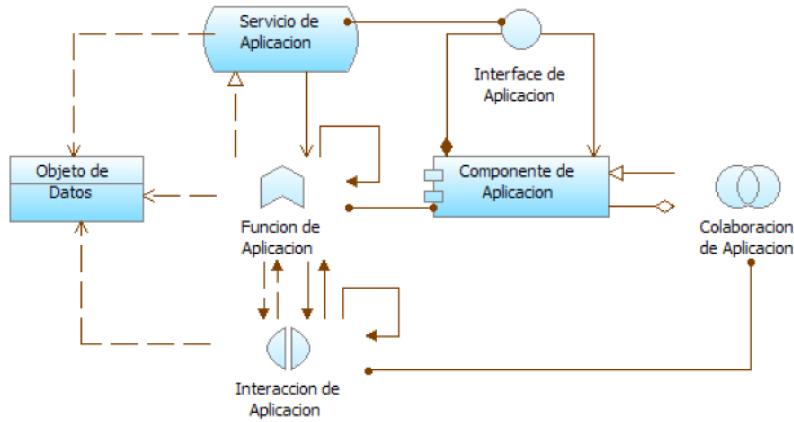


Figure 10.13: Metamodelo Punto de Vista de Comportamiento de Aplicación

En la Figura 10.14 se observan los componentes principales de la aplicación los cuales ofrecen funcionales esenciales para la aplicación, tales como: Control de acceso, realizar pedidos, validar estado de un pedido, manejar el historial entre otros.

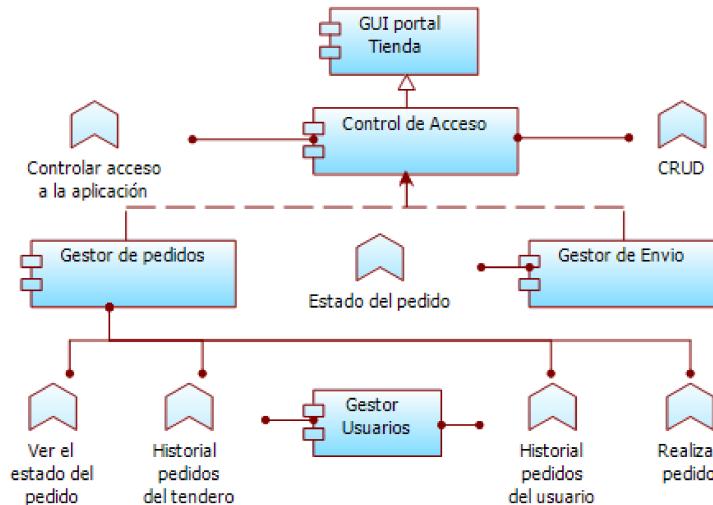


Figure 10.14: Comportamiento de Aplicación

10.1.8 Punto de Vista de Cooperación de Aplicación

Describe las relaciones entre componentes en términos del flujo de información entre ellos. Ayuda a expresar la cooperación o orquestación de servicios que juntos soportan la ejecución de procesos de negocio. La Figura 10.15 muestra el metamodelo del punto del vista.

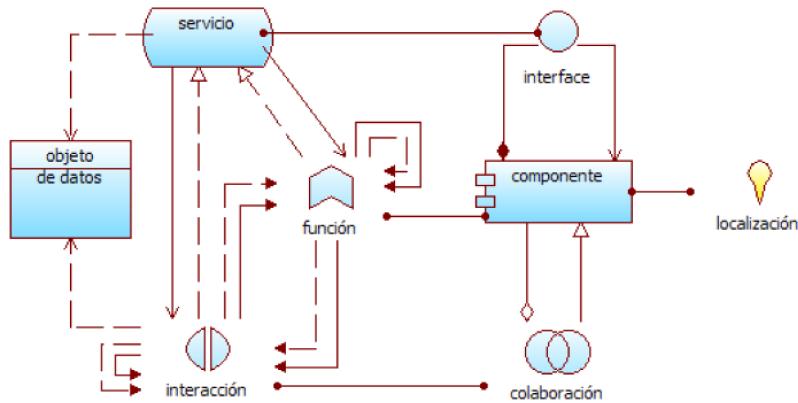


Figure 10.15: Metamodelo Punto de Vista de Cooperación de Aplicación

La Figura 10.16 muestra dos locaciones lógicas un front-end y un back-end, en las que se distribuyen los componentes principales de la aplicación y a su vez muestra la relación entre ellos. En donde se aprecia que la lógica de negocio se ejecuta en el back-end dejando en el front tan solo la presentación.

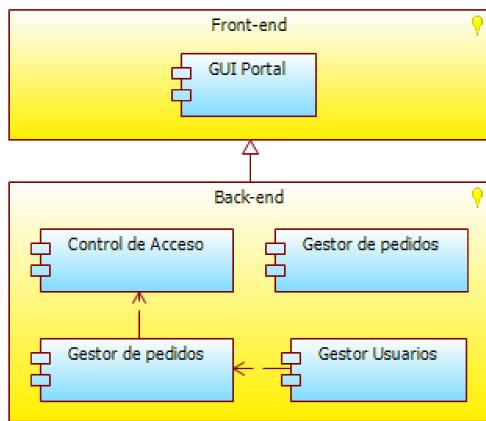


Figure 10.16: Vista de Cooperación de Aplicación

10.1.9 Punto de Vista de Estructura de Aplicación

Muestra la estructura de una o mas aplicaciones o componentes. Es útil en el diseño y entendimiento de la estructura principal de paliación o de los componentes y sus datos asociados. El metamodelo se muestra en la Figura 10.17.

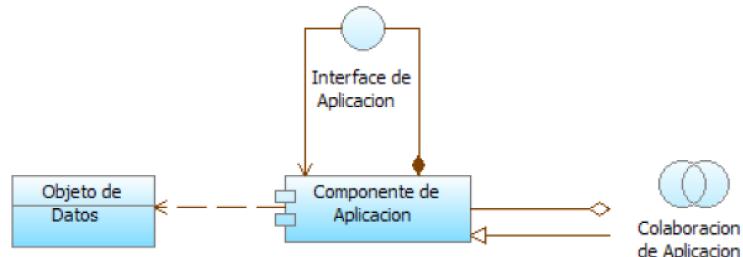


Figure 10.17: Metamodelo Punto de Vista de Estructura de Aplicación

En la Figura 10.18 se muestra la asociación de los componentes principales, como se conectan, con el componente “GUI”, lo cual quiere representar que los componentes son independientes entre ellos.

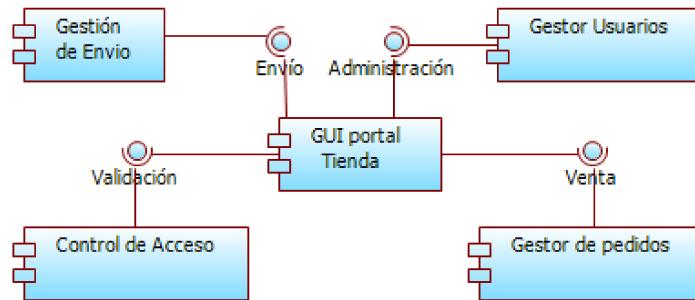


Figure 10.18: Punto de Vista de Estructura de Aplicación

10.1.10 Punto de Vista de Uso de Aplicación

Describe como las aplicaciones son usadas para soportar uno o más procesos de negocio, y como ellos son usadas por otras aplicaciones. Pueden ser usadas en el diseño de la aplicación para identificar servicios necesarios o en diseño

de procesos de negocio para describir los servicios que están disponibles. La Figura 10.19 ilustra el metamodelo del punto de vista.

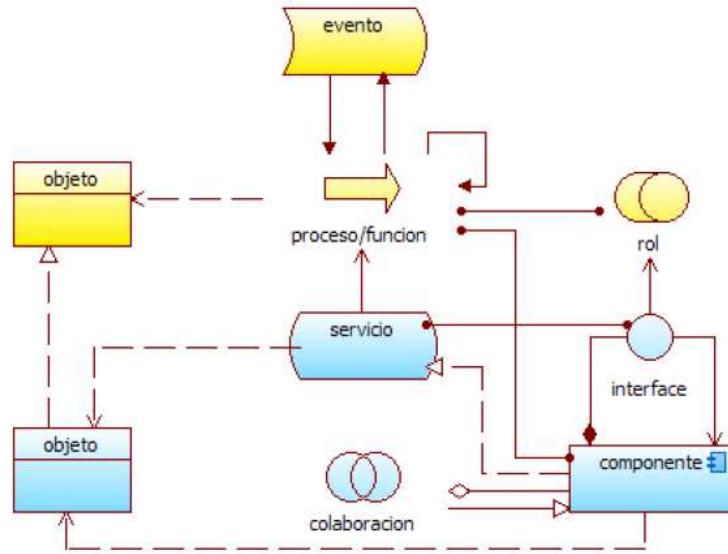


Figure 10.19: Metamodelo Punto de Vista de Uso de Aplicación

En la Figura 10.20 vemos que se ofrecen dos servicios de aplicación a los procesos de negocio, estos son “Gestionar Pedido” que soporta el proceso de “Venta de productos” y el servicio de aplicación de “Gestionar estado del pedido” que es usado en el proceso de entrega del pedido.

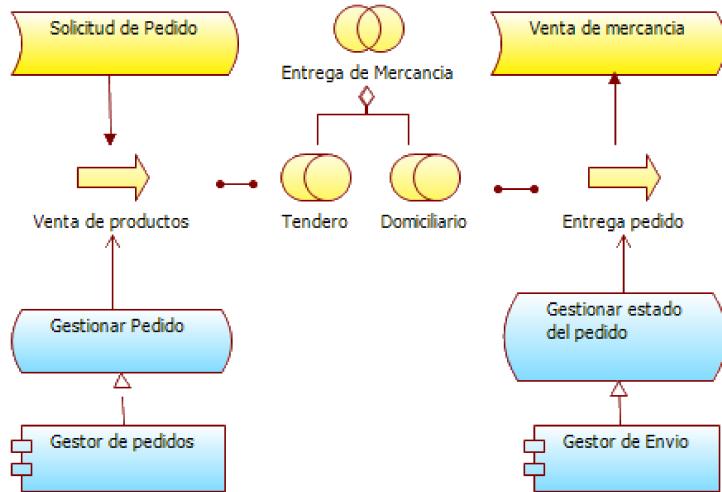


Figure 10.20: Punto de Vista de Uso de Aplicación

10.1.11 Punto de Vista de Infraestructura

El punto de vista de infraestructura contiene los elementos de software y hardware necesarios para soportar la capa de aplicación, así como los dispositivos físicos, redes o sistemas de software. El objeto central del metamodelo es el nodo como se aprecia en la Figura 10.21.

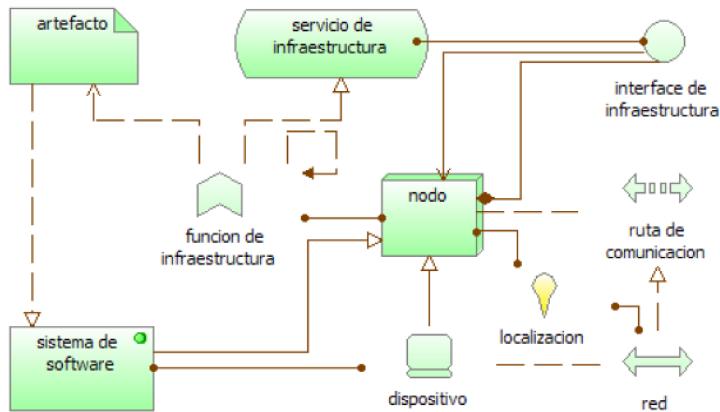


Figure 10.21: Metamodelo Punto de Vista de Infraestructura

En la Figura 10.22 se observa que la capa de infraestructura ofrece 3 servicios “Servicio de despliegue”, “Servicio de incrementar capacidad” los cuales son ofrecidos por que las aplicaciones son desplegadas como micro servicios en un contendor Docker y el “Servicio de red” que permite la conectividad entre los dispositivos y la aplicación.

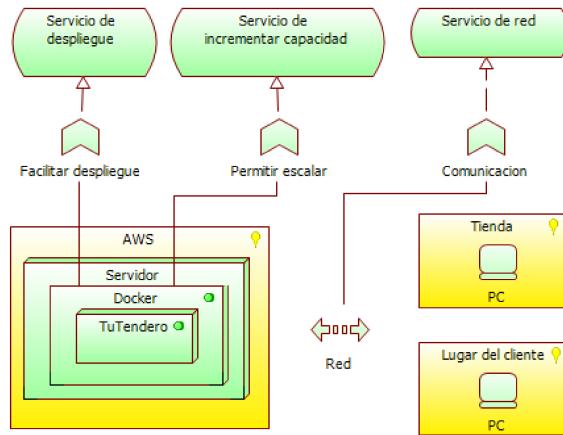


Figure 10.22: Punto de Vista de Infraestructura

10.1.12 Punto de Vista de Uso de Infraestructura

Muestra como las aplicaciones son soportadas por la infraestructura de hardware y software. Este punto de vista juega un papel importante en el análisis del rendimiento y escalabilidad, dado que relaciona la capa física con la lógica. El metamodelo se puede observar en la Figura 10.23.

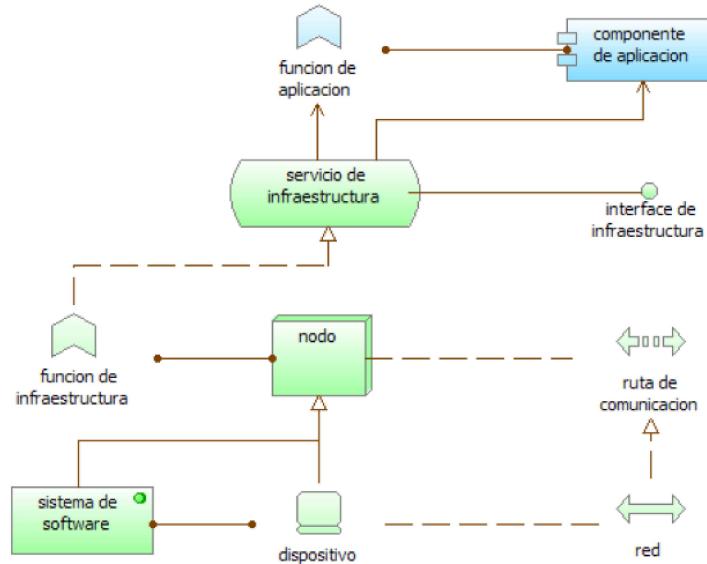


Figure 10.23: Metamodelo Punto de Vista de Uso de Infraestructura

En la Figura 10.24 vemos la distribución de los componentes en los servidores, en los que cada componente está desplegado en un contenedor independiente para facilitar la escalabilidad y el despliegue de cada servicio.

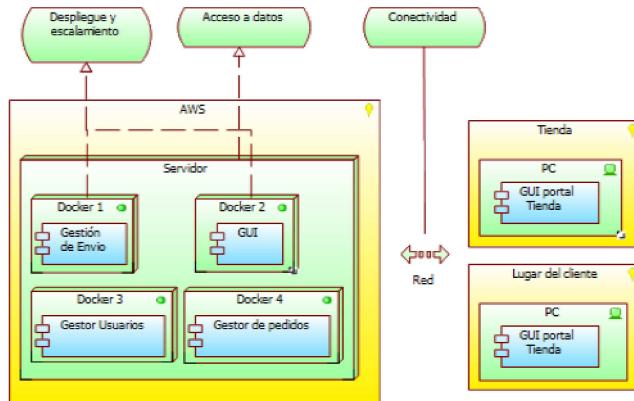


Figure 10.24: Punto de Vista de Uso de Infraestructura

10.1.13 Punto de Vista de Organización e Implementación

Muestra como una o más aplicaciones se implementan sobre la infraestructura. Mapea las aplicaciones lógicas y aplicaciones en artefactos –físicos-, es decir, las tecnologías específicas. Adicionalmente se muestra el manejo de la información (como su almacenamiento en BD). Son usadas para identificar riesgos que se puedan generar. El metamodelo de la Figura 10.25. nos ilustra al respecto.

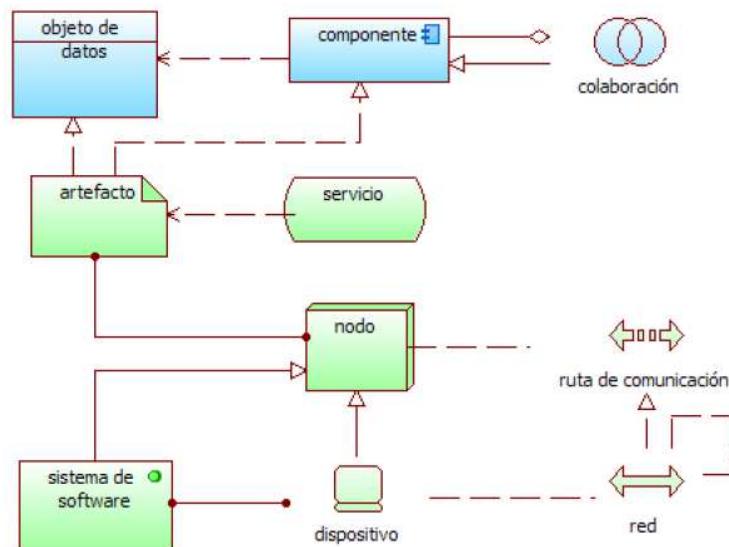


Figure 10.25: Metamodelo Punto de Vista de Organización e Implementación

En la Figura 10.26 se observa la implementación que los componentes esenciales van a tener. Se opta por tener un enfoque basado en micro servicios con lo que cada uno de los componentes estará desplegado en un contenedor Docker. Para los componentes de aplicación tales como gestión de envíos, usuarios y pedidos se implementaran usando Node JS y el Framework Loopback, dada las ventajas que proporciona para micro servicios. Y cada uno de ellos contara con una base de datos propia que estará implementada en Mysql, dada la facilidad de uso de un esquema relacional, aunque esta puede ser modificada a otra, como MongoDB dependiendo de la necesidad del servicio en cuestión. El componente de “Control de Acceso” estará implementado por el API Gateway de Strongloop. Y el componente de presentación está construido en Angular JS.

