ENTORNO VIRTUAL PARA LA FORMACIÓN DE TECNÓLOGOS E INGENIEROS DE SISTEMAS EN PATRONES DE DISEÑO DE SOFTWARE[1]

VIRTUAL ENVIRONMENT FOR TRAINING TECHNOLOGISTS AND SYSTEMS ENGINEERS IN SOFTWARE DESIGN PATTERNS

Ramiro Andrés Bedoya Escobar y Tania Isadora Mora Pedreros
Grupo de Investigación GRINTIC
Institución Universitaria Antonio José Camacho
Recibido: 15/03/2021 Aprobado: 25/05/2021
Cómo citar este artículo:
Bedoya Escobar, R.A. y Mora Pedreros, T.I.(2021). Entorno virtual para la formación de tecnólogos e ingenieros de sistemas en patrones de diseño de software. *Revista Sapientía, 13(25)*, 69 - 77.

RESUMEN

El desconocimiento de los patrones de diseño es un problema común entre los desarrolladores de software, causando que muchas veces se tenga que reinventar la rueda cuando el desarrollador se enfrenta a problemas de diseño de diferente índole, factor que tiene importantes efectos en el tiempo y en los costos de los proyectos. El objetivo de este trabajo es buscar que en la Institución Universitaria Antonio José Camacho (UNIAJC), en su programa de Ingeniería en Sistemas, se fortalezcan los cursos de programación con el aprendizaje de patrones de diseño por medio de guías y laboratorios técnicos brindados por medio de la plataforma UNIAJC Virtual.

Usando una metodología clásica de investigación basada en la exploración y análisis de los patrones actuales, se seleccionaron nueve patrones de desarrollo que han sido divididos en tres niveles: básico, intermedio y avanzado, organizados de esta forma con el objetivo de que puedan asociarse al nivel de progreso del estudiante con las asignaturas de programación y gradualmente aprender técnicas de diseño que le permitirán tener un lenguaje común para una comunicación más efectiva y un diseño de software más sofisticado.

PALABRAS CLAVE

Patrones de Diseño, GOF (The Gang of Four), UNIAJC Virtual, Javascript, ES6.

ABSTRACT

The Lack of knowledge of design patterns is a common problem among software developers, causing the wheel to have to be reinvented many times when the developer faces different kinds of design problems, a factor that has important effects over time and time. project costs. The objective of this work is to seek that in the Antonio José Camacho University Institution (UNIAJC), in its Systems Engineering program, programming courses are strengthened with the learning of design patterns through guides and technical laboratories provided through of the UNIAJC Virtual platform.

---

[1] Este artículo se deriva del trabajo de grado del egresado de Ingeniería en Sistemas, Ramiro Andrés Bedoya Escobar, titulado Entorno virtual para la formación de tecnólogos e ingenieros de sistemas en patrones de diseño de software, el cual obtuvo calificación meritoria en la evaluación de los jurados.

Using a classic research methodology based on the exploration and analysis of current patterns, nine development patterns were selected that have been divided into three levels: basic, intermediate and advanced, organized in this way with the objective that they can be associated with the level of student progress with programming subjects and gradually learn design techniques that will allow them to have a common language for more effective communication and more sophisticated software design.

KEYWORDS

Design Patterns, GOF (The Gang of Four), UNIAJC Virtual, Javascript, ES6

INTRODUCCIÓN

El presente trabajo tiene el propósito de explicar a los estudiantes de Ingeniería en Sistemas de la Institución Universitaria Antonio José Camacho (UNIAJC) los patrones de diseño de software. En tal sentido, se expondrán nueve patrones de diseño seleccionados del clásico libro Gang of Four (Gamma et al., 1995), los cuales constituyen un conjunto de soluciones típicas a problemas frecuentes en el diseño de software. Estas soluciones son agnósticas a un lenguaje de programación, los patrones no son simplemente código que puede ser copiado y pegado, sino que el objetivo es entender el concepto y ajustar el patrón al problema que estás intentando resolver en tu código.

