Revista Tecnocientífica URU
Universidad Rafael Urdaneta
No. 19 Julio - Diciembre 2020
Depósito Legal: PPI 201402ZU4464
ISSN: 2343-6360

Desarrollo de un marco de trabajo con *node.js* basado en componentes para el manejo de solicitudes a objetos de negocio embebidos en el backend

Andrés Mármol y Jubert Perez

Escuela de Ingeniería de Computación. Facultad de Ingeniería.
Universidad Rafael Urdaneta. Maracaibo, Venezuela.

Correo electrónico: andresmarmolm@gmail.com y jubertperez@gmail.com

Recibido: 21-01-2020 Aceptado: 09-07-2020

Resumen

La presente investigación se realiza con el propósito de crear un marco de trabajo basado en componentes dentro del entorno de ejecución de JavacriptNode.js para la recepción y manejo de solicitudes a objetos de negocio embebidos en el *backend,* proveyendo una interfaz sobre la web que permite acceder a objetos de negocio con un *endpoint*único a través del cual usuarios agentes pueden ejecutar lógica de negocio. La investigación fue de tipo descriptiva y de diseño no experimental. La técnica de recolección de datos aplicada fue la observación directa, y la metodología de desarrollo fue el modelo cascada aplicando la técnica del Test Driven Development. Se logró desarrollar un marco de trabajo que permitiera definir objetos de negocio a ser embebidos en el *backend,* así como mecanismos de acceso a bases de datos relacionales para los propósitos de lógica interna o de lógica de negocio especificada por un programador usuario.

Palabras clave: Backend, Node.js, marco de trabajo, TDD.

Development of a component-based framework in node.js for the management of requests to business objects embedded in the backend

Abstract

This investigation was made with the purpose of creating a component-based framework inside the Javascript runtime Node.js for the reception and management of requests to business objects embedded in the backend, providing an interface over the web that allows accessing business objects with a single endpoint through which user agents can execute business logic. The investigation was of the descriptive type and non-experimental. The data collection technique applied was direct observation, and the development methodology was the cascade model, applying the Test Driven Development technique. The investigation achieved the development of a framework that allows the definition of business objects to be embedded in the backend, as well as mechanisms for relational database access for the purposes of logic internal to the framework or of business logic specified by the user.

Keywords: Backend, Node.js, framework, TDD

Introducción

Los marcos de trabajo son una de las herramientas más utilizada en el campo laboral que es el desarrollo de software, estas proveen a los desarrolladores una base fundamental sobre la cual fundamentar su lógica de negocio, tomando el código del desarrollador y abstrayendo el diseño e implementación de las estructuras básicas del desarrollo a través de componentes o código encapsulado que trabajan con el código del desarrollador.

Andrés Mármol y Jubert Pérez. ***CC BY 3.0 VE***
Revista Tecnocientífica URU, No. 19 Julio - Diciembre 2020 (21 - 28)

Tomando la importancia de los marcos de trabajo en mente, se desarrollará un marco de trabajo que permita abstraer a un desarrollador necesidades fundamentales de un *backend*¸ como lo es el manejo de peticiones a la lógica de negocio y el acceso a los datos, todo esto aplicando el paradigma de la programación basada en componentes, creando código encapsulado con interfaces bien definidas intercomunicadas para alcanzar este fin.

Los marcos de trabajo resultan imprescindibles en el desarrollo de soluciones de software hoy en día, dado que acortan el valioso tiempo de desarrollo, permitiendo alcanzar mayores niveles de productividad y menor tiempo para llevar aplicaciones a producción. Sin el uso de un marco de trabajo, el tiempo de desarrollo resulta ser alto, debido a que se debe dedicar tiempo para el desarrollo de estructuras base que soporten la lógica de negocio de una aplicación; un marco de trabajo elimina este paso del proceso de trabajo, permitiendo a un desarrollador empezar directamente por el orquestado de la lógica de negocio para implementar la aplicación deseada.

El problema al cual la presente investigación buscó solución fue la creación de un marco de trabajo el cual, siguiendo el paradigma de la programación orientada a componentes, resultara ser uno que permitiera la petición a lógica de negocio en el *backend* embebida en objetos de negocio, los cuales pueden tener acceso a la base de datos con una interfaz provista por el marco de trabajo la cual es configurada por el desarrollador usuario de éste. La investigación, entonces, se enfocó en crear una solución que abstrajera al desarrollador usuario del marco de trabajo el acceso a la base de datos en los objetos de negocio, así como el direccionamiento de peticiones a esta lógica almacenada en los objetos de negocio.