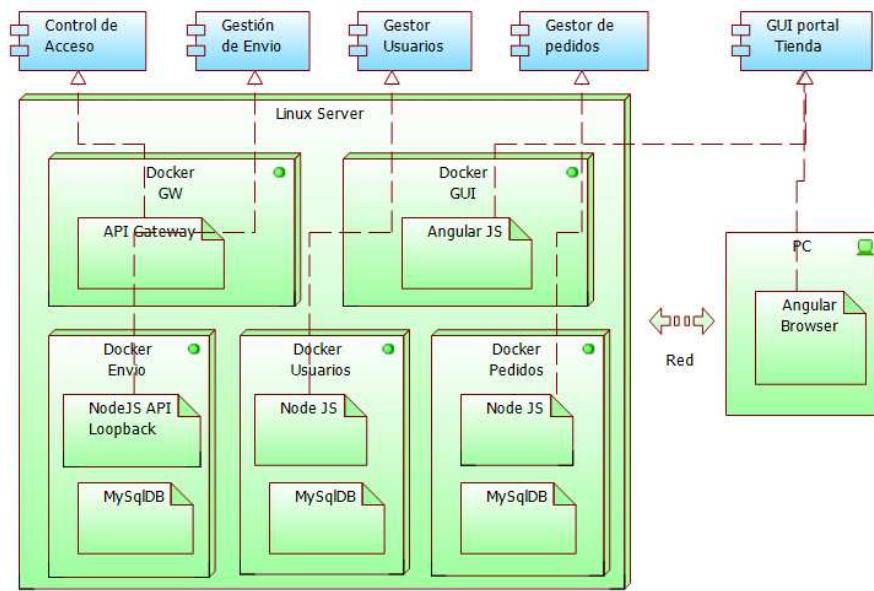


Figure 10.26: Punto de Vista de Organización e Implementación

10.1.14 Punto de Vista de Estructura de Información

El punto de vista de la estructura de información es comparable a los modelos de información tradicionales creados en el desarrollo de sistemas de información. Muestra la estructura de la información usada en la empresa o en el proceso de negocio específico en términos de tipos de datos o estructuras de clase (en orientación a objetos). Mas allá, puede mostrar como la información en el nivel de negocio es representada en la aplicación y como estas son mapeadas a una infraestructura subyacente. El metamodelo se ilustra en Figura 10.27.

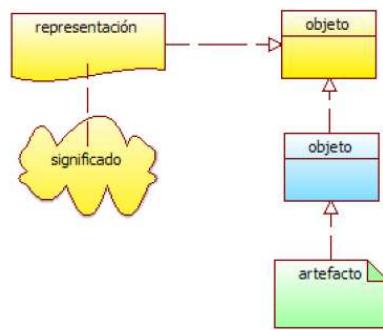


Figure 10.27: Metamodelo Punto de Vista de Estructura de Información

El punto de vista de estructura de información puede ser apreciado en la Figura 10.28. En esta se pueden ver los objetos de negocios relacionados a la venta de productos, estos son Producto, Usuario, Tienda y Pedido. De los cuales se componen los objetos de la aplicación.

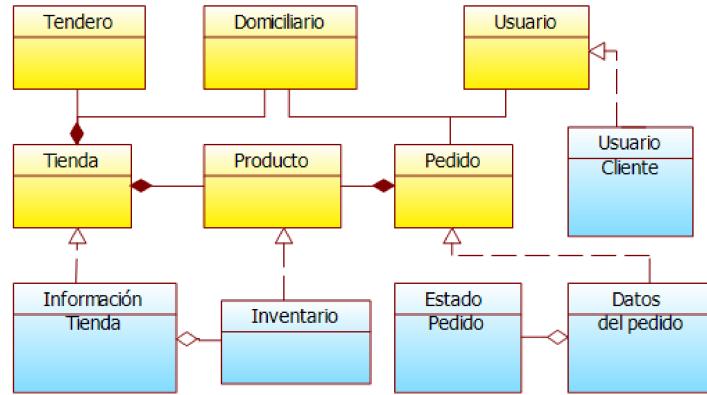


Figure 10.28: Punto de Vista de Estructura de Información

10.1.15 Punto de Vista de Realización del Servicio

Este punto de vista es usado para mostrar como uno o más servicios de negocio son implementados en procesos subyacentes. Forma el puente entre los puntos de vista de productos de negocio y las vistas de procesos de negocio. Provee una vista desde el exterior de uno o más procesos de negocios. La Figura 10.29 muestra el metamodelo del punto de vista.

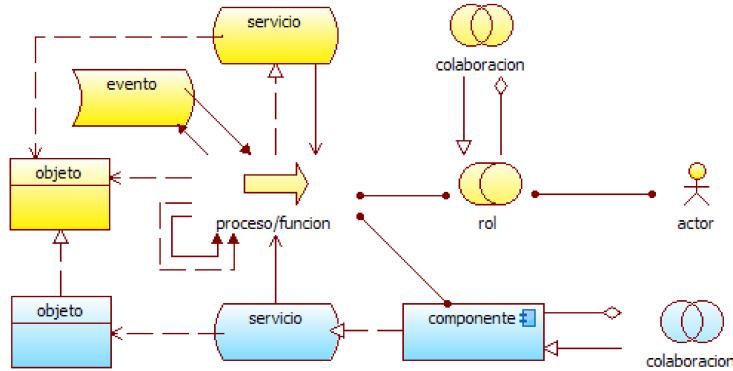


Figure 10.29: Metamodelo Punto de Vista de Realización del Servicio

La Figura 10.30 representa que el servicio de negocio “Solicitar productos de calidad sin intermediarios” es soportado a nivel de aplicación por dos servicios principales que son: El “Gestionar Pedido” y “Gestionar Estado

Pedido” los cuales permiten la creación de pedidos y el seguimiento de los mismos.

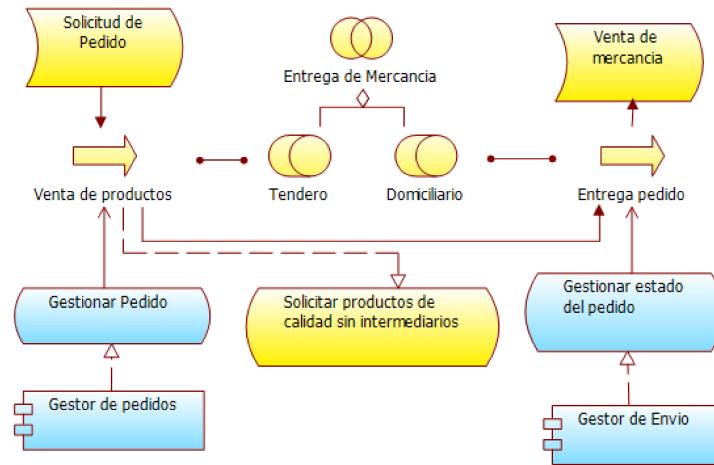


Figure 10.30: Punto de Vista de Realización del Servicio

10.1.16 Punto de Vista de Capas

Permite visualizar varias capas y aspectos de una arquitectura empresarial en un solo diagrama. Existen dos categorías de capas, estas son: capas dedicadas y capas de servicio. Este punto de es el resultado del agrupamiento de relaciones. El objetivo principal es proveer una visión general en un diagrama. Adicionalmente, puede ser usado como soporte del impacto del análisis del cambio y análisis de rendimiento, o para extender el portafolio de servicios.

La Figura 10.31 permite visualizar una vista global de las capas de negocio, aplicación e infraestructura. En esta se pueden observar como las capas inferiores (negocio e infraestructura) trabajan juntas con el fin de ofrecer una página web que permita a los clientes solicitar productos a su tendero de confianza.

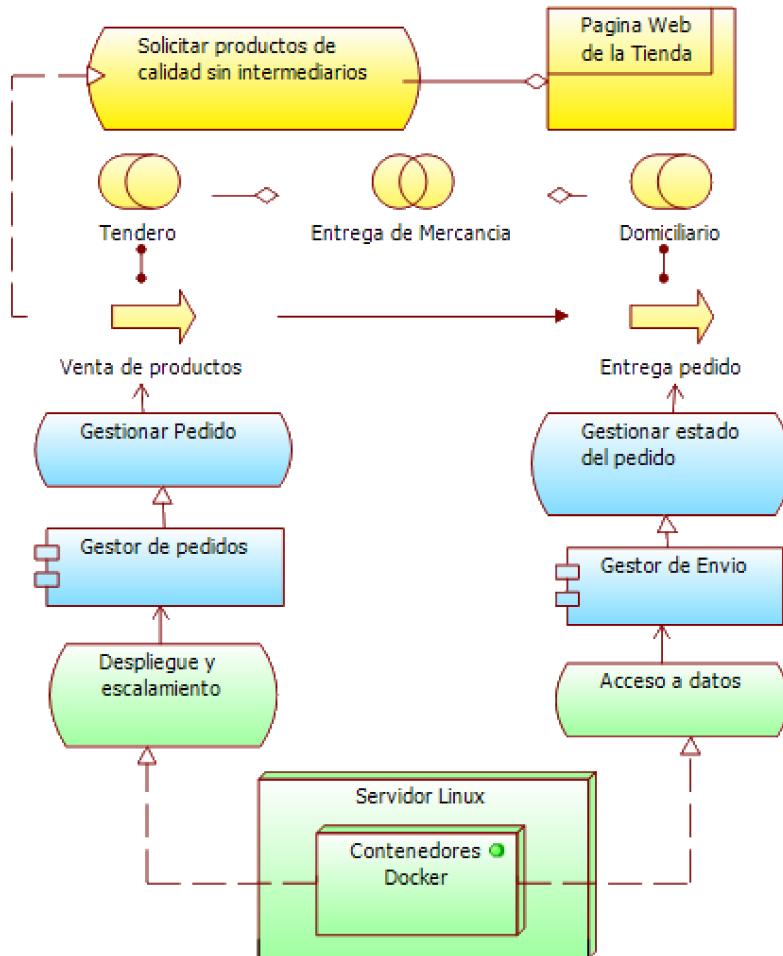


Figure 10.31: Punto de Vista de Capas

10.1.17 Punto de Vista de Interesado

Permite al analista modelar los interesados, manejadores de cambio, internos y externos y las valoraciones (en términos de fortalezas, debilidades, oportunidades y amenazas) de esos manejadores, además del vínculo con los objetivos iniciales. El metamodelo se aprecia en la Figura 10.32

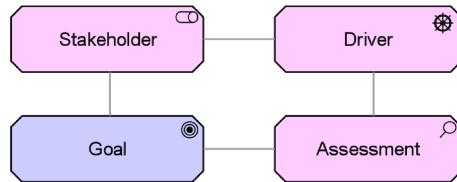


Figure 10.32: Metamodelo Punto de Vista de interesado. [6]

La Figura 10.33 permite apreciar los dos interesados principales, el tendero y el cliente. Ellos tienen objetivos principales como facilitar la adquisición de productos, en el caso del tendero, y poder comprar productos al tendero de confianza, en caso del cliente. Se puede apreciar que los objetivos tienen valoraciones específicas y que ambos comparten el hecho de querer conseguir -y ofrecer- productos sin necesidad de depender de la disponibilidad de línea telefónica.

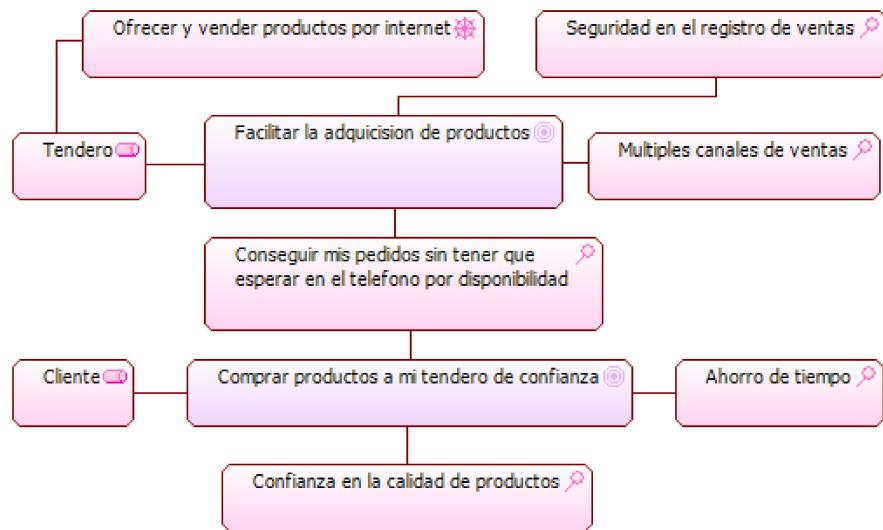


Figure 10.33: Punto de Vista de Interesado

10.1.18 Punto de Vista de Realización de Objetivos

La Figura 10.34 presenta el metamodelo en donde el objetivo es representar el refinamiento de los principales objetivos organizacionales en requerimientos y restricciones.

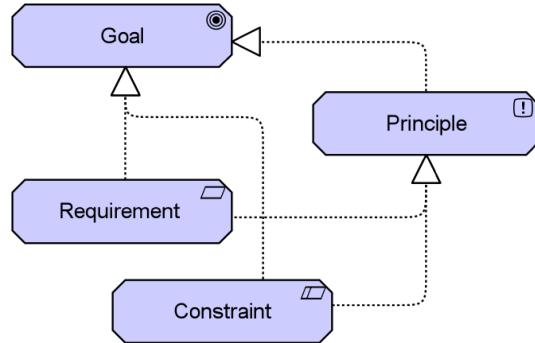


Figure 10.34: Metamodelo Punto de Vista de Realización de Objetivos. [6]

En la Figura 10.35 se pueden ver un poco más a fondo la especificación de los requerimientos que abarca cada uno de los objetivos así como el principio de Disponibilidad que busca describir el desarrollo de los objetivos.

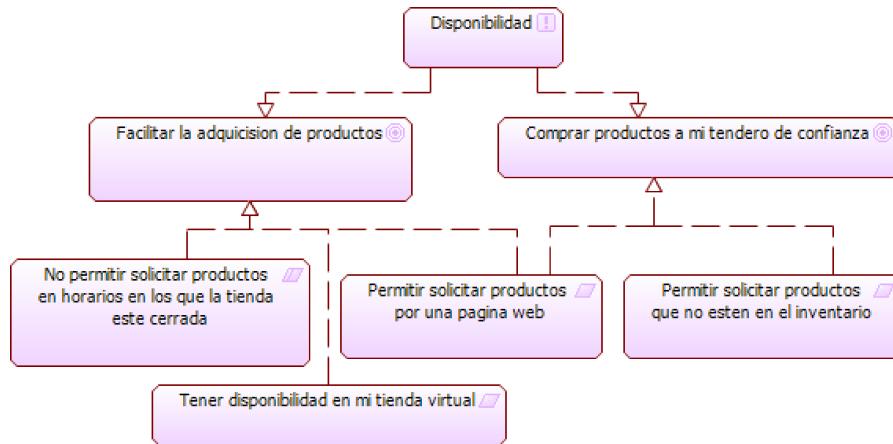


Figure 10.35: Punto de Vista de Realización de Objetivos

10.1.19 Punto de Vista de Contribución de Objetivos

Permite al diseñador o analista modelar las relaciones que influyen entre los objetivos y los requerimientos. Las vistas resultantes pueden utilizarse para analizar el impacto que unos objetivos tienen sobre otros o para detectar conflictos entre los objetivos de las diferentes partes interesadas. La Figura 10.36 permite observar el metamodelo de este punto de vista.

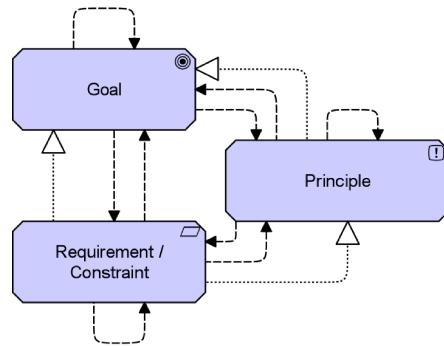


Figure 10.36: Metamodelo Punto de Vista de Contribución de Objetivos. [6]

En la Figura 10.37 se aprecia como requerimientos como “tener disponibilidad en mi tienda virtual” buscan influenciar positivamente el principio de “Disponibilidad” o como este principio influencia positivamente nuestro objetivo de “Confianza al cliente”. Sin embargo también se puede apreciar que otros requerimientos también pueden influenciar negativamente algún otro objetivo, por ejemplo el caso del requerimiento “Permitir solicitar productos por una página web” afecta al objetivo de interactuar con el cliente pues se reduce el trato persona a persona.

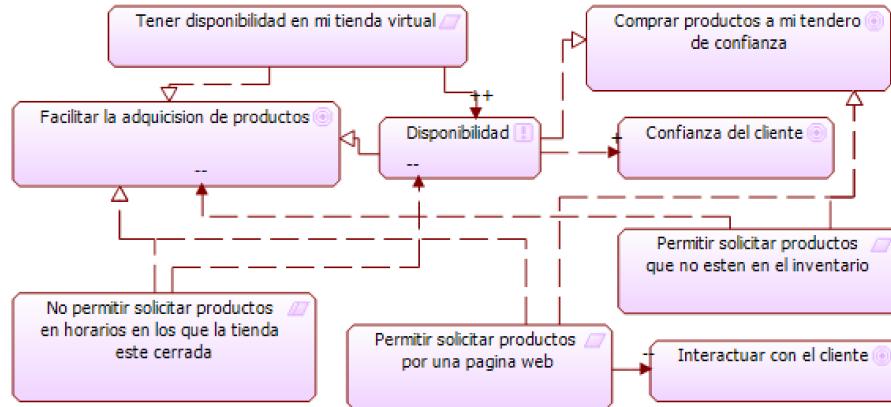


Figure 10.37: Punto de Vista de Contribución de Objetivos

10.1.20 Punto de Vista de Principios

El punto de vista de principios permite al analista o diseñador modelar los principios que son relevantes para el problema. Adicionalmente, las relaciones entre los principios y sus objetivos, pueden ser modelados. El metamodelo puede ser apreciado en la Figura 10.38.