Facilitar el estudio de los patrones de diseño por medio de ejemplos del mundo real y diagramas contextualizados, permitirá que el estudiante o Ingeniero en Sistemas aprenda a resolver problemas de diseño de software de una forma más rápida y lograr mejorar sus habilidades; acto seguido, mejorará su proceso de adaptación al ámbito laboral.

La propuesta permitirá dividir los contenidos en 3 niveles: básico, intermedio y avanzado, de acuerdo con la separación propuesta en el libro Ganf of Four (Gamma et al, 1995). En cada nivel se explicarán tres patrones. En el nivel básico se expondrán patrones creacionales cuyo fin es proveer mecanismos para la creación de objetos de una forma flexible y reusable; los patrones a exponer son: factory method, builder y abstract factory. En el nivel intermedio se expondrán los patrones estructurales para explicar cómo se pueden crear largas estructuras de objetos y clases de forma flexible y eficiente; los patrones a exponer son: composite, decorator y façade. Y finalmente, en un nivel avanzado, se expondrán los patrones de comportamiento en el que se hace hincapié en tener una mejor comunicación y asignación de responsabilidades entre objetos; los patrones a exponer son: *observer, state y strategy*.

A fin de crear contenidos virtuales relevantes para el lector, los patrones presentados se explicarán usando el lenguaje de programación Javascript a partir de la especificación ES6, que permite tener funcionalidades útiles de la programación orientada a objetos, como lo son declaración de clases, importación y exportación de módulos usando import y export. Además de ser el lenguaje que entienden los navegadores (siendo una de sus principales características), también puede ser usado en el servidor para la creación de servicios web. Por tal razón, se seleccionaron patrones que podrían ser más relevantes para el entorno web (frontend), ayudando a estructurar mejor este tipo de aplicaciones que hoy en día están teniendo más lógica de negocio debido a la popularización de aplicaciones de una sola página (SPA) usando *frameworks como react y angular* (Angular, 2020; React, 2021).

MARCO TEÓRICO

Modalidades de Educación Virtual

B-learning: esta modalidad de estudio, también conocida como aprendizaje semipresencial o blended learning, es una mezcla entre educación presencial (en un aula de clases) y del trabajo en línea (Usando las TIC).

Moreira et al (2010) (Moreira et al., 2010) propone tres modelos que nos permiten observar cómo un modelo de b-learning puede fortalecer la educación presencial. Estos son:

Modelo de docencia presencial con internet: usa como recursos de apoyo o complemento el aula virtual. Usualmente, el docente sigue con sus clases presenciales y actividades de forma presencial, pero utiliza el aula virtual como medio para transmitir información, publicación de documentos, apuntes, calificaciones o exámenes. La comunicación entre el docente y el estudiante en la plataforma no es habitual.

Modelo de docencia semipresencial: en este modelo, el aula virtual comienza a tener más protagonismo. El docente comienza a crear actividades y elaborar materiales para que sus alumnos formulen preguntas, abran debates, planeen trabajos, entre otras actividades de forma virtual. Todo esto fuera del contexto de la clase tradicional.

E-learning: el termino e-learning, derivado del término en inglés electronic learning, refiere al aprendizaje online usando las TIC (tecnologías de la información y comunicación). Usualmente es conocido también como Educación en Línea. Esta metodología de estudio es 100% virtual, convirtiéndose en la opción perfecta para quien no se puede desplazar o, por falta de tiempo, no puede acudir a una institución de forma presencial (Ganduxé, 2018). El contenido que se presenta sobre el e-learning puede ser dado en múltiples formatos, estos pueden ser: videos, multimedia, pdf, entre otros, que puedan ayudar a mejorar el entendimiento del tema a estudiar.

LMS en educación

El e-learning requiere una plataforma de aprendizaje virtual, conocida como LMS, espacio donde los estudiantes desarrollan el curso. LMS (Easy LMS, 2021), acrónimo de Learning Management System, que en español refiere a los Sistemas de gestión de aprendizaje (Easy LMS, 2021). Una de sus principales características es brindar materiales de capacitación a los participantes de los contenidos virtuales registrados; así como gestionar todo el proceso de flujo de los contenidos virtuales, como: participantes, comunicación entre los alumnos y docentes, progreso de aprendizaje, resultados y poder medir la efectividad.