Fundamentos Teóricos

A continuación, se exponen las bases teóricas sobre las cuales se fundamentó la investigación:

Componente de software

Heineman y Councill[1] definen el concepto de un componente de software como “un elemento de software que conforma con un modelo de componentes y puede ser independientemente desplegado y compuesto sin modificación de acuerdo a un estándar de composición”. La más importante distinción entre componentes y objetos es que los componentes conforman a estándares definidos por una tecnología de componentes. Weinreich y Sametinger[2].

Programación basada en componentes

Según Clemente *et al.*, [3, Pág.109]:

> La programación basada en componentes consiste en la construcción de aplicaciones a través de un proceso de ensamblaje módulos de software independientes y reusables llamados componentes, donde los componentes son una unidad de composición de software con un conjunto de interfaces y un conjunto de requerimientos.

Arquitectura cliente-servidor

La arquitectura cliente servidor consiste de una red de computadoras en la cual muchos clientes (procesadores remotos) piden y reciben servicios de un servidor centralizado (computadora host). Las computadoras clientes proveen una interfaz para permitir al usuario realizar peticiones a servidores y desplegar los resultados de éstas. Los servidores esperan por peticiones de los clientes para poder brindarles los servicios que ofrecen. *Enciclopedia Británica*[4].

Javascript

Javascript es un lenguaje de programación ligero, interpretado o compilado justo-a-tiempo con funciones de primera clase. Javascript es un lenguaje dinámico, multi-paradigma y basado en prototipos, soportando estilos de programación orientado a objetos, imperativos y declarativos, *Mozilla Foundation*[5]. Javascript es el lenguaje que ejecuta el entorno a ser utilizado en el presente trabajo de investigación, Node.js

Node.js

Según el sitio oficial de Node.js[6], es un entorno de ejecución asíncrono y orientado a eventos de Javascript construido sobre el motor de Javascript V8 de Chrome, diseñado para construir aplicaciones de red escalables, Linux Foundation[7]. Node.js será el entorno donde se construirá el marco de trabajo del presente trabajo de investigación, ya que este ofrece flexibilidad para la creación de servicios genéricos que ofrecerá el marco de trabajo.

Framework (marco de trabajo)

Según Crnkovic, et al.,[8] se han dado varias definiciones de marco de trabajo a lo largo del tiempo, acercándose al concepto desde diferentes puntos de vista y diferentes niveles de detalle; afirman que un marco de trabajo puede ser visto como un diseño reusable de un sistema, donde el diseño consiste de la representación de clases abstractas y la interacción de las distintas instancias de estos, también describen los marcos de trabajos de componentes, como una "tarjeta de circuito" con espacios vacíos donde se insertan los componentes para crear una instancia funcional; en estos los componentes se insertan en tiempo de ejecución.

Metodología de la Investigación

La presente investigación se define como una de tipo descriptiva dado que consiste en la especificación de las propiedades, características y rasgos importantes de un conjunto homogéneo de fenómenos, dejando manifestada la estructura o comportamiento de aquello que se estudia. Además, la investigación además no busca la verificación de una hipótesis, sino de la descripción de hechos a partir de un criterio definido previamente. Así mismo, la descripción se define como del tipo proyectiva, dado que busca definir cómo se debería comportar la solución final.

El diseño de investigación es de campo, dado que consiste en la recolección de datos directamente de aquello que se investiga, sin manipular variable alguna que intervenga en el fenómeno investigado. En el contexto de la investigación, este diseño encaja dado que consistió en la recolección de datos del diseño, modelado y construcción del marco de trabajo, así como de su comportamiento final definido por las fases anteriormente mencionadas.

La metodología de desarrollo utilizada fue el modelo cascada, que consiste en una serie simple y lineal de pasos, utilizada cuando los requerimientos del software a desarrollar están bien definidos. Consiste en un flujo desde la especificación, al diseño, construcción y entrega del proyecto a desarrollar, siendo este un proceso lineal y no iterativo de desarrollo.