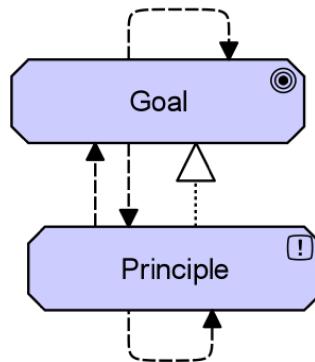


Figure 10.38: Metamodelo Punto de Vista de Principios. [6]

En la Figura 10.39 podemos observar otros de nuestros principios que realizan nuestros objetivos principales. En esta se aprecian lo importantes que son los principios de disponibilidad, confianza y orientación al usuario.

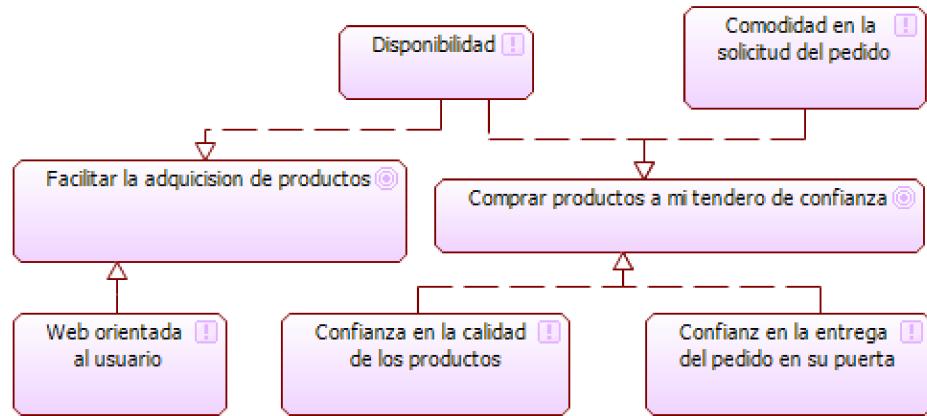


Figure 10.39: Punto de Vista de Principios

10.1.21 Punto de Vista de Realización de Requerimientos

Permite al diseñador modelar la realización de los requerimientos por los elementos base, tales como actores de negocio, servicios de negocio, procesos de negocio, servicios de aplicación, etc. El metamodelo se observa en la Figura 10.40.

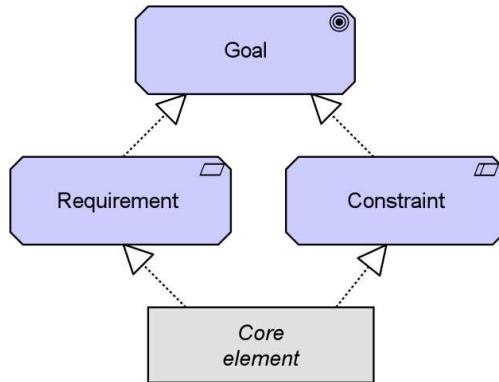


Figure 10.40: Metamodelo Punto de Vista de Realización de Requerimientos.
[6]

La Figura 10.41 permite apreciar no solo los requerimientos y restricciones relevantes para cada objetivo de negocio, sino que, por ejemplo el requerimiento de “Permitir solicitar productos por una página web” está directamente dirigido al servicio de negocio de “solicitar productos de calidad sin intermediarios”.

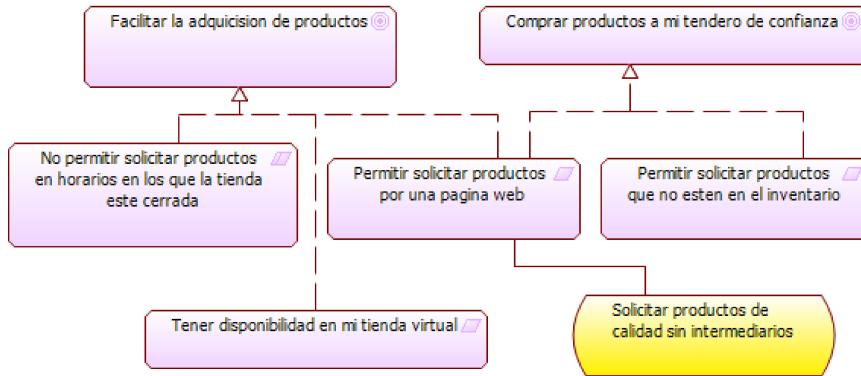


Figure 10.41: Punto de Vista de Realización de Requerimientos

10.1.22 Punto de Vista de Motivación

Permite modelar el aspecto de motivación, sin centrarse en ciertos elementos dentro de este aspecto. Por ejemplo, puede utilizarse para presentar una visión completa o parcial del aspecto de motivación relacionando interesados, sus principales objetivos, los principios que se aplican, y los principales requerimientos de servicio, proceso, aplicaciones y objetos. La Figura 10.42 permite observar el metamodelo de este punto de vista.

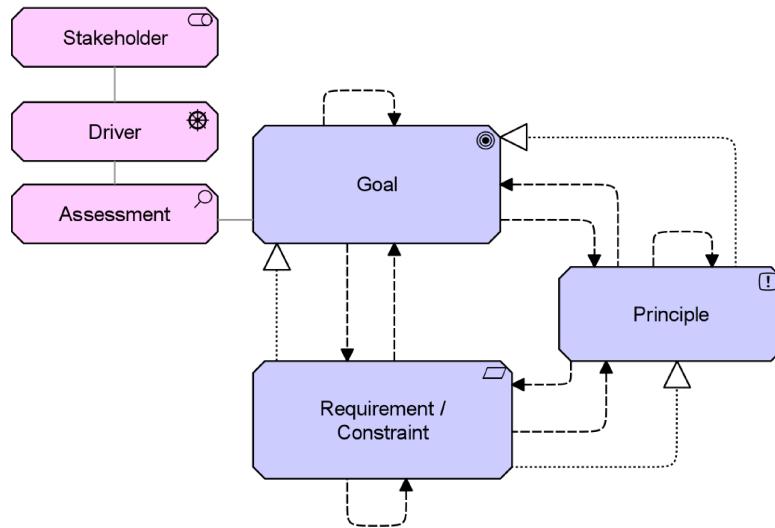


Figure 10.42: Metamodelo Punto de Vista de Motivación. [6]

La Figura 10.43 nos brinda una mayor perspectiva de como nuestros interesados (tendero y cliente), sus manejadores y sus valoraciones se relacionan con los objetivos definidos. Por ejemplo, la valoración de “Conseguir mis pedidos sin tener que esperar en el teléfono por disponibilidad” se relaciona con los dos objetivos “Facilitar la adquisición de productos” y “Comprar productos a mi tendero de confianza”.

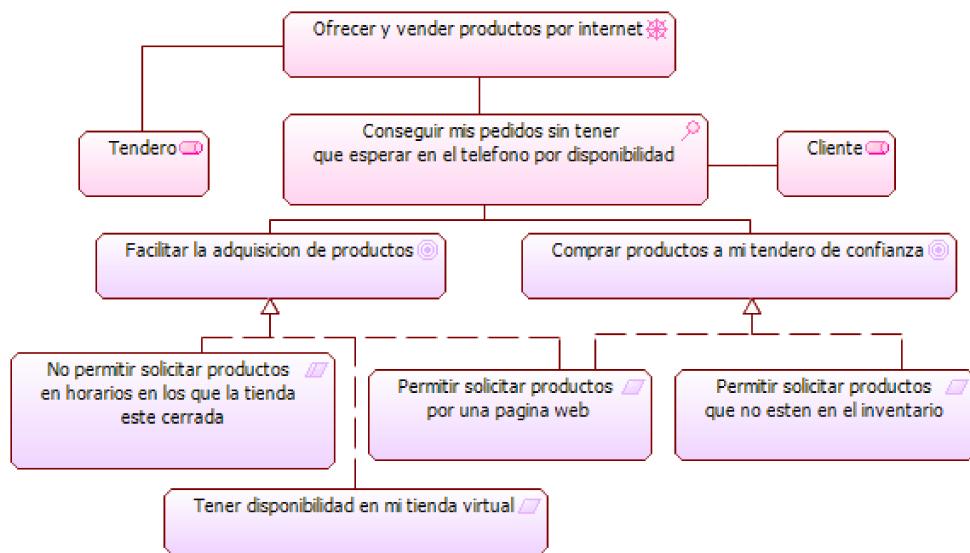


Figure 10.43: Punto de Vista de Motivación

10.1.23 Punto de Vista de Proyecto

Un punto de vista del proyecto se utiliza principalmente para modelar la gestión del cambio de arquitectura. La “arquitectura” del proceso de migración, de una situación vieja (estado actual de la arquitectura empresarial) a una nueva situación deseada (estado deseado de la arquitectura empresarial) tiene consecuencias importantes sobre la estrategia de crecimiento de mediano y largo plazo y el subsecuente proceso de toma de decisiones. El metamodelo de esta vista puede ser observado en la Figura 10.44.

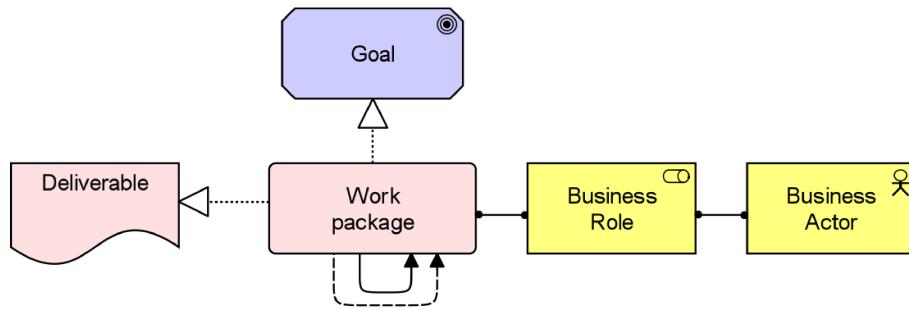


Figure 10.44: Metamodelo Punto de Vista de Proyecto. [6]

En la Figura 10.45 se puede observar nuestro rol de “Tendero” que es quien va a interactuar directamente con el paquete de trabajo “Proyecto la Tienda”, el cual a su vez se compone de un liberable específico que es la “Tienda Virtual” que está realizada por el paquete de trabajo “Pagina Web de la Tienda” el cual cumple el objetivo de “facilitar la adquisición de productos”.

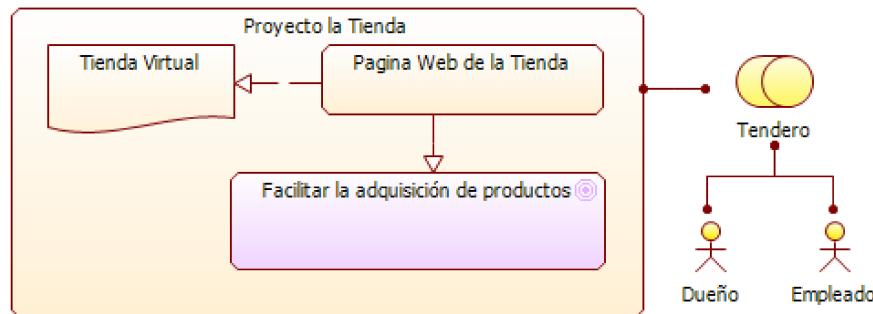


Figure 10.45: Punto de Vista de Proyecto. [6]

10.1.24 Punto de Vista de Migración

El punto de vista de migración implica modelos y conceptos que pueden ser utilizados para especificar la transición de una arquitectura existente a una arquitectura deseada. El metamodelo se observa en la Figura 10.46.

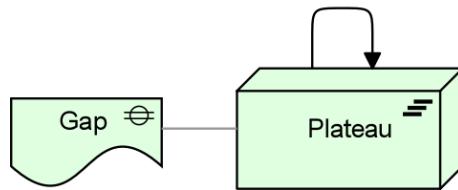


Figure 10.46: Metamodelo Punto de Vista de Migración. [6]

La Figura 10.47 permite observar el progreso que se espera para nuestro proyecto. Este está dado por 3 grandes mesetas las cuales se irán construyendo conforme madure el producto. Adicionalmente se puede observar las brechas que se desean solucionar con el fin de lograr nuestra meseta final “Tienda Virtual Con pagos en línea”.



Figure 10.47: Metamodelo Punto de Vista de Proyecto. [6]

10.1.25 Punto de Vista de Implementación y Migración

Es usado para relacionar programas y proyectos al a las partes de la arquitectura que ellos representan. Esta vista permite modelar el alcance de programas, proyectos, actividades en términos de las mesetas que son realizadas o los elementos individuales de arquitectura que son afectados. La Figura 10.48 ilustra el metamodelo de esta vista.

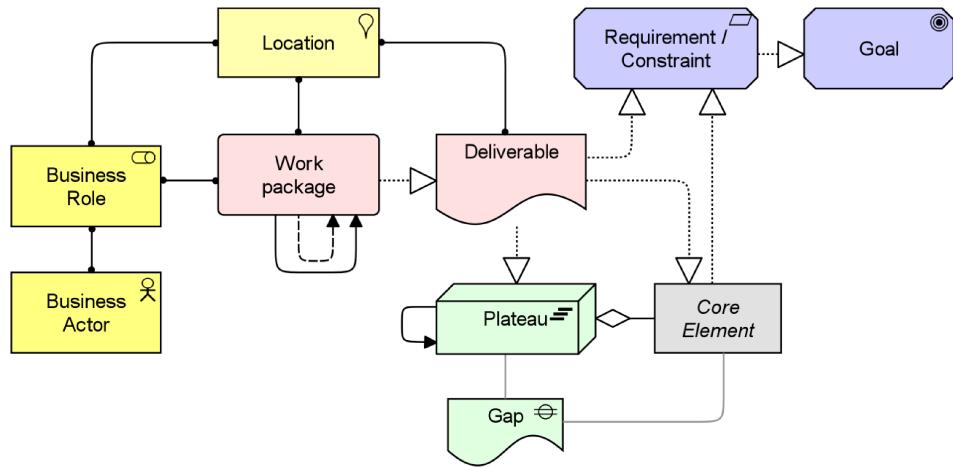


Figure 10.48: Metamodelo Punto de Vista de Implementación y Migración. [6]

En la Figura 10.49 podemos apreciar como nuestras mesetas se relacionan con los liberables, así como su representación en la capa de negocio. Podemos ver que nuestra ubicación virtual “Pagina Web” se representa en el liberable “Tienda Virtual” el cual a su vez es la realización de la meseta actual de “Tienda Virtual Web”

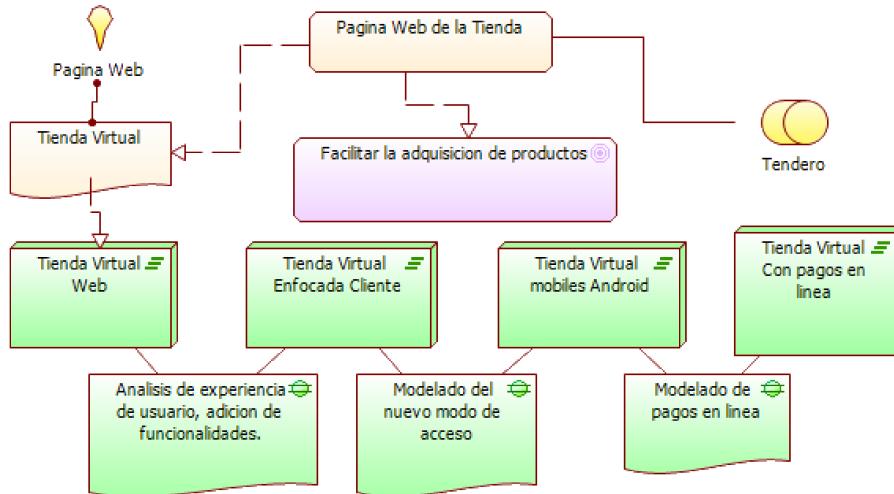


Figure 10.49: Punto de Vista de Implementación y Migración. [6]

Capítulo 11

Patrones de diseño

En el presente capítulo se busca explorar la relación que existe entre los patrones de diseño convencionales GoF y su relación con la implementación realizada en AngularJS para el prototipo del presente proyecto.[4]

11.1 Singleton

Es un patrón de diseño que restringe la instanciación de una clase a un objeto único.

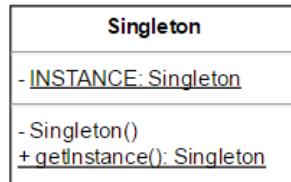


Figure 11.1: Patrón Singleton

El patrón es implementado directamente por angularJS en el uso de servicios, dado que cada servicio es instanciado una única vez. Sin embargo existe una pequeña variación del diagrama UML Figura 11.1, porque en vez de mantener una referencia estática dentro del constructor, esta es almacenada en el cache (el cual puede verse como un administrador singleton). El Algoritmo 11.1 muestra un ejemplo de la implementación de uno de los servicios usados en el prototipo.

Algoritmo 11.1 Implementación Singleton

```
service('DeliveryStatus', function($http) {
  var delivery = this,
    path = 'OrderStatuses/';
})
```

11.2 Facade

El patrón facade puede resumirse como el encargado de proveer una interfaz simplificada de un subsistema complejo. La Figura 11.2 muestra el modelo del mismo.