Actualmente existen diferentes tipos de LMS:

LMS de código abierto: al ser de código abierto, estos LMS son mantenidos por una comunidad de desarrollo o empresas interesadas. No tienen costo para su uso, entre los más conocidos están: Moodle, Chamilo y OpenEdx. Es posible encontrar más en el artículo de Pappas (2015).

LMS de software propietario: LMS son plataformas de paga y se paga una licencia para su uso. Solo la empresa creadora puede realizar modificaciones al software y controlar las funcionalidades que se van desarrollando. Adicionalmente, una de las características importantes en este tipo de plataformas es la posibilidad de obtener soporte técnico y una empresa robusta que lo respalde. Entre las opciones más conocidas son: Blackboard, Educativa y Edmodo.

LMS desarrollados a la medida: LMS son software propietario creados a la medida para una organización o empresa que tenía requerimientos muy específicos de funcionalidad.

La Institución Universitaria Antonio José Camacho (UNIAJC) usa Moodle como LMS. Esta plataforma, al ser código Open-Source (GNU, 2007), es altamente configurable e instalable on-premise, una de las más usadas en el mundo académico. Actualmente cuenta con más de 200 millones de usuarios en todo el mundo y ha sido traducida a 120 idiomas. Para la UNIAJC **Moodle** es una herramienta de apoyo tecnológico mediadora entre estudiantes y docentes (tutores) de las asignaturas en modalidades de B-learning y E-learning, que hace posible que los estudiantes logren su aprendizaje a través de actividades asincrónicas conforme a su disponibilidad de tiempo y dedicación a las actividades académicas, pero también contiene elementos con los cuales los docentes apoyan su aprendizaje como foros, wikis y chat sincrónicos.

EDUCACIÓN VIRTUAL EN EL DESARROLLO DE SOFTWARE

En las últimas décadas, la educación virtual ha aportado un gran valor a la educación si se toma como

ejemplo la Institución varios de sus programas desarrollados van guiados a través de la educación virtual, en la perspectiva de los autores, esto ha incidido aún más en el desarrollo de software y la programación. Muchas de las actividades que usualmente podían ser 100% presenciales, se han combinado con metodologías de e-learning permitiendo que a estudiantes con limitaciones en tiempo y ubicación geográfica se les permita continuar con su proceso de estudio tomando el control de su propio aprendizaje, debido a que no existe hasta el momento algún apoyo tecnológico para estudiantes sobre el diseño de patrones de software, siendo este parte fundamental dentro de su crecimiento profesional, se desarrolla esta temática en particular.

Para sustentar lo anterior, se encontró un estudio publicado en la Universidad de Hong Kong, que implementó una metodología de enseñanza llamada OBTL (resultados basados en la enseñanza y el aprendizaje), soportada con b-learning para la carrera de ciencias de la computación (Wang et al., 2007). Los investigadores hicieron uso de las plataformas de enseñanza y actualizaron el curso de programación básica logrando mejorar el índice de excelencia de estudiantes, reduciendo la tasa de personas que perdían la materia de una forma significativa con respecto a años anteriores y mejorando la orientación tradicional que tenían los cursos. Los estudiantes evaluaron la estructura del curso por medio de cuestionarios y entrevistas, y todos coincidieron que la combinación entre estos dos modelos ayuda a aprender programación de computadores de una forma más efectiva.

Otro caso de éxito que ha sido implementado en la Universidad de Afyon Kocatepe University, en Turquía, está relacionado con el desarrollo de software (Deperlioglu y Köse, 2013). En esta investigación, usando la combinación de e-learning y educación presencial, rediseñaron el curso de Estructuras de datos y algoritmos. Los estudiantes realizaban actividades anunciadas por el docente en la plataforma de la universidad llamada @KU-UZEM, que es un LMS diseñado por ellos. El curso contaba con varios componentes como: comunicación en línea, recursos adicionales para la materia y evaluaciones. Los resultados del ejercicio fueron exitosos, logrando que los estudiantes del curso pasaran la materia en el primer año en el que se implementó, y el promedio del curso fue mejorado considerablemente.