Resultados

En esta investigación se llevaron a cabo las fases definidas por el modelo mencionado en la sección anterior, siendo estas la comunicación, planeación, modelado, construcción y despliegue.

Andrés Mármol y Jubert Pérez. ***CC BY 3.0 VE***
Revista Tecnocientífica URU, No. 19 Julio - Diciembre 2020 (21 - 28)

Fase de comunicación

Tomando en cuenta los objetivos definidos, se prosiguió a definir los requerimientos del marco de trabajo, estos se exponen en la Figura 1 que se presenta a continuación:

Permitir el registro de objetos de negocio
Manejar las peticiones a los objetos de negocio
Brindar funcionalidad para la autenticación de usuarios
Permitir la autorización de usuarios para la ejecución de métodos
Exponer funcionalidad para la configuración para el acceso a bases de datos relacionales
Permitir a los objetos de negocio la consulta a las base de datos configuradas

Figura 1. Requerimientos del marco de trabajo

Fase de planeación

Una vez definidos los requerimientos, se realizó la planificación de las actividades a realizar (Ver Figura 2), siguiendo el esquema lineal definido por el modelo cascada. Se definió la secuencia de actividades, así como el tiempo estimado de culminación de cada actividad; la planificación, expuesta en la forma de un diagrama de Gantt, se expone a continuación:

Figura 2. Planificación de actividades para desarrollar el marco de trabajo

Fase de modelado

En esta fase de realizó la arquitectura de la solución, definiendo concretamente el alcance del marco de trabajo, así como los mecanismos fundamentales que implementará éste, como el protocolo de capa de aplicación a utilizar (HTTP/HTTPS), la necesidad de realizar la creación y registro de objetos de negocio, la implementación de la permisología, y la manera en la que las peticiones a los objetos de negocio se realizarán, siguiendo el formato definido en la Figura 3.

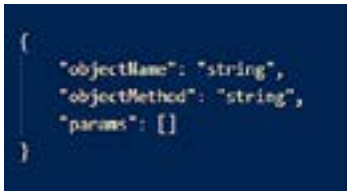

Figura 3. Formato de las peticiones al marco de trabajo

A continuación, se definió la arquitectura basada en componentes del marco de trabajo, definiendo el comportamiento de los componentes que conformarán éste, así como las interfaces que estos proveen entre sí para lograr la funcionalidad deseada. En la Figura 4 expuesta a continuación se muestra el diagrama de componentes del marco de trabajo.