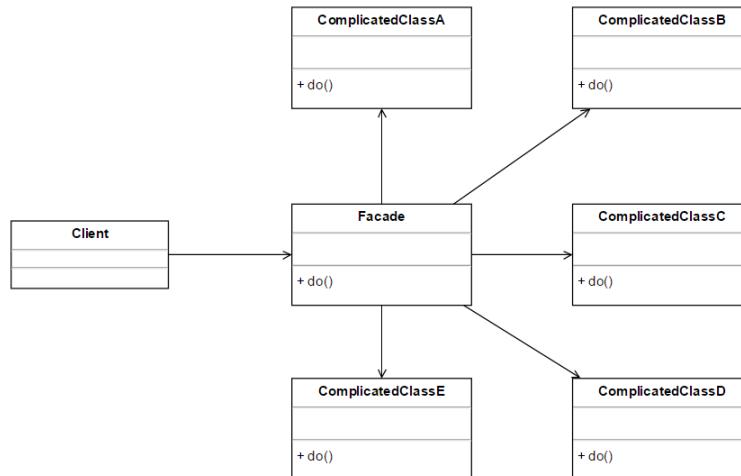


Figure 11.2: Patrón Facade

AngularJS cuenta con algunas fachadas implementadas. Para el prototipo se hizo uso del modulo `$http`, el cual provee una interfaz simple para crear peticiones HTTP. En el Algoritmo 11.2 se puede observar un ejemplo de su uso .

Algoritmo 11.2 Implementación Facade

```
service.createOrder = function(order) {
  return $http.post(apiOrder, order);
};
```

11.3 Intercepting filter

Aunque no es un patrón GoF, este patrón es de mucha utilidad para crear cadenas de filtros compuestos para implementar tareas de pre y post procesamiento común durante peticiones web. La Figura 11.1 ilustra este patrón.

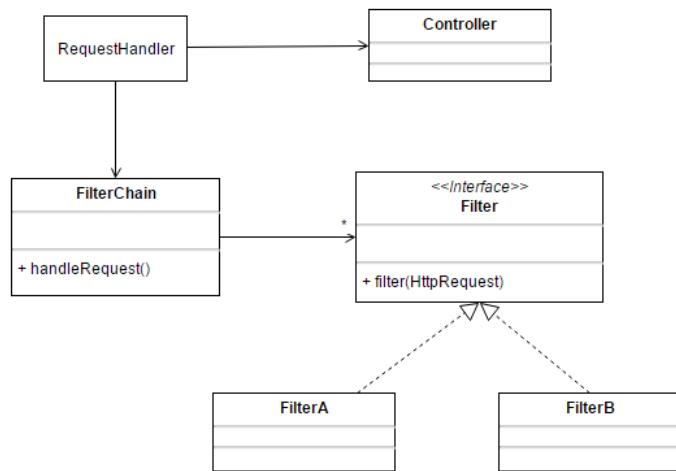


Table 11.1: Patrón Intercepting filter

En angularJS se tiene la idea de filtros interceptores en el módulo \$httpProvider. \$httpProvider tiene una propiedad llamada interceptor la cual permite implementar esta funcionalidad. Para el prototipo se usó para validar que el usuario tuviera una sesión activa. Su implementación puede ser apreciada en el Algoritmo 11.3.

Algoritmo 11.3 Implementación Intercepting filter

```
$httpProvider.interceptors.push('APIInterceptor');

appServices.service('APIInterceptor', function($rootScope,
    UserService) {
    var service = this;
    service.request = function(config) {
        var currentUser = UserService
            .getCurrentUser(),
            accessToken = currentUser ?
                currentUser.access_token : null;
        if (accessToken) {
            config.headers.authorization
                = accessToken;
        }
        return config;
    };

    service.responseError = function(response) {
        if (response.status === 401) {
            $rootScope.$broadcast('unauthorized');
        }
        return response;
    };
})
```

11.4 Interpreter

El patrón de diseño interprete (Figura 11.3) especifica como evaluar sentencias en un leguaje.

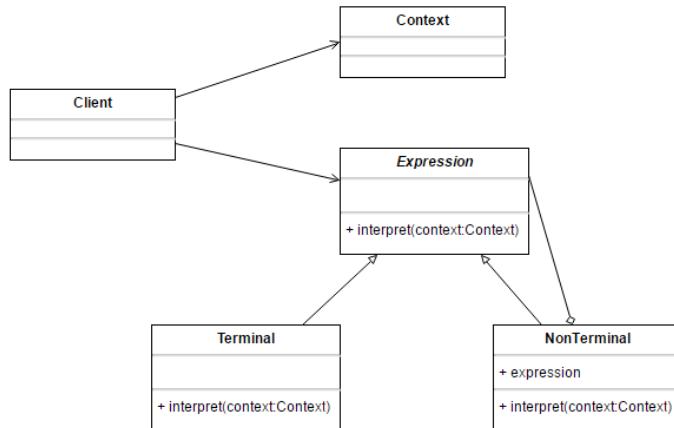


Figure 11.3: Patrón Interpreter

AngularJS provee una implementación propia de un DSL (Domain Specific Language) el cual es usado para dar formato a las fechas como se ve a continuación en el Algoritmo 11.4.

Algoritmo 11.4 Implementación Interpreter

```

<div class="form-group">
  <label>Fecha
  {{ dashboard.editedItem.dateIn | date : 'yyyy-MM-dd HH:mm:ss' }}
</label>
</div>
  
```

11.5 Observer

El patrón observer (Figura 11.4) permite relacionar diferentes objetos entre sí en torno a uno principal, así cada vez que este cambie su estado, los demás lo sabrán.

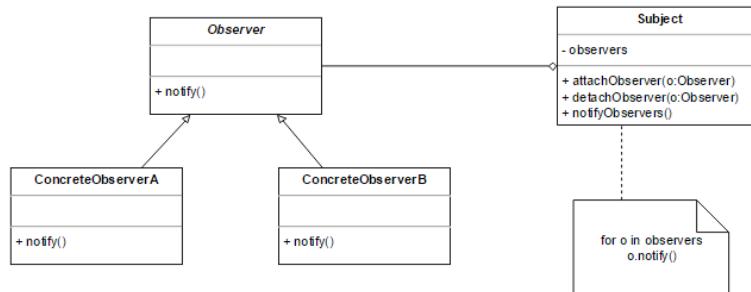


Figure 11.4: Patrón Observer

AngularJS provee una implementacion del patrón observador, para esto provee los metodos `$on`, `$emit` y `$broadcast`. Para el prototipo fue especialmente útil para que en caso de que el usuario no estuviese autorizado, este fuera enviado a un punto en particular de la aplicación, como puede ser observado en el Algoritmo 11.5.

Algoritmo 11.5 Implementación Observer

```

$rootScope.$on( 'unauthorized' , function() {
    main.currentUser = UserService
        .setCurrentUser( null );
    $state.go( 'welcome' );
});
  
```

Capítulo 12

Modelado de la Solución

El presente capítulo se buscará extender y/o especificar a más bajo nivel los modelos planteados - en ArchiMate- en el capítulo de “Arquitectura Empresarial”. Para ello se usarán diagramas UML 2.0 que reflejen la especificación de la solución.

Se ha tomado como objetivo plantear una arquitectura basada en microservicios, por lo que a continuación se describirán por separado cada uno de ellos.

Para el modelado, se ha partido del hecho de que cada uno de ellos es independiente por lo que se aplicó una primera aproximación de los principios aplicados en el desarrollo dirigido por modelos (MDD). Para esto se generó un modelo de los conceptos esenciales del microservicio y luego automáticamente se realiza una transformación para exponer estos mediante interfaces REST (Representational State Transfer). Adicionalmente, se intentó restringir la cantidad de información que es representada en el modelo con el fin de mantener este, lo más simple posible (con el fin de apegarse al principio KISS).

12.1 Definición de la tecnología y herramientas

12.1.1 Back-end

Para el desarrollo de los microservicios se ha escogido utilizar nodeJS, el cual ha tenido un crecimiento y aceptación en los últimos años, convirtiéndose en un lenguaje en auge que cuenta con un gran apoyo por parte de la comunidad que lo soporta. Adicionalmente se desea explorar sus ventajas, tales como su enfoque a la escalabilidad y a eventos no bloqueantes (en este proyecto no se pretende ahondar en las ventajas o desventajas del lenguaje,

simplemente se intenta tener un primer acercamiento con este, e intentar aprovechar sus ventajas). Después de una revisión inicial de los principales Frameworks y librerías que se utilizan, se ha seleccionado usar Loopback como Framework dado el soporte ofrecido por Strongloop¹, y su posibilidad de desarrollo dirigido por modelos.

12.1.2 Front-end

Para el desarrollo de la parte grafica se utilizará AngularJS, dado su naturaleza JavaScript y su facilidad de integrarse con servicios REST en el back-end.

12.1.3 Plataforma

Se seleccionaran servidores Linux para el despliegue de la aplicación, dado el conocimiento y soporte que se tiene sobre estos. Estos servidores se encontrarán en la nube, con el fin de poder escalar fácilmente, y sobre todo bajar los costos, de no contar con una infraestructura física propietaria.

12.1.4 Despliegue

Con el fin de facilitar el despliegue y permitir un crecimiento horizontal de manera simple y rápida (incluso en el mismo servidor) se ha optado por utilizar Docker² como el contenedor de aplicaciones. Cada uno de los microservicios será contenido en una imagen de Docker, la que permitirá iniciar otro contendor completamente funcional tan solo utilizando la consola del mismo. Este nos facilitará el escalar de aplicación así como el manejo de versiones pero dificultara el problema de monitoreo, que puede ser atenuado con el uso de diversas herramientas que cumplen este propósito, pero que se salen del objetivo del prototipo de la solución.

12.1.5 Almacenamiento de datos.

Aunque cada microservicios puede tener su base de datos en el motor que mejor cumpla sus necesidades, se ha decidido seleccionar para todos los microservicios bases de datos MySQL, con el fin de minimizar la curva de aprendizaje sobre otras. Es importante denotar que cada base de datos es independiente.

¹Para más información <https://strongloop.com>

²Sitio oficial: <https://www.docker.com>

12.2 Definición de Microservicios

12.2.1 Gestión de órdenes.

En la Figura 12.1 se puede observar el modelo que representa la gestión de órdenes.

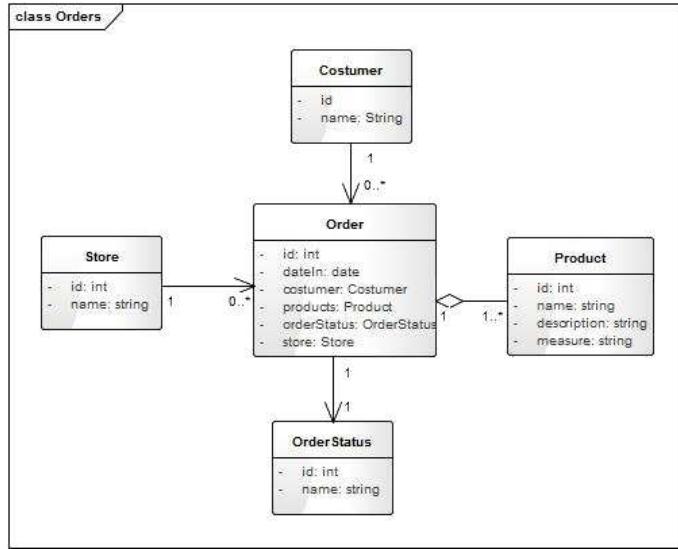


Figure 12.1: Modelo de Gestión de órdenes

En este diagrama se aprecian las siguientes entidades:

- **Costumer**: Referencia al consumidor de los productos. La persona que solicita un producto.
- **Store**: Tienda.
- **OrderStatus**: Estado de la orden, si es aceptada, rechazada en proceso, etc.
- **Product**: El producto de que se ofrece para la venta.
- **Order**: Hace referencia a la solicitud de pedido, está constituida por muchos productos, así como la pertenencia a un cliente.

En la Figura 12.2 muestra el despliegue que tiene el servicio.

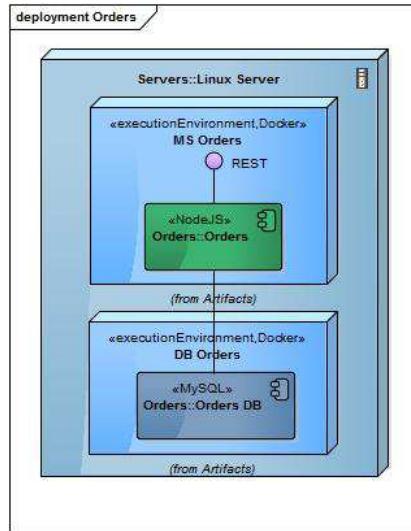


Figure 12.2: Despliegue Órdenes

12.2.2 Estado del pedido

Este servicio será encargado de gestionar el estado actual, responsable de su entrega y ubicación del pedido. Su evolución puede llegar a la ubicación en tiempo real del pedido. La Figura 12.3 permite observar el modelado de la información que contara el servicio en esta iteración.

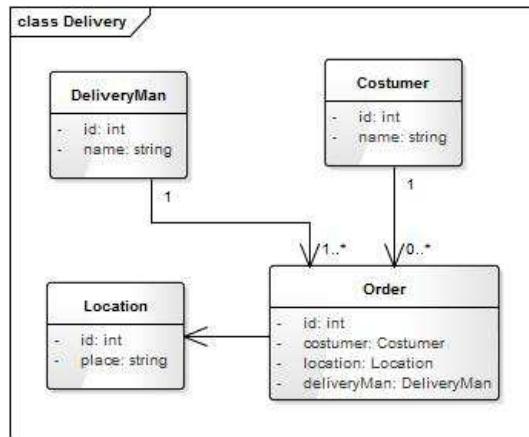


Figure 12.3: Modelo del estado del pedido.

Las entidades contenidas en el diagrama hacen referencia a:

- **Order:** Hace referencia al pedido, está compuesta por una ubicación, el encargado de la entrega y el dueño del pedido -cliente-.
- **Costumer:** Referencia al consumidor de los productos. La persona que solicita un producto.
- **DeliverMan:** Es el encargado de llevar el pedido al cliente.
- **Location:** Ubicación del pedido.

En la Figura 12.4 muestra el despliegue que tiene el servicio.

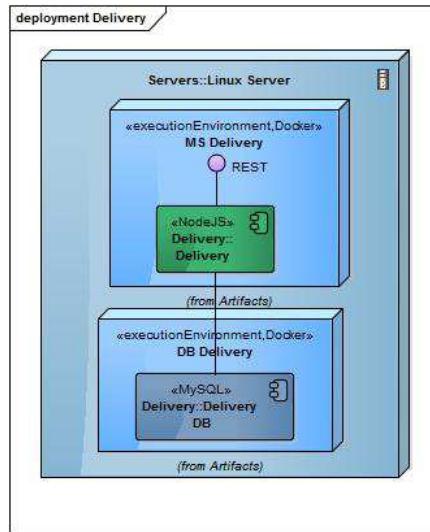


Figure 12.4: Despliegue Órdenes

12.2.3 Gestión de Usuarios

Loopback nos proporciona un modelo de control de usuarios que proporciona ciertas funcionalidades relacionadas con el control de estos y que puede ser extendido para las necesidades específicas del producto. Así, que tomando como base en este modelo se adicionan las entidades relacionadas con los usuarios involucrados en la solución. La Figura 12.5 refleja la extensión hecha al modelo, en donde:

- **User:** Entidad base proporcionada en Loopback, que representa un usuario del sistema.

- **Person:** Hace referencia a una persona del modelo de negocio, esta adiciona atributos adicionales necesarios para el producto.
- **DeliveryMan:** Es la persona encargada de llevar el pedido al cliente.
- **ShopKeeper:** Es el encargado de recibir y organizar el pedido (Tendero).

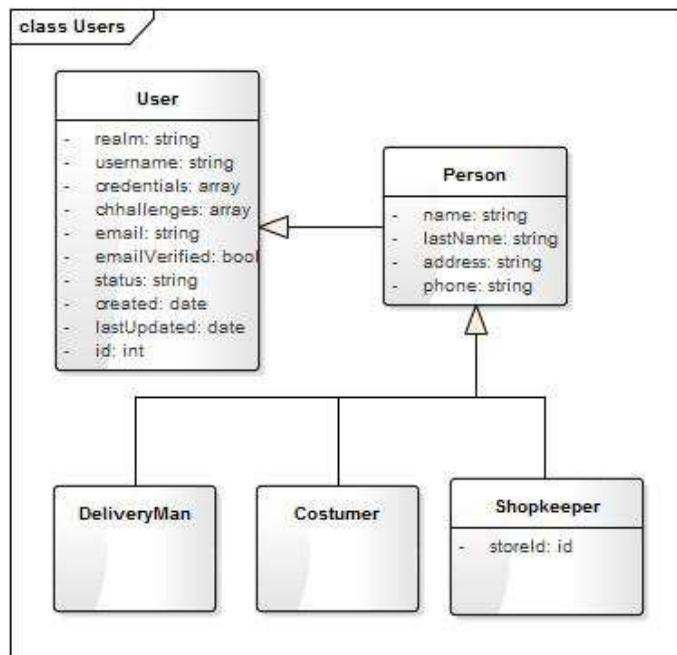


Figure 12.5: Modelo de usuarios.

En la Figura 12.6 muestra el despliegue que tiene el servicio.