METODOLOGÍA

Para el desarrollo de este proyecto y lograr los objetivos planteados, la metodología se compone de cuatro fases: exploración, análisis, desarrollo y publicación.

1. Exploración: en esta etapa se quiere identificar los patrones de diseño acordes para cada nivel de formación y poder realizar una curvatura de aprendizaje conforme a los niveles: básico, intermedio y avanzado.

- Para esta tarea se hizo un estudio de cuáles son los libros más relevantes en el ámbito de patrones de diseño y las plataformas en línea con el contenido más completo sobre patrones de diseño.

2. Análisis: analizar el patrón de diseño seleccionado y contextualizarlo en un entorno arquitectónico y con un escenario de calidad. Para determinar si el laboratorio que se realizará será lo suficientemente completo para su entorno.

- Adicionalmente, se revisaron todos los patrones tradicionales del libro Design patterns: Elements of reusable object-oriented software (Gamma et al., 1995) y, haciendo un análisis con diferentes fuentes, se determinó cuáles eran los patrones más frecuentes para el lenguaje de programación (JavaScript, 2021).

3. Desarrollo: desarrollar y representar el laboratorio del patrón adaptado a los lineamientos dados por el área UNIAJC Virtual y definir si requiere elementos necesarios para complementar la arquitectura del patrón. A cada patrón a desarrollar se le determinaron 6 partes importantes para completar su desarrollo:

- ¿Por qué necesitamos el patrón?: hace una breve explicación de la necesidad del patrón.
- En definición, ¿qué es el patrón?: presenta la definición encontrada en la literatura.
- El problema en un ejemplo: expone un problema de la vida real, cuál sería el problema si no se usa el patrón y cómo podría resolverse usándolo.
- Estructura del patrón: presenta un diagrama formal del patrón contextualizado con las clases del ejemplo anterior y cada uno de los integrantes que componen el patrón de diseño.
- Conclusiones: hace un resumen puntual del patrón.
- Ejercicio a resolver: incluye el planteamiento de un problema, una estructura inicial de código y la solución esperada.

Publicación: publicar los laboratorios de acuerdo con los niveles de formación determinados (básico, intermedio y avanzado), y ajustarlos a un esquema de curso para la plataforma UNIAJC Virtual y la sintaxis de escritura de artículos del Moodle.

- Para la creación de los contenidos virtuales en la plataforma UNIAJC Virtual, se debe de presentar la ficha "FORMATO DISEÑO Y DESARROLLO DE VIRTUALIZACIÓN" diligenciada con el contenido del curso a publicar.

RESULTADO Y CONCLUSIONES

En este curso se han desarrollado nueve patrones de desarrollo, aquí se presentará uno de los patrones de desarrollo del curso ilustrando las evidencias de la publicación en UNIAJCVirtual.

PATRONES AVANZADOS

Evidencia del curso en UNIAJCVirtual – Nivel Avanzado

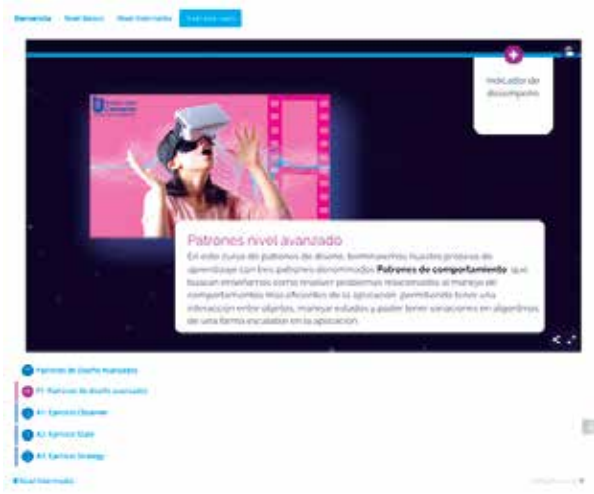

Fig. 1. Evidencia curso en UNIAJCVirtual - Nivel Avanzado
Fuente: Elaboración Propia