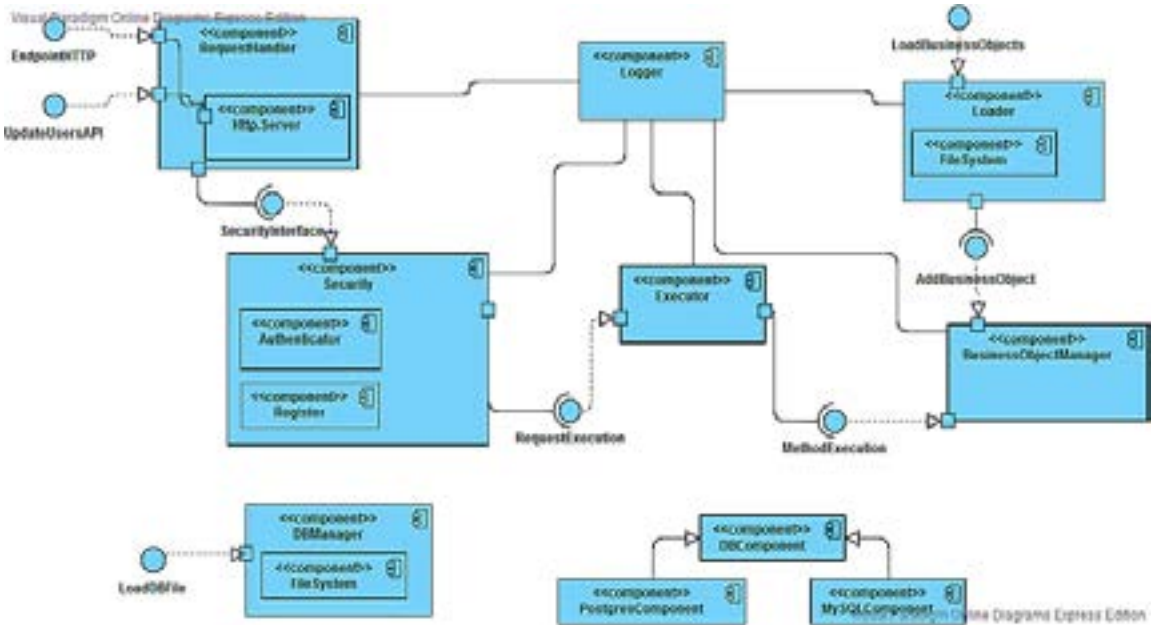

Figura 4. Diagrama de componentes del marco de trabajo

El marco de trabajo consiste en varios componentes, empezando con el componente RequestHandler, el cual está compuesto de un servidor HTTP y se encarga de recibir las peticiones y dirigirlas al componente Security, el cual se encarga de realizar la autenticación y autorización para el acceso a los objetos de negocio (la permisología), dirigiendo al componente Executor las peticiones válidas ya autorizadas.

En cuanto al registro de objetos de negocio, se tiene el componente Loader, el cual toma clases Javascript que implementan métodos que definen la lógica de negocio deseada, y los carga al Business Object Manager quien es el que se encargará de la instanciación de la lógica de negocio definida en los métodos a través de las peticiones válidas y autorizadas que lleguen a través del componente Executor.

Hay que destacar que el marco de trabajo implementa un esquema sencillo de inyección de dependencias, donde las clases definidas por el usuario desarrollador pueden optar por una conexión a la base de datos según la definición del constructor del objeto de negocio, el cual puede desconstruir el objeto pasado por el Business Object Manager y obtener un objeto *db*, el cual posee métodos definidos por la clase abstracta DBComponent para el acceso a la base de datos, los objetos de negocio, dependiendo de la configuración, utilizan un PostgresComponent o un MySQLComponent. Los métodos expuestos se presentan en la figura 5 la cual se presenta a continuación.

Andrés Mármol y Jubert Pérez. ***CC BY 3.0 VE***
Revista Tecnocientífica URU, No. 19 Julio - Diciembre 2020 (21 - 28)

```
module.exports = class DBComponent {
    constructor(){
        if(this.constructor === DBComponent){
            throw new Error('DBComponent is an abstract class')
        }
    }

    query(sql, params, callback){
        throw new Error("Can't call abstract method 'query' on DBComponent ")
    }

    queryWithoutParams(sql, callback){
        throw new Error("Can't call abstract method 'queryWithoutParmas' on DBComponent ")
    }

    async stop(){
        throw new Error("Can't call abstract method 'stop' on DBComponent ")
    }
}
```

Figura 5. Clase abstracta BDComponent

En cuanto a la permisología, se realizó un modelado de base de datos (ver figura 6) que permite relacionar usuarios con roles a métodos y sus objetos, este modelado se expone a continuación:

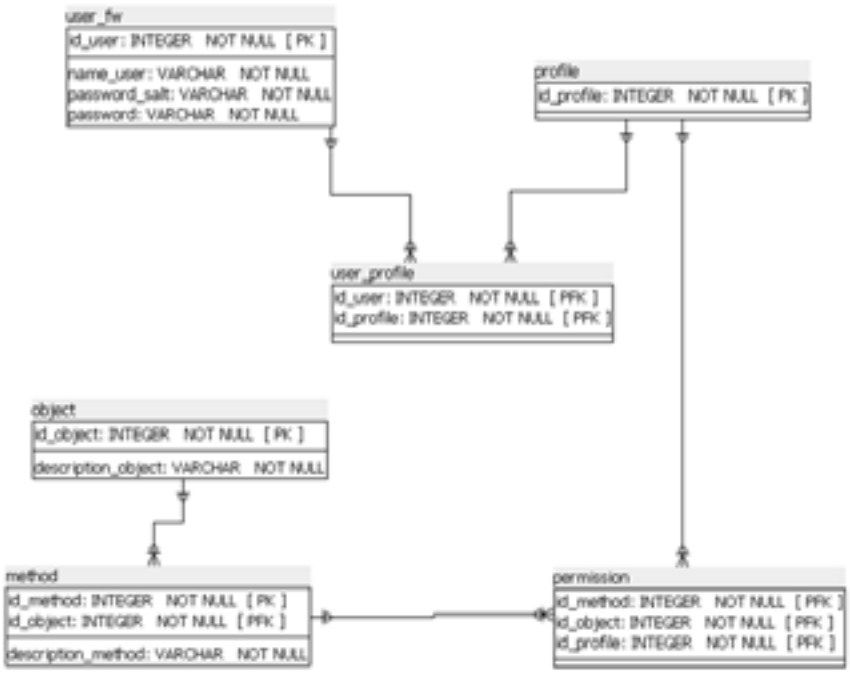

Figura 6. Modelado para la permisología

Fase de construcción

Se realizó la codificación de lo definido en la fase de modelado, empezando por el componente RequestHandler, el cual consiste en la creación de un endpoint HTTP utilizando las virtudes del entorno Node.JS utilizado, utilizando el marco de trabajo Express para facilitar el manejo de solicitudes a través de los objetos de petición y respuesta HTTP, así como el parseo de los bodies JSON que siguen el formato

definido anteriormente. El endpoint provee cuatro rutas, la más importante siendo la ruta / POST, la cual es la que recibe las peticiones para la ejecución de la lógica de negocio dado que el body de la petición siga el formato definido anteriormente.

Se codificó la configuración para la base de datos; el marco de trabajo expone interfaces para configurar la base de datos a utilizar utilizando un archivo .json o un objeto plano de javscript que contiene la información necesaria (driver, host, puerto, nombre de base de datos, etc.). El marco de trabajo entonces utiliza esta información para instanciar hijos de la clase DBComponent con el componente DBManager e inyectarlos donde sea necesario (ya sea para componentes inherentes al marco de trabajo o a objetos de negocio).

Para el proceso de autenticación y autorización, el componente Security recibe las peticiones que llegan a RequestHandler y autentica o autoriza según la petición realizada, utilizando la librería Passport.js para facilitar el proceso de autenticación y el modelo relacional definido para la permisología para realizar la autorización, para luego redirigir las peticiones aceptadas al componente Executor.

Para el registro de objetos de negocio, se crean las clases Javascript que implementen la lógica necesaria y se registran con el componente Loader, el cual las pasa al componente Business Object Manager, quien es el que finalmente crea instancias Singleton de estos objetos de negocio para ejecutar las peticiones que se reciban del componente Executor. El Business Object Manager, utilizando el DBManager, también se encarga de pasar a los objetos de negocio instancias de DBComponent para que estos puedan tener acceso a la base de datos configurada.

Fase de despliegue

Definimos en la fase de despliegue la realización de las pruebas, se realizaron pruebas unitarias (ver Figura 7) para verificar la funcionalidad de los componentes individuales, así como pruebas de estrés para verificar el comportamiento del marco de trabajo en condiciones de estrés utilizando un contenedor Docker limitado en capacidad de procesamiento y memoria.