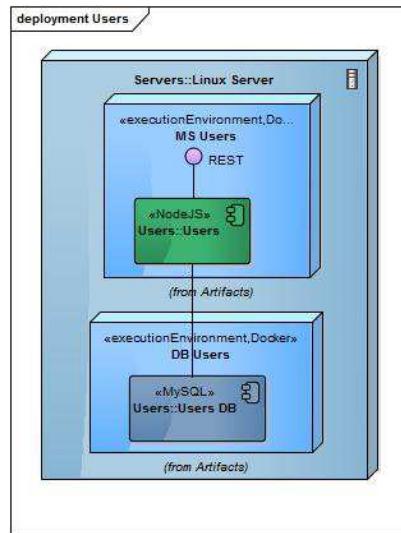


Figure 12.6: Despliegue Órdenes

12.3 Despliegue

La Figura 12.7 nos permite visualizar el diagrama de despliegue general del producto. En este se logran apreciar los servicios planteados (verde), sus respectivas bases de datos (azul grisáceo) y la interfaz (violeta). Cada uno de los elementos está contenido en su respectiva imagen docker (azul claro).

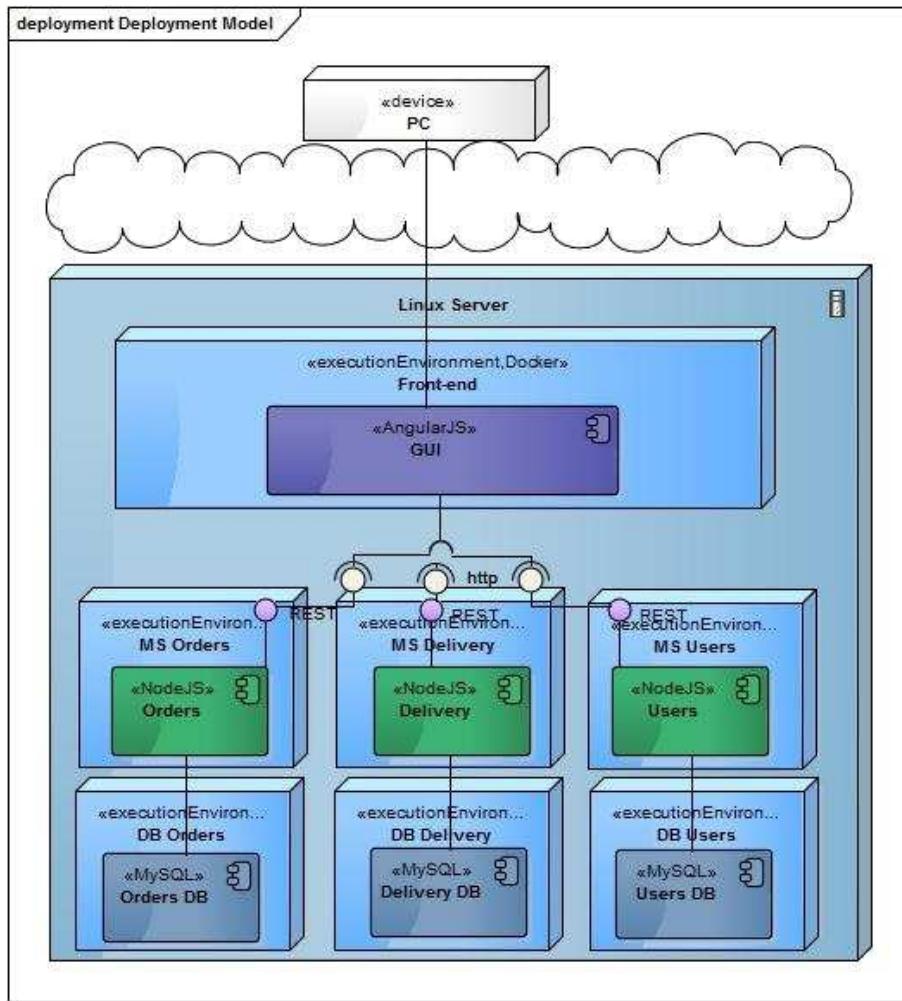


Figure 12.7: Diagrama de despliegue.

PARTE III

**CIERRE DE LA
INVESTIGACIÓN**

Capítulo 13

RESULTADOS Y DISCUSIÓN

A continuación se describe brevemente el proceso que se utilizó para la construcción de uno los microservicios (los otros se fabricaron de una manera similar). Adicionalmente, se revisará como se adoptaron ventajas de realizar un proceso de desarrollo dirigido por modelos (MDD) y de las herramientas usadas para su construcción.

13.1 Descripción del proceso de construcción

Lo primero que se abordó fue crear el diseño del modelo de información del servicio, por ejemplo el servicio de Gestión de Órdenes que puede ser apreciado en la Figura 12.1. Tras tener el modelo se inició por crear el nuevo proyecto utilizando Loopback (Figura 13.1).

```
diego@diego-VirtualBox:~/pgrado$ slc loopback
  _-----_
 |(o)-| | Let's create a LoopBack |
 |(U)-| | application!
 / \ A \
 |   |
 ,---Y---,
?
? What's the name of your application? orders
? Enter name of the directory to contain the project: orders
  create orders/
    info change the working directory to orders
?
? What kind of application do you have in mind? api-server (A LoopBack API server with local User auth)
Generating .yo-rc.json
```

Figure 13.1: Crear proyecto en Loopback

Una vez creadoel proyecyo, ya se pueden aprovechar las transformaciones y las herramientas que nos proporciona Loopback, en la generación de código.

En la Figura 13.2 podemos ver cómo se puede generar un servicio funcional con tan solo algunas líneas de comandos –lo que tiene como objetivo no solo agilizar el desarrollo sino minimizar los posibles errores que se puedan cometer en procesos repetitivos-.

```
diego@diego-VirtualBox:~/pgrado/orders$ slc loopback:model
? Enter the model name: Product
? Select the data-source to attach Product to: db (memory)
? Select model's base class PersistedModel
? Expose Product via the REST API? Yes
? Custom plural form (used to build REST URL): products
? Common model or server only? common
Let's add some Product properties now.

Enter an empty property name when done.
? Property name: id
    invoke loopback:property
? Property type: number
? Required? Yes
? Default value[leave blank for none]: 

Let's add another Product property.
Enter an empty property name when done.
? Property name: name
    invoke loopback:property
? Property type: string
? Required? No
? Default value[leave blank for none]: 

Let's add another Product property.
Enter an empty property name when done.
? Property name: description
    invoke loopback:property
? Property type: string
? Required? No
? Default value[leave blank for none]: 

Let's add another Product property.
Enter an empty property name when done.
? Property name:
diego@diego-VirtualBox:~/pgrado/orders$ █
```

Figure 13.2: Creación modelo en loopback

Una vez creado el modelo, podemos iniciar la aplicación y ya tenemos nuestro modelo con las interfaces REST básicas, como se puede ver en el API Explorer (Figura 13.3).

The screenshot shows the StrongLoop API Explorer interface. At the top, there's a header with the title 'StrongLoop API Explorer', a 'Token Not Set' button, an 'accessToken' input field, and a 'Set Access Token' button. Below the header, the word 'orders' is displayed in bold. Underneath, there's a section titled 'Product' with four entries:

- GET /products**: Find all instances of the model matched by filter from the data source.
- PUT /products**: Update an existing model instance or insert a new one into the data source.
- POST /products**: Create a new instance of the model and persist it into the data source.
- GET /products/{id}**: Find a model instance by id from the data source.

Each entry includes a brief description of its function below the URL.

Figure 13.3: API Explorer

Con lo anterior ya podemos utilizar el microservicio desde los clientes, sin embargo antes de eso se despliega en un contendor Docker y se genera una imagen de este servicio. Ahora solo resta iniciar el contenedor (Figura 13.1) y ya tenemos la aplicación funcionando y desplegada desde un contenedor.

```
root@diego-VirtualBox:~# docker run -d -it --name ordersMS -p 3002:3000 diego/nodeserv:v0.4
root@diego-VirtualBox:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
61a725da10fd        diego/nodeserv:0.4   "node"              6 seconds ago     Up 4 seconds       0.0.0.0:3002->3000/tcp   ordersMS
27074a8c8743        diego/nodeserv:0.4   "node"              3 days ago        Up 3 days          0.0.0.0:3001->3000/tcp   orders4
cea62ae0eb07        diego/orders:v0.2    "node"              3 days ago        Up 3 days          3306/tcp           order2
7046de680886        percona/percona-server:latest  "/entrypoint.sh"  2 weeks ago       Up 3 days          3306/tcp           ps
root@diego-VirtualBox:~#
```

Table 13.1: Iniciar un conenedor con el microservicio

Con este despliegue los clientes ya pueden hacer uso del nuevo microservicio por el puerto especificado (3002).

Es importante recalcar que el middleware que se encarga del descubrimiento y/o balanceo de servicio no se tiene en cuenta para este prototipo, pero puede escogerse según las necesidades del proyecto. Este puede ir desde un servicio básico que permita hacer esto o alguna maquina física que lo haga (como un balanceador).

13.2 Descripción del prototipo

En esta sección se describirá brevemente el prototipo funcional del proyecto. La aplicación cuenta con una interfaz visual sencilla realizada con AngularJs, la cual cuenta con las funcionalidades básicas para la creación y despacho de pedidos a las tiendas. La aplicación es una aplicación de página única - SPA por sus siglas en inglés, Simple Page Application- que consume los servicios REST (explicados en capítulos anteriores) expuestos (fabricados en NodeJS) en el back-end.

Gráficamente, la aplicación intenta mantener la menor cantidad de componentes visuales que permitan una simplicidad en su utilización, sin embargo la interfaz, está diseñada bajo una perspectiva del autor, por lo que es importante denotar que aún falta un trabajo más profundo para la evaluación del producto desde la perspectiva del usuario final, trabajo que se espera se vaya dando a lo largo de la maduración de la aplicación.

A continuación se presentaran y expondrán las características principales de la aplicación desde la perspectiva visual. Para iniciar la Figura 13.4 permite observar la página inicial de bienvenida que cuenta con un menú con 3 opciones: Inicio, Acceder y Registrarme.



Figure 13.4: Página inicial.

La opción de “Acceder” del menú nos permitirá ingresar a la aplicación y nos solicita nuestro email registrado y la contraseña como se observa en la Figura 13.5.

The screenshot shows the login interface for the application. At the top, there is a header bar with three buttons: 'Inicio', 'Acceder', and 'Registrarme'. Below this is a central 'Login' form. It contains two input fields: 'Email' with the value 'tendero@tienda.com' and 'Contraseña' with several dots indicating the password. At the bottom of the form is a blue 'Acceder' button.

Figure 13.5: Acceder.

La opción de “Registrarme”, nos permite visualizar un pequeño formulario (Figura 13.6) con los datos básicos para suscribirse a la aplicación.

The screenshot shows the registration interface for the application. At the top, there is a header bar with three buttons: 'Inicio', 'Acceder', and 'Registrarme'. Below this is a left sidebar with the title 'Registrarme' and a note: 'Para registrarte tan solo llena el formulario con tus datos y LISTO!!!'. To the right is a main form with several input fields: 'Nombre' with the value 'Steve Harris', 'Telefono' with the value '3222222222', 'Direccion' with the value '22. Avenue', 'Email' with the value 'steve@iron.com', and 'Contraseña:' with several dots indicating the password. At the bottom of the form is a blue 'Registrarme' button.

Figure 13.6: Registrarme.

Tras acceder a la aplicación, se puede observar una pantalla inicial, igual a la pantalla principal sin estar dentro de la aplicación, sin embargo el menú cambia para ofrecer dos opciones nuevas “Realizar Pedido” y “Estado de mis pedidos”, como se observa en la Figura 13.7.



Figure 13.7: Inicio, con usuario registrado.

La opción “Realizar Pedido” permite al usuario crear un nuevo pedido (Figura 13.8), o modificar un pedido en curso (Figura 13.9). Una vez el usuario haya completado su pedido, podrá utilizar la opción de “Realizar Pedido” lo que permitirá que el tendero pueda atender el pedido, y al usuario crear una nueva orden (Figura 13.8).

La opción “Realizar Pedido” permite al usuario crear un nuevo pedido (Figura 13.8), o modificar un pedido en curso (Figura 13.9). Para modificar -o crear- el pedido el usuario solo tendrá que escribir que producto desea en el campo “Producto” y dar clic en el botón de “Aregar”. Automáticamente el producto se mostrara en la parte izquierda en forma de un listado (como se aprecia en la Figura 13.9). Si el usuario desea editar el producto, deberá dar clic sobre este en el listado y este se podrá editar en el formulario central, que contara con la opción de “Guardar” y “Cancelar” (Figura 13.10). En caso de querer eliminar de la lista el producto, el usuario solo deberá dar clic a la “X” del producto. Una vez el usuario haya completado su pedido, podrá utilizar la opción de “Realizar Pedido” lo que permitirá que el tendero pueda atender el pedido, y al usuario crear una nueva orden (Figura 13.8).

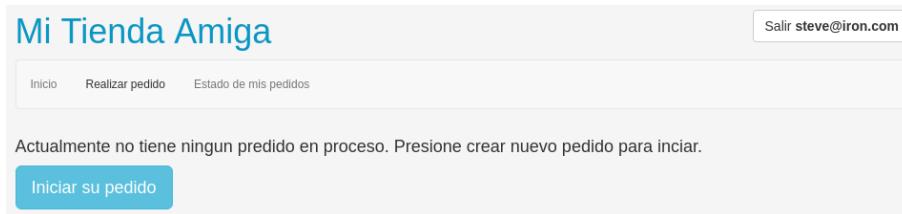


Figure 13.8: Realizar pedido.



Figure 13.9: Modificar el pedido.



Figure 13.10: Editar producto.

En la opción “Estado de mis pedidos” el usuario puede ver un listado de sus órdenes en la parte izquierda y al darle clic al pedido puede ver algunos detalles de este (Figura 13.11).

Orden #: 7
Repartidor: Felipe
Ubicación: En Camino

Detalles del estado

#	Producto
1	12 huevos
2	2 libras de arroz
3	1 bolsa de leche deslactosada

Figure 13.11: Estado de pedidos.

Si un tendero ingresara el menú principal contara con otra opción adicional “Administrar Pedidos”, la cual permite ver las órdenes realizadas por los usuarios e ir cambiando el estado de estas según su progreso (Figura 13.12).

Orden #: 5
Fecha: 2016-05-14 17:14:57
Estado: Pedido Entregado

Orden #: 6
Fecha: 2016-05-14 17:19:23
Estado: Preparando Pedido

Orden #: 7
Fecha: 2016-05-15 09:36:44
Estado: Preparando Pedido

Editar

Orden #: 7
Fecha: 2016-05-15 09:36:44

Estado del pedido: Preparando Pedido ▾
Guarda
Cancelar

#	Producto
1	12 huevos
2	2 libras de arroz
3	1 bolsa de leche deslactosada

Figure 13.12: Administrar Pedidos.

Capítulo 14

Conclusiones

14.1 Verificación, contraste y evaluación de los objetivos

A continuación se presentan algunas reflexiones que se generaron a lo largo del desarrollo del proyecto y del análisis de sus objetivos.

- Las encuestas y charlas con las personas que pueden estar interesadas, ayudan a dar un punto de partida para el diseño de lo que se desea, sin embargo no dan el panorama completo de lo que se puede llegar a construir, así que deben ser complementadas con otras técnicas y/o herramientas.
- Es muy útil tener a la mano diferentes técnicas para el diseño del producto, tales como la técnica persona, elevator pitch o visual story mapping (VSM). Este tipo de técnicas fueron desarrolladas con propósitos específicos, y se van desarrollando naturalmente, lo que permite comprenderlas con facilidad y generan inputs importantes para el planteamiento del producto -como se evidencio en la fase de incepción del proyecto-, es muy importante tenerlo en cuenta para futuros proyectos.
- Desarrollar un ejercicio de arquitectura empresarial (AE) ayuda a dar un norte y comprender las necesidades de negocio. Aunque no se logró desarrollar un ejercicio completo de AE, el desarrollo de los puntos de vista descritos en ArchiMate amplió la visión de negocio que generó un objetivo claro para el planteamiento técnico.
- El planteo de un backlog nutrido en conjunto con los puntos de vista de implementación y migración –de ArchiMate- generan un plan de

crecimiento del producto que se va generando por sí mismo por lo que ayuda a visualizar el potencial del proyecto y el camino que se quiere llevar.

- Gracias al auge que los microservicios han tenido en los últimos años, se han incrementado las herramientas para su diseño e implementación, y estas crecen exponencialmente. Con este abanico de nuevas posibilidades es de vital importancia para el desarrollo de un proyecto, el analizarlas cuidadosamente, teniendo como base no solo los requerimientos técnicos sino que también el equipo que desarrollará el mismo. El diseño de una solución se vuelve un poco más complejo sobre todo al nivel de datos, pues el independizar implica que de alguna manera tengo que tener una estrategia que me permita enlazar los datos con otros servicios, lo que se representa – al menos en el presente ejercicio – en duplicidad de información, lo cual rompe con los conceptos preconcebidos en sistemas relacionales.
- Se evidenció en el uso de ciertas herramientas -como Docker- que están relacionadas con la práctica de DevOps, permite ayudar a explotar el potencial de contar con una arquitectura basada en microservicio puesto que su uso mejora temas esenciales como el despliegue, escalado (no solo vertical sino horizontal) y aseguramiento de funcionamiento de un servicio.
- No se debe olvidar que el uso de cada nueva técnica y/o herramienta no nos trae solo ventajas, también nos ocasiona algunas complicaciones. El usar Docker, nos ayuda en diferentes temas y es una herramienta muy útil, pero añade mucha más complejidad por lo que es importante contar con estrategias que minimicen estos puntos. Por ejemplo, al usar múltiples contenedores es crítico contar con alguna estrategia para su monitoreo.
- La arquitectura de microservicios se puede ver y usar como una combinación (no es la única) de otros modelos -o arquitecturas-, y/o de sus principios y no por esto entra en conflicto con estas, por ejemplo el uso de SOA o de arquitecturas REST que están inmersos dentro de esta. También permite el uso de distintos modelos de desarrollo -como el dirigido por modelos- según las capacidades del equipo y de la tecnología usada.
- Es importante tener claras las capacidades del equipo de trabajo para plantear una solución, pues este punto puede generar una toma errónea

de decisiones lo que generará sobre costos en el proyecto.