OBSERVER DESIGN PATTERN

El patrón de diseño **Observer** es usado para ofrecer un modelo de suscripción en el que los objetos se suscriben a un evento y obtienen una notificación cuando el evento ocurre. Esto se puede realizar dinámicamente en el tiempo de ejecución, permitiendo al usuario tener una suscripción y eliminar la suscripción a eventos cuando se necesite.

¿POR QUÉ NECESITAMOS EL OBSERVER?

Imagine que está trabajando en un proyecto que contiene dos tipos de objetos, el Cliente y la Tienda. Los clientes están muy interesados en los nuevos productos que puede traer la tienda, en especial en sus últimos lanzamientos.

Los clientes pueden visitar la Tienda todos los días para preguntar si ya llegó el nuevo iPhone, pero hay un retraso en las entregas y en la tienda les notifica a los clientes que no está disponible.

Una posible solución es que la tienda envíe cientos de correos electrónicos (en muchos casos, considerados como spam) a todos los clientes siempre que un nuevo producto llegue a la tienda. Esto podría evitar que el cliente perdiera su visita a la tienda, aunque esto podría molestar a los clientes que no están interesados en los nuevos productos.

Estas soluciones pueden ser ineficientes al causar que el cliente pierda tiempo valioso o la tienda gaste recursos notificando al cliente que no está en realidad interesado.

Aquí es donde el patrón de diseño Observer entra en acción al sugerir agregar un mecanismo de suscripción y des-suscripción, para estar informados de las novedades, como cambios en las propiedades de algún objeto. Estas notificaciones deben tener información útil sobre el evento, para tener idea quien lo está desencadenando y cuáles son los argumentos de este.

EN DEFINICIÓN, ¿QUÉ ES EL PATRÓN OBSERVER?

Un Observer es un objeto que desea ser informado sobre eventos que están ocurriendo en el sistema. La generación de la entidad ocurre cuando es un Observable.

EL PROBLEMA EN UN EJEMPLO

Dando continuidad al ejemplo pasado, supongamos que tenemos una tienda física y queremos que esta tienda tenga un mecanismo para suscribir a los clientes que estén interesados en el nuevo iPhone. Usando el patrón Observer, se implementará una solución bastante práctica que ayudará con el problema que tiene la tienda.

Vamos a crear la clase Event, en esta clase padre, vamos a usar toda la lógica relacionada con las notificaciones y el manejo de los observadores o también llamadas 'Personas interesadas'.

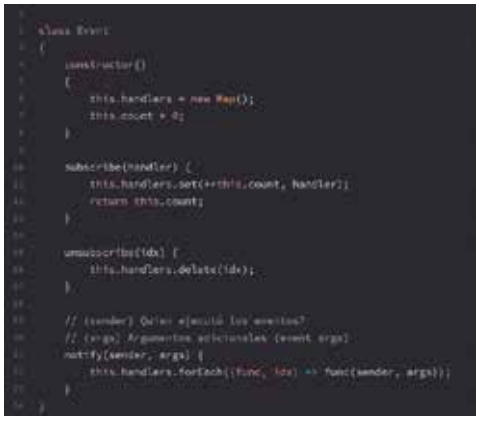

Fig. 2. Observer – Clase Event2
Fuente: Elaboración Propia

En este caso, se va a usar una lista tipo Map, que permite manejar los arrays de tipo clave-valor. Adicionalmente se declarará una variable count para realizar el contador del array.