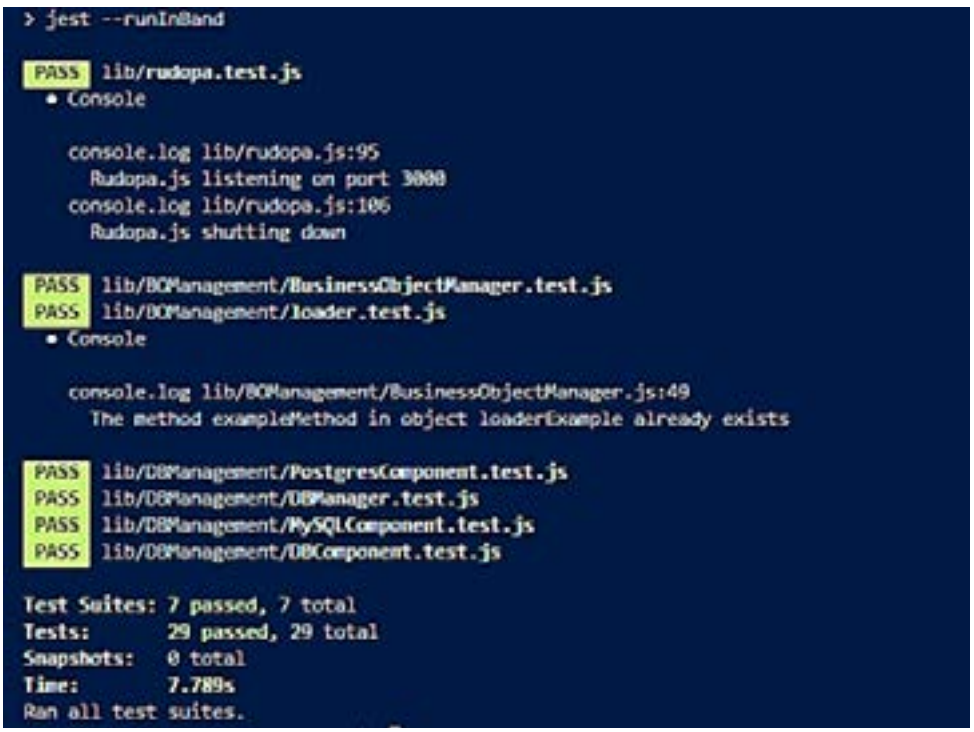

Figura 7. Extracto de ejecución de las pruebas unitarias.

Andrés Mármol y Jubert Pérez. ***CC BY 3.0 VE***
Revista Tecnocientífica URU, No. 19 Julio - Diciembre 2020 (21 - 28)

Conclusiones

Este trabajo de investigación tuvo como fruto un marco de trabajo orientado a la creación de una forma sencilla de acceder a objetos de negocio, donde toda la lógica y autenticación reside en un solo punto del *backend* fácilmente accesible a través de la tecnología web que es HTTP, y donde se brindan a estos objetos de negocio fácil acceso a la base de datos, siendo transparente con el usuario programador en cuanto a qué manejador de base de datos relacional se utiliza.

El entorno de ejecución Node.js resultó ser una gran herramienta para este marco de trabajo desarrollado; este proveyó un ambiente ideal para la creación de un marco de trabajo que ofrece sus servicios para crear aplicaciones web, ya que con él se pudieron utilizar varias herramientas que aceleraron sino mejoraron el resultado final de esta investigación, como lo fueron las características para la creación de clases que representaban los componentes diseñados, funcionalidad para la creación, mantenimiento y terminación de conexiones HTTP o a base de datos, entre otras grandes bondades del entorno.

Alcanzados estos objetivos, se puede concluir que se creó un marco de trabajo sencillo, que actúa como una alternativa a otros patrones o estilos arquitectónicos vistos hoy en día en la comunidad del desarrollo web; una alternativa que provee un punto único y sencillo de acceso a la lógica definida por el programador, en comparación a interfaces engorrosas de dudosa utilidad en el contexto del buen desarrollo de software.

Aunque se alcanzaron los objetivos y los requerimientos delimitados, durante el desarrollo y culminación de este trabajo de investigación se hallaron puntos en los cuales el marco de trabajo desarrollado puede crecer y fortalecerse como una mejor herramienta de desarrollo de aplicaciones.

Referencias Bibliográficas

[1] Heineman, G. y Councill, W. Component-Based Software Engineering, Addison-Wesley: Nueva York, Estados Unidos de América, (2001)

[2] Weinreich, R. y Sametinger, J. Component Models and Component Services – Concepts and Principles. Addison-Wesley: Boston, Estados Unidos de América, (2001).

[3] Clemente P., Hernández J., Murillo J., Pérez M. y Sánchez F. "Component-based System Design and Composition: And Aspect-oriented Approach". Series on Component based software development. Component based software development: Case Studies Kung-Kiu Lau, pp.109-128, World Scientific Publishing: Manchester, Reino Unido, (2003).

[4] Britannica, The Editors of Encyclopaedia. "Server". En Encyclopedia Britannica, 5 Apr. 2018, Disponible en: https://www.britannica.com/technology/server. [4 May 2020].

[5] Mozilla Foundation. HTTP. (2019). Disponible en: https://developer.mozilla.org/en-US/docs/Web/HTTP [4 May 2020].

[6] OpenJS Foundation. Node.js. (s.f.). Disponible en: https://nodejs.org/es/ [4 May 2020].

[7] Linux Foundation Node.js. (2019). Recuperado de https://nodejs.org/en/ [4 May 2020].

[8] Crnkovic I., Hnich B., Jonsson T. y Zeynep K. Building Reliable Component-Based Software Systems. En Crnkovic I. y Larsson M. (Eds.), Basic Concepts in CBSE.Norwood, Estados Unidos de América. ArtechHouse.(2002).pp.53-54.