- Dada la esencia asíncrona de esta arquitectura, su desarrollo y prueba en conjunto se vuelve un poco más complicada, por lo que es muy importante identificar el alcance de nuestro producto, para así tomar la decisión de adoptar este estilo o no.
- Tener la independencia de cada uno de los servicios permite hacer su mantenimiento y desarrollo mucho más simple, pues el enfoque está puesto en esta tarea y en ese contexto en específico. Adicionalmente permite cambiar técnicamente todo el sistema, permitiendo escoger diferentes lenguajes y herramientas que pueden resolver mejor un determinado problema.
- El contar con muchos servicios escritos diferentes lenguajes se puede convertir en un caos, si no se tiene claramente definido las capacidades de la empresa desarrolladora. Una definición de arquitectura debe tener en cuenta este tipo de restricciones y no limitarse a los temas técnicos inmediatos, por ejemplo, el mantenimiento a largo plazo del sistema.

14.1.1 Síntesis del modelo propuesto

Con el fin de presentar una pequeña síntesis del modelo propuesto para la solución la Figura 14.1 nos permite visualizar el diagrama de despliegue de la solución (a un alto nivel). En esta imagen se puede observar los puntos de interés que se quisieron abarcar con el modelado, a lo largo del proyecto. A grandes rasgos estos fueron:

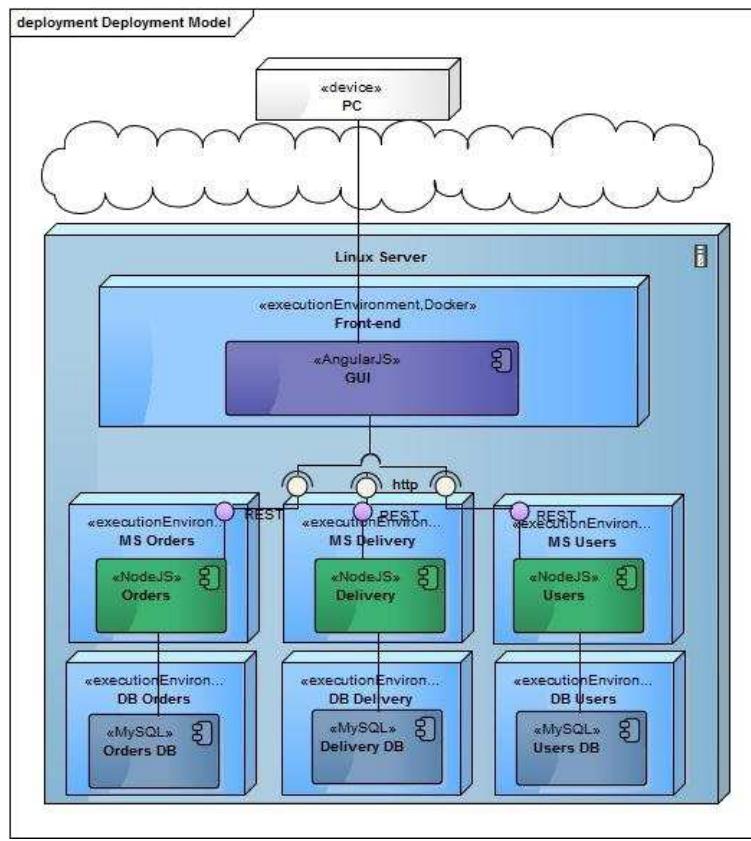


Figure 14.1: Despliegue de la Solución

Independencia entre los microservicios. Cada uno de ellos puede existir sin la necesidad de conocer la existencia de otro.

Tamaño. Cada microservicio es relativamente pequeño ya que cuenta con muy pocas clases.

Enfoque y mantenimiento. Los microservicios se enfocan en realizar un fragmento de la lógica específico, lo que permite que el mantenimiento de estos sea más simple (también gracias a su tamaño).

Facilidad en construcción. Gracias a su enfoque y tamaño, el microservicio se puede construir con técnicas de desarrollo que agilizan su construcción. Por ejemplo el uso de un desarrollo dirigido por modelos que se utilizó en el proyecto.

Facilidad en el despliegue. El uso de una herramienta como Docker, permite encapsular nuestro microservicio de manera tal que una vez se ha validado en un entorno (como pre producción o QA o cualquier otro) de-

terminado luego es fácilmente desplegable en otro asegurando su igual funcionamiento.

Escalamiento. Nuevamente gracias al estar auto contenido en una imagen Docker esta nos permite crear fácilmente una instancia –una línea de comando-, la cual es funcional. Si necesitamos escalar, simplemente creamos otro contenedor basado en la misma imagen. Esto nos da muchas posibilidades, sobre todo para entornos en las que se requiere manejar un alto nivel de transacciones en momentos específicos. La Figura 14.2 nos permite visualizar como se vería un servicio que necesitará aumentar su capacidad.

Crecimiento específico. Una de las más grandes ventajas –a modo personal- de contar con este modelo es el permitir escalar lo que realmente se necesita, es decir, centrarnos en escalar los puntos críticos que realmente se necesitan en un momento determinado.

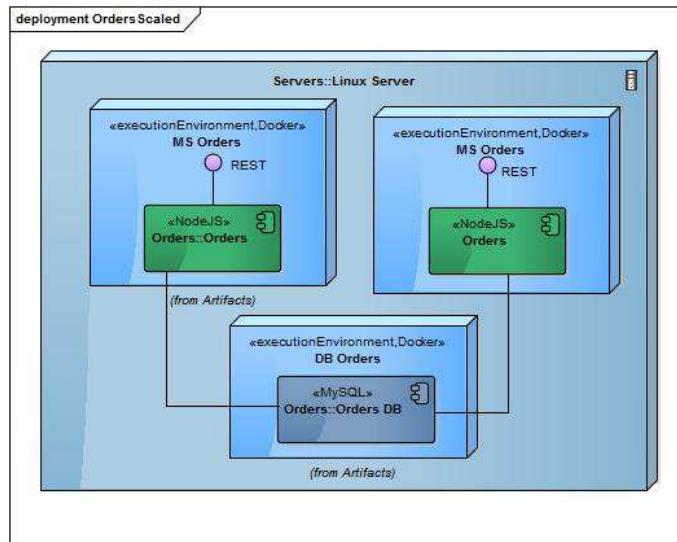


Figure 14.2: Aumento de capacidad de un microservicio

14.1.2 Aportes originales

Los aportes que este trabajo pretende brindar se pueden denotar por:

- Es importante tener en el radar nuevos modelos o técnicas para el desarrollo de software, estas pueden ser usadas según las necesidades del proyecto en el que nos encontramos, así que estar actualizados con respecto a este punto es de vital importancia.

- Identificar a negocios que no cuentan con un gran capital para competir en un mundo tecnológico abrir las puertas no solo para beneficio de estos sino de los consumidores finales y puede desarrollar un producto realmente provechoso para las diferentes partes.

Capítulo 15

PROSPECTIVA TRABAJO DE GRADO

Es importante tomar el presente trabajo como un punto de partida desde las dos perspectivas que se desearon abordar. Una perspectiva técnica –de desarrollo de software- y la otra comercial.

De la primera, la experimentación y creación de una idea básica del modelado de soluciones de software utilizando una enfoque de microservicios, se espera, que sirva como un punto de partida para la aplicación de este tipo de soluciones en diferentes tipos de proyectos de software, en el que sea realmente aprovechable este tipo de modelado. La segunda parte a tener en cuenta es la continuación del producto descrito a lo largo del presente documento. Para esto el siguiente paso es realizar una pequeña prueba de concepto en algunos negocios, de diferentes ubicaciones, por ejemplo, alguno en zonas familiares y otro en zonas de oficinas, con el fin de obtener información inicial para validar el producto. Dependiendo del resultado de estas pruebas, se podría continuar con el desarrollo del producto, con lo que se ha planteado en capítulos anteriores y alimentando el backlog del proyecto.

No se da una estrategia clara, pero es importante dar seguimiento y retroalimentación en cada uno de los pasos, aunque si hay un punto claro del producto en su evolución es la inclusión de este mismo en las plataformas móviles (Android y IOS), puesto que estas se convierten cada día en un elemento imprescindible para el consumidor.

BIBLIOGRAFÍA

- [1] DINERO. La tienda, el aliado de la casa, Abril 2014.
- [2] EMAG. Architectures you always wondered about lessons learnt from adopting microservices at ebay, google, gilt, hailo and nearform. Magazine, August 2015.
- [3] FOWLER, M. Microservices, March 2014.
- [4] GECHEV, M. Angularjs in patterns, 2010.
- [5] GOMEZ, D. U. Microservicios, arquitectura de información para un desarrollo ágil y eficiente, April 2015.
- [6] GROUP, T. O. Archimate® 2.0 specification. Electronic, January 2012.
- [7] GROUP, T. O. Welcome to togaf® version 9.1, an open group standard, February 2016.
- [8] HARRISON, R. *TOGAF 9 Foundation Study Guide*. Van Haren Publishing, 2009.
- [9] MAURO, T. Adopting microservices at netflix: Lessons for architectural design, February 2015.
- [10] NEWMAN, S. *Building Microservices*. O'Reilly, 2015.
- [11] PATTON, J. *User Story Mapping: Discover the Whole Story, Build the Right Product*. O'REILLY, 2009.
- [12] RICHARDSON, C. Introduction to microservices, May 2015.

Anexo A

Encuestas

Con el fin de recopilar información inicial para determinar la viabilidad del proyecto, se creó una encuesta muy sencilla la cual buscaba dar una referencia del uso de tiendas y la posibilidades tecnológicas, así como algunos puntos de partida para dar forma al producto. La encuesta se compone de las preguntas:

- Compra usted en tiendas de barrio.
- Que tan frecuente compra usted en las tiendas de barrio.
- Si utiliza tiendas de barrio, estas son cerca a:
- Que tan a menudo usa usted la misma tienda(s).
- Realiza pedidos a la tienda de barrio, telefónicamente.
- Sus pedidos o compras se realizan en un horario similar.
- Utiliza algún dispositivo diariamente con acceso a internet.
- Utiliza alguna aplicación, o realiza compras de productos al detal por internet.

Los resultados de las mismas se pueden observar a continuación:

- Compra usted en tiendas de barrio.

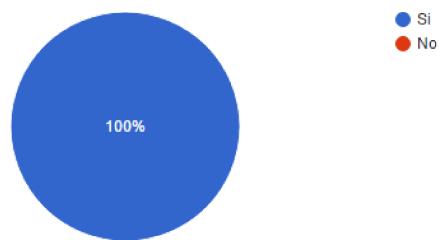


Figure A.1: Resultados Compra en tiendas.

- Que tan frecuente compra usted en las tiendas de barrio.

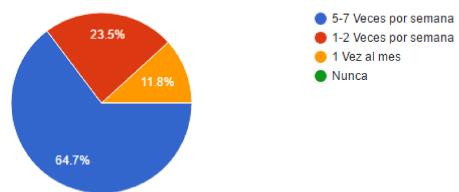


Figure A.2: Resultados Frecuencia de compra.

- Si utiliza tiendas de barrio, estas son cerca a:

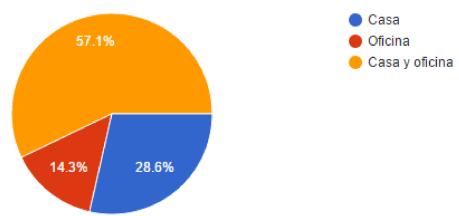


Figure A.3: Resultados Cercania

- Que tan a menudo usa usted la misma tienda(s).

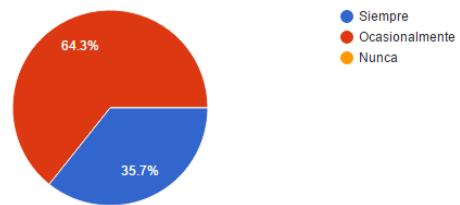


Figure A.4: Resultados Frecuencia de compra en el mismo lugar.

- Realiza pedidos a la tienda de barrio, telefónicamente.

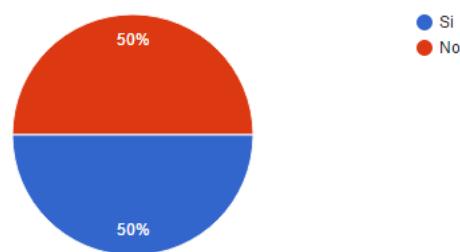


Figure A.5: Resultados Pedidos por teléfono.

- Sus pedidos o compras se realizan en un horario similar.

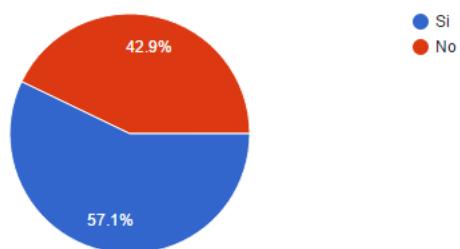


Figure A.6: Resultados Horario similar

- Utiliza algún dispositivo diariamente con acceso a internet.



Figure A.7: Resultados Acceso a Internet Diariamente.

- Utiliza alguna aplicación, o realiza compras de productos al detal por internet.

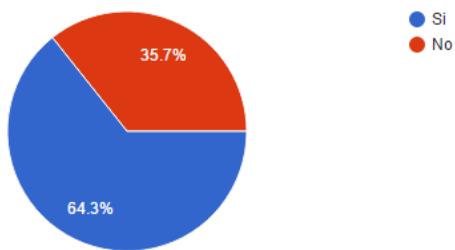


Figure A.8: Resultados Compras en linea

Teniendo en cuenta los resultados de la encuesta, se infiere que las personas continúan haciendo uso de las tiendas de barrio y estas cuentan con acceso a internet por lo que abrir otro canal de comunicación puede llegar a todos los clientes. También nos ayuda a plantear algunas funcionalidades básicas que puede tener el producto con base, es su frecuencia de uso, e incluso la fidelidad hacia alguna tienda.

Anexo B

Descripción conceptual de la Arquitectura

La arquitectura empresarial busca la optimización de los diferentes procesos y recursos de una empresa (ya sean nuevos o legados) para que se integren correctamente y puedan soportar y llevar a cabo la estrategia de negocios de la misma. Con el propósito de mantener una ventaja competitiva y responder más efectivamente al cambio. Con el propósito de satisfacer esta necesidad nacen diferentes marcos de trabajo, como los son: Zachman EA Framework, FEAFF (Federal Enterprise Architecture Framework), SAP EA Framework y TOGAF, tan solo por nombrar algunos de los más importantes.

Los Framework empresariales, buscan aumentar la velocidad en la adopción e implementar de la práctica de arquitectura empresarial. Estos generan diferentes ventajas, tales como [8]:

- Una operación de IT más eficiente.
 - Bajando el costo de desarrollo y soporte, así como mejorando la interoperabilidad de los sistemas.
- Mejorar el retorno de inversión, reduciendo el riesgo para futuras inversiones.
 - Reduciendo la complejidad en IT, maximizándola y permitiéndole flexibilidad para comprar o elaborar soluciones de IT
- Adquisiciones más rápidas, simples y económicas.

ANEXO B. DESCRIPCIÓN CONCEPTUAL DE LA ARQUITECTURA108

- Con un plan coherente y la información disponible las decisiones son más simples, los procesos más rápidos y la heterogeneidad de sistemas, permiten entornos con diferentes vendedores.

B.1 TOGAF

The Open Group Architecture Framework (TOGAF) es un framework abierto, creado por el esfuerzo de más de 300 empresas miembros del Foro de Arquitectura, líderes en IT y representantes de las mejores prácticas del desarrollo de la arquitectura.

TOGAF es presentado en la forma de un documento que puede ser encontrado en la página web de the open group[7]. TOGAF está compuesto por siete partes como lo indica la Figura B.1, estas son:

- I. Introduction. Contempla los conceptos clave necesarios para el uso de TOGAF.
- II. ADM (Architecture Development Method). Describe el método paso a paso para llevar a cabo la práctica de la arquitectura empresarial, es la esencia de TOGAF
- III. ADM Guidelines and Techniques. Técnicas y guías para utilizar en TOGAF y en el ADM.
- IV. Architecture Content. Contiene un metamodelo estructural de artefactos arquitecturales así como entregables de arquitectura más comunes.
- V. Enterprise Continuum and Tools. Describe la taxonomía para almacenar los productos del ejercicio de arquitectura.
- VI. Reference Models. Provee los modelos de referencia arquitectural, tales como el III-RM(Integrated Information Infrastructure Reference Model).
- VII. Architecture Capability Framework. Esta parte contiene los procesos, organización, habilidades, responsabilidades y roles requeridos para la práctica de arquitectura.

ANEXO B. DESCRIPCIÓN CONCEPTUAL DE LA ARQUITECTURA109

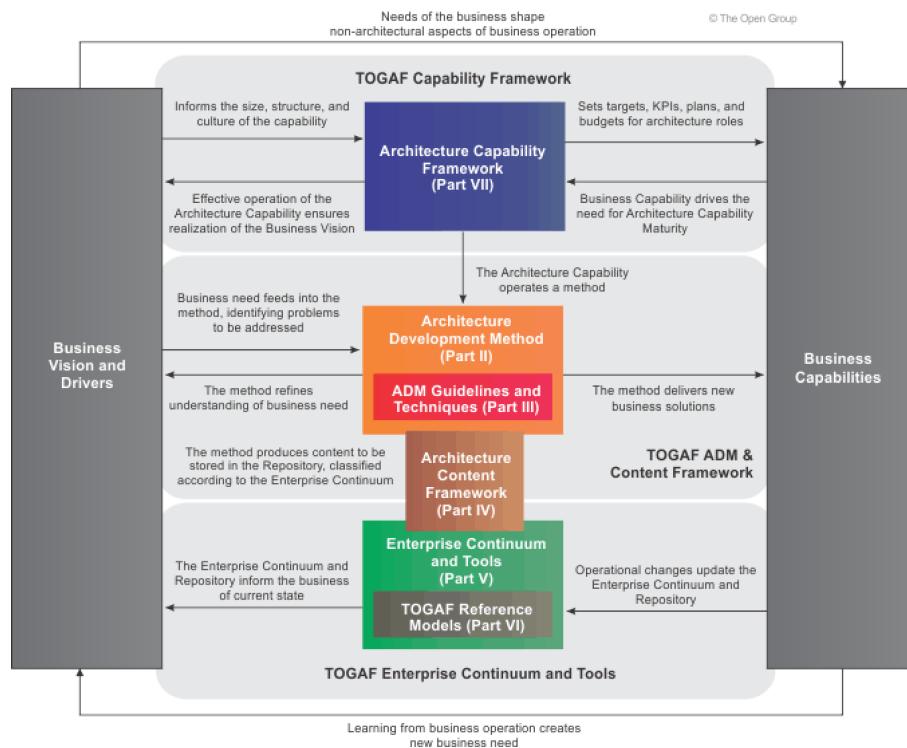


Figure B.1: Estructura TOGAF [7]

B.2 ADM

El ADM (Architecture Development Method), constituye el corazón de TO-GAF, es el paso a paso que guía la práctica de arquitectura. Se caracteriza por ser iterativo, no solo sobre todo el proceso sino dentro de cada fase. La estructura básica del ADM se puede ver en la Figura B.2.

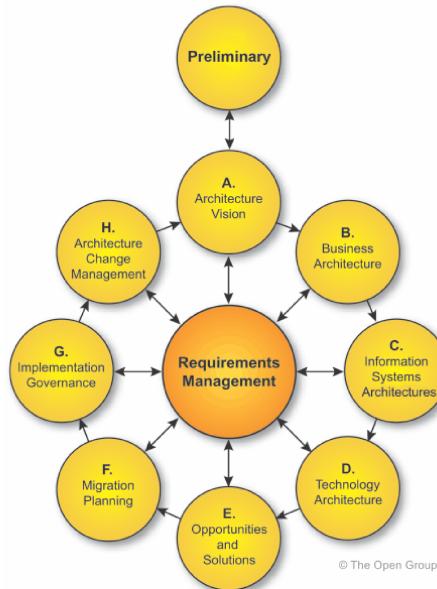


Figure B.2: ADM [7]

Cada fase consta de una serie de pasos específicos para cada una de ellas.

La fase de requerimientos es trasversal al método y es encargada de controlar que los requerimientos sean tratados de una manera apropiada.

Otro punto importante en el ADM es que un producto generado en alguna fase, es usado por fases posteriores como entrada en su proceso.

B.3 ArchiMate

ArchiMate, estandar del Open Group, es un lenguaje de modelado abierto e independiente para representar una arquitectura empresarial, que es soportado por diferentes vendedores empresas de consultoría. ArchiMate proporciona instrumentos a los arquitectos empresariales para describir, analizar y visualizar las relaciones entre los dominios de negocio de una manera precisa[6].

Un gran desafío para el desarrollo de un meta modelo genérico para la arquitectura empresarial es determinar el balance entre la especificidad de los lenguajes de cada domino arquitectural y una descripción general de conceptos arquitecturales que permita reflejar un sistema tan solo como un conjunto de entidades relacionadas. La Figura B.3 ilustra que estos conceptos pueden ser descritos en diferentes niveles de especialización.

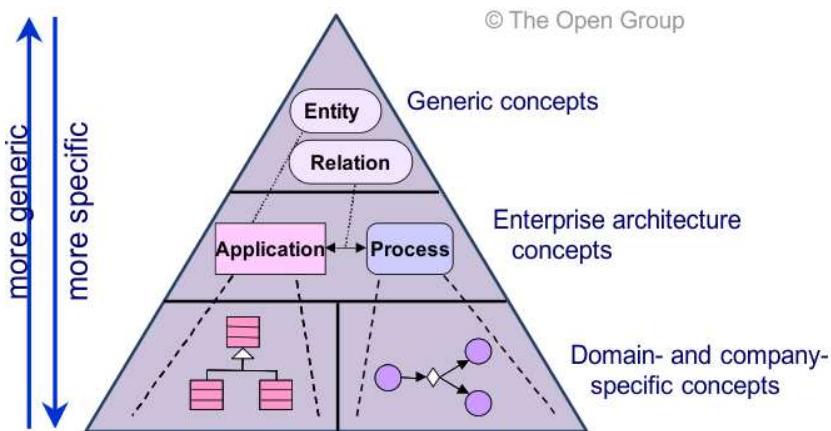


Figure B.3: Metamodelos en diferentes niveles de especificidad [6].

ArchiMate inicia de un conjunto de conceptos –relativamente- genéricos (parte superior de la Figura B.3). Uno de los elementos más importantes del lenguaje es el hecho que está diseñado para ser lo más pequeño posible, olvidándose de cumplir las necesidades de todos los tipos de usuario y enfocándose en permitir el modelado de 80% de casos prácticos.

B.3.1 Conceptos Base

La base del lenguaje se puede observar en la Figura B.4, en este se muestra los elementos que componen el lenguaje, los tres tipos principales de elementos son:

- Elementos de estructuras activos (active structure elements): definido como una entidad que es capaz de desempeñar un comportamiento. Ej. Actores de negocio, componentes de aplicación, dispositivos que muestran el comportamiento actual, etc.
- Elementos de comportamiento (behavior elements): Es definido como una unidad de actividad desempeñada por uno o más elementos de estructuras activos.
- Elementos de estructuras pasivos (passive structure elements): definido como el objeto en el cual el comportamiento es desempeñado.

Estos elementos fueron inspirados por el lenguaje natural, una oración que se compone de sujeto (estructura activa) verbo (comportamiento) y objeto

ANEXO B. DESCRIPCIÓN CONCEPTUAL DE LA ARQUITECTURA112

(estructura pasiva). Adicionalmente se hace una distinción entre la vista interna y externa. Los elementos de la vista externa son:

- Servicio (service): Es definido como una unidad de funcionalidad que el sistema expone a su entorno, mientras que esconde operaciones internas las cuales proveen un valor determinado.
- Interfaz (interface): Es definida como un punto de acceso donde uno o más servicios están disponibles al entorno.

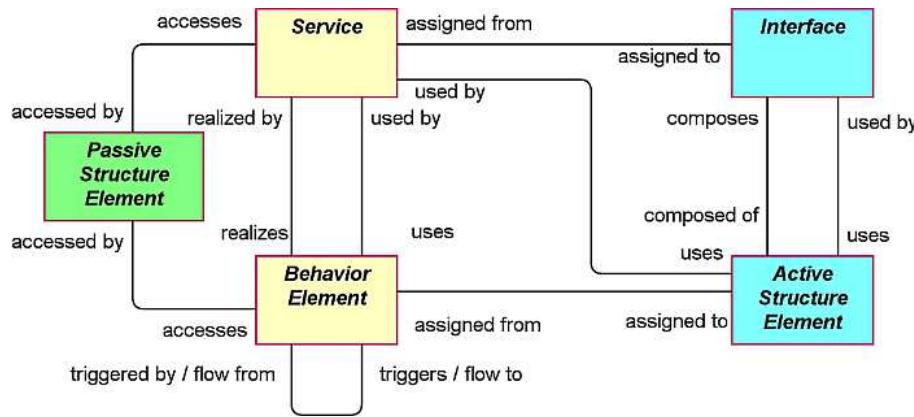


Figure B.4: Modelo genérico de conceptos base de ArchiMate[6].

B.3.2 Capas

ArchiMate define tres capas principales estas son:

- Capa de Negocios: ofrece productos y servicios a clientes externos que son realizados en la organización por procesos de negocio y desarrollados por actores de negocio.
- Capa de Aplicación: soporta la capa de negocios con servicios de aplicación los cuales son realizados por aplicaciones de software.
- Capa de Tecnología: ofrece servicios de infraestructura (almacenamiento, procesamiento, comunicación, etc.) necesarios para las aplicaciones.

B.3.3 Framework ArchiMate

Los conceptos y capas explicadas anteriormente pueden verse representados como un framework de nueve celdas, como se ilustra en la Figura B.5 .

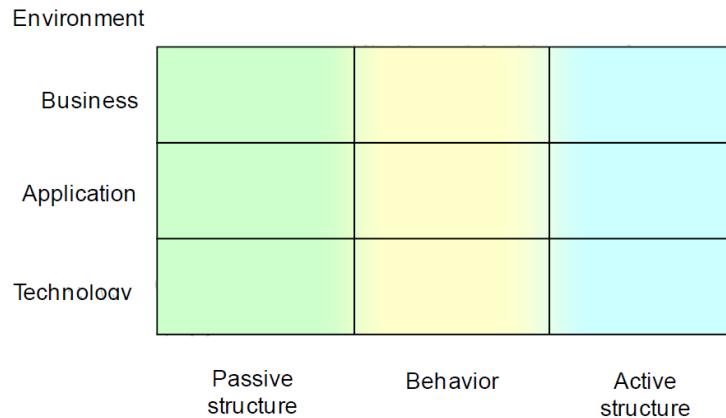


Figure B.5: Framework Arquitectural [6].

Dado que existen muchos otros conceptos que son requeridos en una arquitectura empresarial, se adicionaron nuevas extensiones para lograr cumplir estas expectativas, estas son la extensión de motivación y la extensión de migración e implementación.

B.3.4 Extensión de motivación

Esta extensión adiciona conceptos motivacionales tales como meta (goal), principio (principle) y requerimiento (requirement). Se encarga de representar como la arquitectura está alineada con el contexto.

La principal razón para introducir esta extensión es soportar el manejo de requerimientos, la Fase preliminar y Fase A (visión de arquitectura) del ADM de TOGAF.

La Figura B.6 ilustra la relación que tienen los elementos motivacionales de la extensión con los elementos base de Archimate.

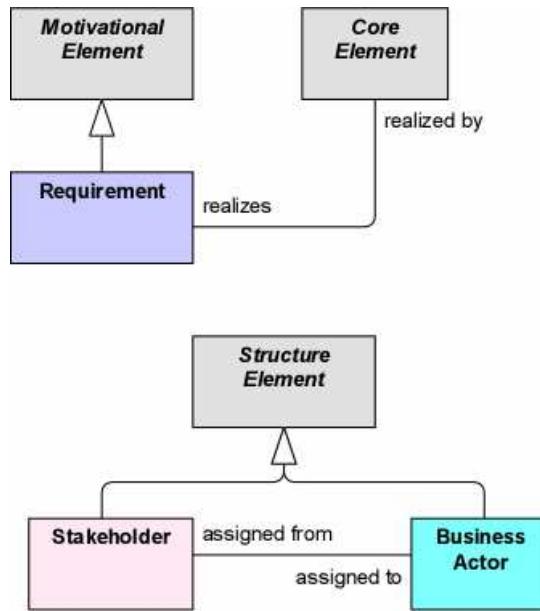


Figure B.6: Relación entre los concetos motivacionales y base[6].

B.3.5 Extensión de implementación y migración

La extensión de implementación y migración adiciona conceptos para soportar las fases finales del ADM, Fase E (Oportunidades y soluciones), Fase F (planeación de migración) y Fase H (gobierno de la implementación).

Esta extensión adiciona conceptos para modelado de programas de implementación, proyectos para soportar programas, portafolios, administración de proyectos, conceptos de mesetas y soporte para planeación de migración. La Figura B.7 ilustra la relación existente entre los conceptos base, la extensión de motivación y la extensión de implementación y migración.

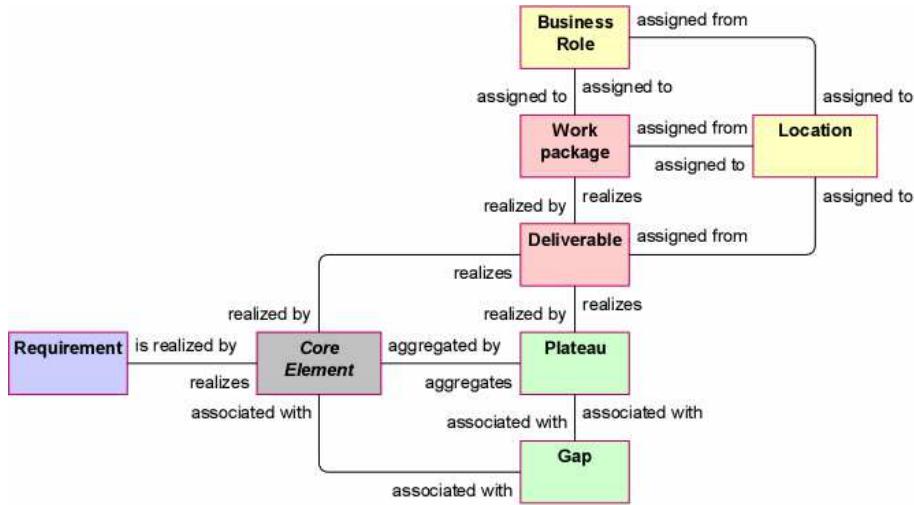


Figure B.7: Relación entre los conceptos base, motivacionales e implementación y migración [6].

B.3.6 ArchiMate y TOGAF

La Figura B.8 ilustra la relación que tiene ArchiMate con las Fases del ADM de TOGAF. En esta ilustración cabe aclarar que no hay una representación uno a uno entre los puntos de vista de TOGAF y los puntos de vista de ArchiMate, ArchiMate soporta los conceptos cubiertos en los puntos de vista de TOGAF.

ANEXO B. DESCRIPCIÓN CONCEPTUAL DE LA ARQUITECTURA116

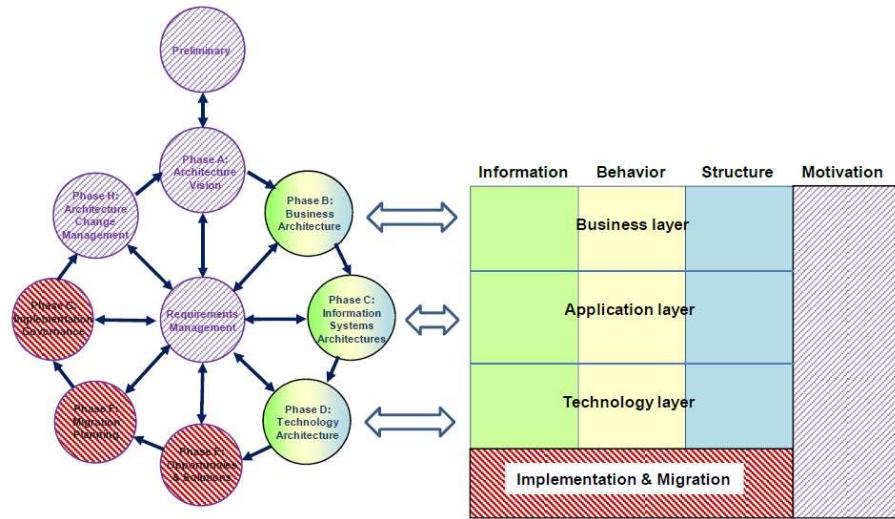


Figure B.8: Correspondencia entre TOGAF y Archimate[6].

B.3.7 Meta modelo de la capa de negocio

La Figura B.9 ilustra el meta modelo de los conceptos de la capa de negocio.

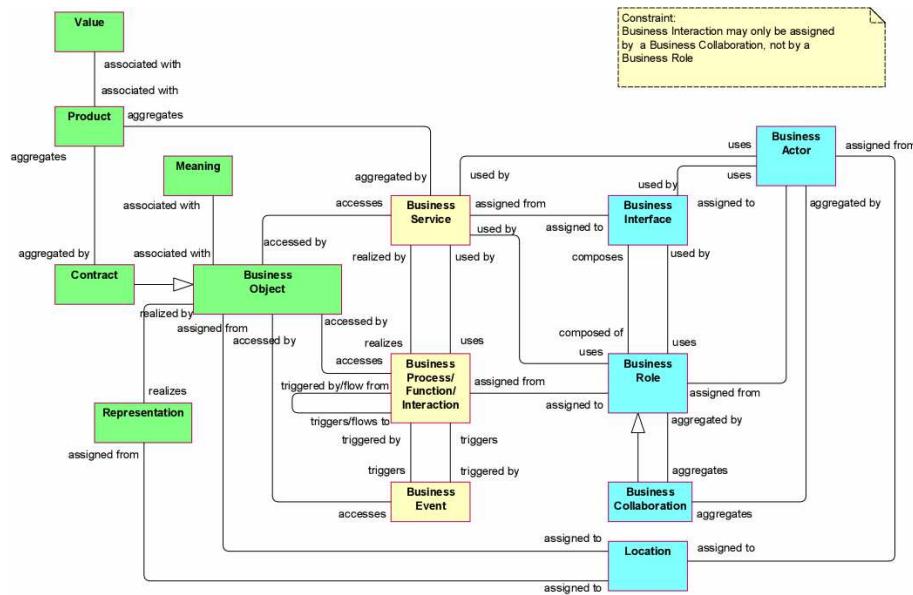


Figure B.9: Meta modelo de la capa de negocio[6].

ANEXO B. DESCRIPCIÓN CONCEPTUAL DE LA ARQUITECTURA117

La tabla B.1 describe los elementos referentes a la capa de negocio.

Concepto	Descripción	Grupo
Actor de Negocios	Entidad organizacional que es capaz de realizar un comportamiento.	Conceptos Estructurales
Rol de Negocios	Responsabilidad para realizar un comportamiento específico, al cual un actor puede ser asignado.	
Colaboración de Negocios	Un conjunto de dos o mas roles de negocio que trabajan juntos para realizar un comportamiento colectivo.	
Interface de Negocios	Punto de acceso en donde un servicio de negocio está disponible al entorno.	
Locación	Punto conceptual o físico.	
Objeto de negocio	Elemento pasivo que tiene relevancia desde una perspectiva de negocio.	
Proceso de Negocio	Elemento que agrupa comportamientos basado en un orden de actividades.	
Función de Negocio	Agrupa comportamiento basado en un criterio.	
Interacción de Negocio	Describe el comportamiento de una colaboración de negocio.	
Evento de Negocio	Algo que ocurre (interno o externo) e influencia un comportamiento.	
Servicio de Negocio	Servicio que satisface una necesidad de negocio para un cliente.	Conceptos de Comportamiento
Representación	Forma perceptible de la información que posee un objeto de negocio.	
Significado	Conocimiento o experticia presente en un objeto de negocio o su representación.	
Valor	El valor relativo, utilidad o importancia de un servicio o producto de negocio.	
Producto	Colección coherente de servicios.	Conceptos de Información
Contrato	Especificación formal o informal de un acuerdo (con derechos y obligaciones asociadas).	

Table B.1: Elementos capa negocio.

B.3.8 Meta modelo de la capa de aplicación

La Figura B.10 ilustra el meta modelo de los conceptos de la capa de aplicación.

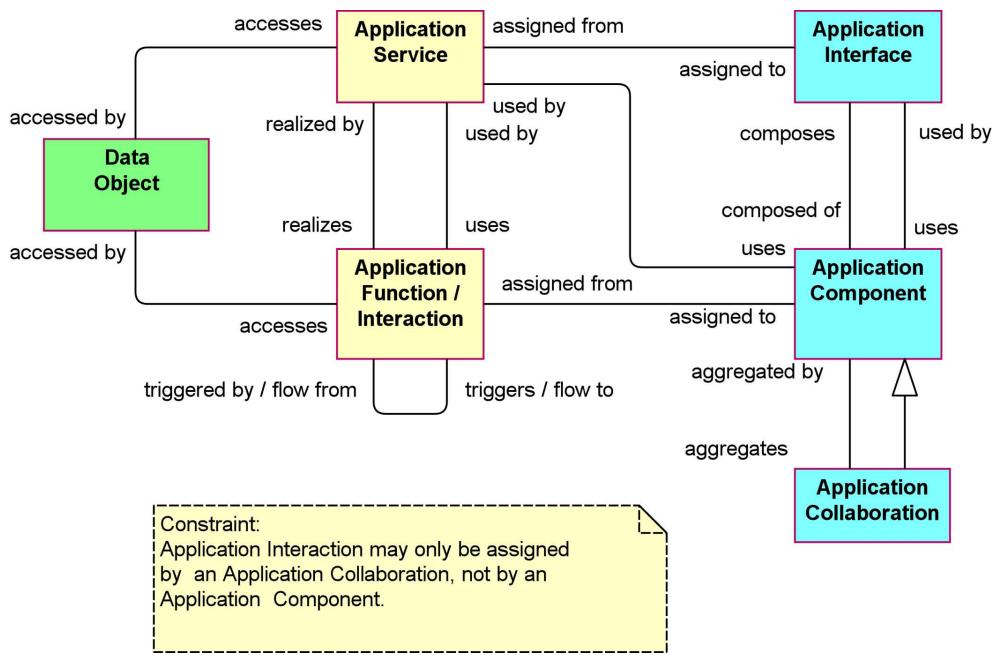


Figure B.10: Meta modelo de la capa de aplicación[6].

La tabla B.2 describe los elementos referentes a la capa de aplicación.

ANEXO B. DESCRIPCIÓN CONCEPTUAL DE LA ARQUITECTURA119

Concepto	Descripción	Grupo
Componente de aplicación	Modular, desplegable y reemplazable parte de un sistema de software que encapsula comportamiento y datos.	Conceptos Estructurales
Colaboración de aplicación	Un conjunto de dos o mas componentes de aplicación que trabajan juntos para realizar un comportamiento colectivo.	
Interface de aplicación	Punto de acceso en donde un servicio de aplicación está disponible al otros componentes.	
Objeto de datos	Elemento pasivo para procesos automatizados	
Función de aplicación	Elemento que agrupa comportamientos automatizados que pueden realizar un componente de aplicación.	Conceptos de Comportamiento
Interacción de aplicación	Elemento de comportamiento que describir el comportamiento de una colaboración de aplicación.	
Servicio de aplicación	Servicio que expone comportamiento automaizado.	

Table B.2: Elementos capa aplicación.

B.3.9 Meta modelo de la capa de tecnología

La Figura B.11 ilustra el meta modelo de los conceptos de la capa de tecnología.

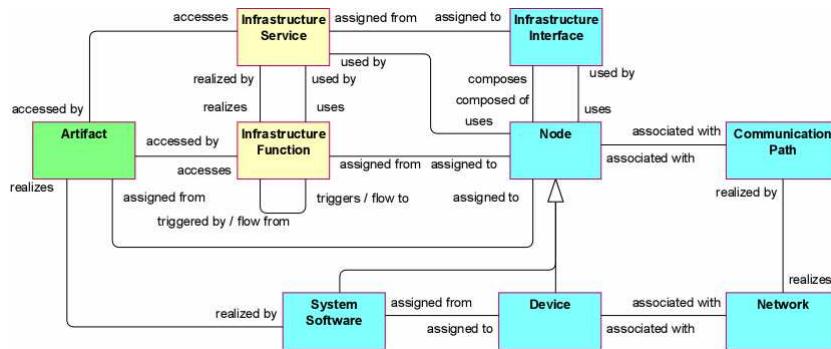


Figure B.11: Meta modelo de la capa de tecnología[6].

ANEXO B. DESCRIPCIÓN CONCEPTUAL DE LA ARQUITECTURA120

La tabla B.3 describe los elementos referentes a la capa de negocio.

Concepto	Descripción	Grupo
Nodo	Recurso computacional en donde artefactos pueden ser almacenados o desplegados.	Conceptos Estructurales
Dispositivo	Recurso de hardware en donde artefactos pueden ser almacenados o desplegados.	
Red	Medio de comunicación entre dispositivos.	
Sendero de Comunicación	Enlace entre dos o mas nodos, en el que pueden intercambiar información.	
Interface de infraestructura	Punto de acceso donde servicios de infraestructura ofrecidos po un nodo pueden ser accedidos por componentes de aplicación.	
Sistema software	Entorno de software para tipos específicos de componentes desplegados en el en forma de atefactos.	Conceptos de Comportamiento
Funcion de infraestructure	Elmento de comportamiento que agrupa otros comportamientos realizados por un nodo.	
Servicio de infraestructura	Unidad de funcionalidad visible externamente, proveeida por uno o mas nodos y expuesta por interfaces.	
Artefacto	Pieza física de datos que es usada o producida en un proceso de desarrollo de software, o por un despliegue.	Conceptos de Información

Table B.3: Elementos capa tecnología.

B.3.10 Dependencias entre capas

La figura B.12 muestra la relación entre los conceptos de las capas de negocio, aplicación y tecnología. Existen tres relaciones principales entre ellas, estas son:

- Usado Por. Se usan entre los servicios de aplicación y los diferentes

ANEXO B. DESCRIPCIÓN CONCEPTUAL DE LA ARQUITECTURA121

tipos de elementos de comportamiento de negocio, y entre las interfaces de aplicación y los roles de negocio.

- Realización. De objetos de datos a objetos de negocio, para indicar la representación digital de este.
- Asignación. Entre componentes de aplicación y procesos, funciones o interacciones de negocio, y entre interfaces de aplicación y servicios de negocio.

Adicionalmente pueden haber relaciones de agregación entre un producto y una aplicación o un servicio de infraestructura para indicar que estos pueden ser ofrecidos directamente al consumidor.

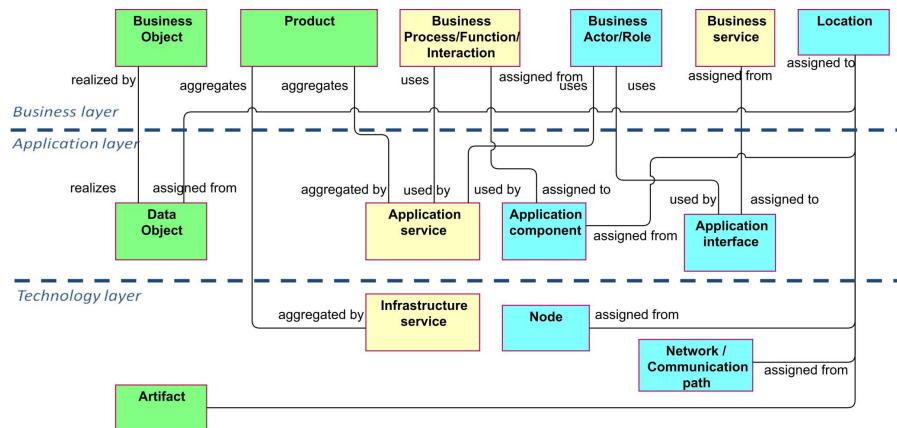


Figure B.12: Relaciones entre las capas de ArchiMate [6].

B.3.11 Relaciones

La Tabla B.4 muestra un resumen de las relaciones usadas por ArchiMate.

ANEXO B. DESCRIPCIÓN CONCEPTUAL DE LA ARQUITECTURA122

Relaciones Estructurales		Notación
Asociación	Relación entre objetos que nos es cubierta por otra.	-----
Acceso	Modela el acceso de conceptos de comportamiento a objetos de negocio o datos.>
Usado por	Modela el uso de servicios por procesos, funciones o interacciones y el accesos a interfaces por roles, componentes o colaboraciones.	→
Realización	Enlaza una entidad lógica con una más concreta que la realiza.▷
Asignación	Enlaza unidades de comportamiento con elementos activos (actores, roles, etc.) que las ejecutan.	•—•
Agregación	Indica que un objeto agrupa otros.	◇—
Composición	Indica que un objeto esta compuesto por otros.	◆—
Relaciones DimáMICAS		Notación
Flujo	Describe intercambio o transferencia (información, valores, etc.).	----->
Dispardor	Describe la relación temporal o causal entre procesos, funciones, interacciones y eventos.	→
Otras Relaciones		Notación
Agrupado	Indica que objetos del mismo o diferentes tipos pertenecen a una caracteristica común.	□
Unión	Conecta relaciones del mismo tipo	•
Specialización	Indica que un pbjecto es una especialización de otro.	→▷

Table B.4: Relaciones

B.3.12 Extensión de motivación

La Figura B.13 muestra el metamodelo de los conceptos de motivación. Los elementos motivacionales están relacionados a los elementos base por medio del concepto de requerimiento o restricción.

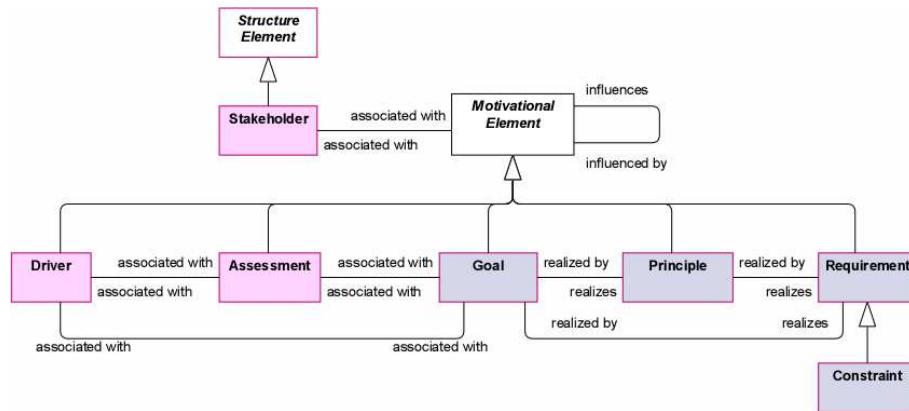


Figure B.13: Metamodelo de los conceptos de la extensión de motivación [6].

Los conceptos de esta extensión pueden ser apreciados en la Tabla B.5.

Concepto	Descripción
Implicado	El rol de un individuo, equipo u organización que representan sus intereses.
Manejador	Algo que crea, motiva e impulsa el cambio en la organización.
Valoracion	El resultado de algún análisis o algún manejador.
Objetivo	Un estado final que un implicado intenta lograr.
Requerimiento	Una necesidad que debe ser realizada por un sistema.
Restricción	Una restricción en la el sistema debe ser realizado.
Principio	Una propiedad normativa de todos los sistemas en un contexto dado.

Table B.5: Conceptos motivacionales.

El propósito de la extensión de motivación es modelar la motivación tras

ANEXO B. DESCRIPCIÓN CONCEPTUAL DE LA ARQUITECTURA124

los elementos base en alguna arquitectura empresarial. Como se aprecia en la Figura B.14 algunos elementos son relacionados directamente a los elementos base. Y aunque otros no, estos son relacionados indirectamente por relaciones derivadas.

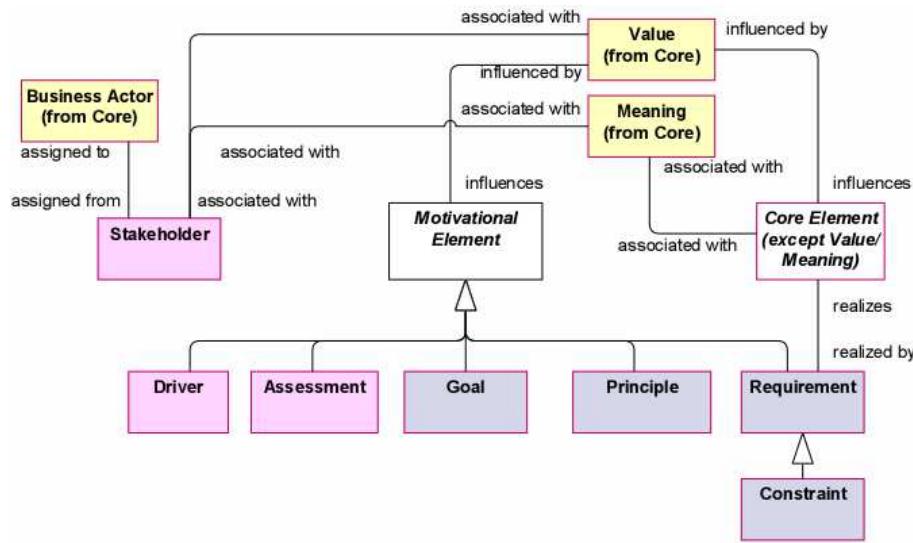


Figure B.14: Relación entre los conceptos base y los de motivación. [6].

B.3.13 Extensión de implementación y migración

En la Figura B.15 se muestra el metamodelo de los conceptos de implementación y migración.

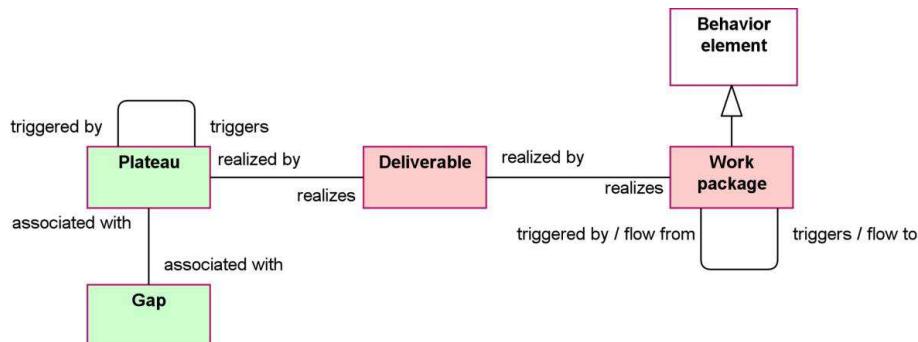


Figure B.15: Metamodelo de los conceptos de la extensión de implementación y migración[6].

ANEXO B. DESCRIPCIÓN CONCEPTUAL DE LA ARQUITECTURA125

Conceptualmente, un paquete de trabajo es similar a un proceso de negocio, este consiste en un grupo de tareas relacionadas con el objetivo de producir un resultado bien definido.

La Tabla B.6 muestra una vista general de los conceptos de migración e implementación, con sus definiciones.

Concepto	Descripción
Paquete de trabajo	Serie de acciones diseñadas para completar un objetivo único en un tiempo específico.
Liberable	Un resultado preciso y definido de un paquete de trabajo.
Meseta	Un estado relativamente estable de la arquitectura que existe durante un limitado periodo de tiempo.
Brecha	Un resultado del análisis de la brecha entre dos mesetas.

Table B.6: Conceptos de implementación y migración.

La Figura B.16 muestra cómo se relacionan los conceptos de implementación y migración con los conceptos base de ArchiMate.

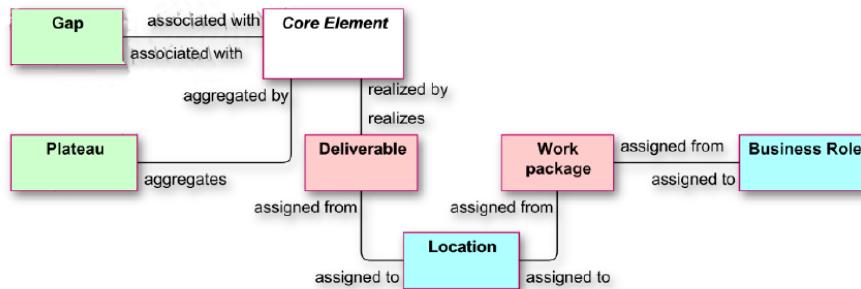


Figure B.16: Relación entre los conceptos base de ArchiMate y los de migración e implementación. [6].