El primer método a implementar es el subscribe, cuyo parámetro será el evento para ejecutar (la función callback). El segundo método es el unsubscribe que nos permite des-suscribirnos de los eventos cuando ya no estamos interesados. El tercer método es el notify, que será el encargado de ejecutar las funciones de los observadores que han sido suscritas por medio de una iteración. Este método recibe dos parámetros, el sender que es la clase quien ejecutó el evento y los args que son los argumentos o datos adicionales. Ahora, se implementa la tienda usando la clase padre de Event.

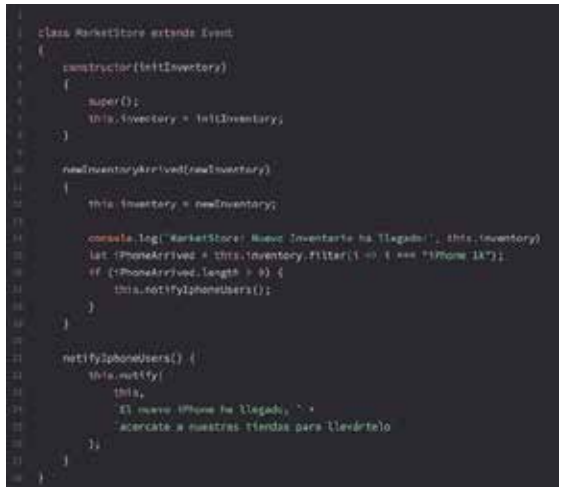

Fig. 3. Observer – Clase MarketStore
Fuente: Elaboración Propia

En la clase MarketStore es posible observar lo siguiente: primero se realiza un extends de la clase Event y en el constructor se define el inventario inicial.

Se crea una función llamada newInventoryArrive que permite cargar nuevo inventario a la tienda y este realizará una validación interna que permitirá determinar si ha llegado el nuevo **iPhone 1k,** para posteriormente notificar a los usuarios inscritos por medio de la función this.notify de la clase padre Event. En esta función se enviará el sender, que sería la clase actual, y algunos argumentos.

Por último se crea la clase Person, que será el observador. Esta clase es bastante sencilla, pero tendrá la función update que se observa a continuación.

```
class Person
{
    constructor(name)
    {
        this.name = name;
    }

    update = (sender, args) =>
    {
        console.log(`Una notificación de ${sender.name} ha llegado!`);
        console.log(`${this.name}: ${args}`);
    }
}
```

Fig. 4. Observer – Clase Person
Fuente: Elaboración Propia

Hay dos aspectos a tener en cuenta aquí. Se pasará por parámetros en el constructor el nombre que será usado posteriormente y se ha definido la función update como un arrow function, debido a que es necesario que el this sea el mismo de la clase Person, esto para poder obtener el this.name de la instancia actual.

Todo está listo para poder utilizar estas clases. La implementación se hará de la siguiente manera:

```
// Observador
let paul = new Person("Paul McCartney")

// Subject
let store = new MarketStore([]);
store.name = "Los Magníficos Store";
let paulId = store.subscribe(paul.update);
// Nuevo Suscriptor Map [ 1 => [Function: update] ]

store.newInventoryArrive(["iPhone 14", "iPad pro"]);
// MarketStore: Nuevo Inventario ha llegado: [ 'iPhone 14', 'iPad pro' ]
/* Llega la notificación */
// Una notificación de Los Magníficos Store ha llegado!
// Paul McCartney: El nuevo iPhone ha llegado, acercate a nuestras tiendas para llevártelo
store.unsubscribe(paulId)

store.newInventoryArrive(["iPhone 14", "Magic Mouse"]);
// MarketStore: Nuevo Inventario ha llegado: [ 'iPhone 14', 'Magic Mouse' ]
/* No han llegado más notificaciones */
```

Fig. 5. Observer - Implementación
Fuente: Elaboración Propia

Aquí se observa que el cliente, cuando ha comprado el nuevo teléfono, ya deja de estar interesado y llama el método store.unsubscribe(paulId). Así que cuando llega nuevo inventario con el nuevo teléfono, al cliente no le llegarán nuevas notificaciones.

Código completo del ejemplo: https://bit.ly/3cEUlth

ESTRUCTURA DEL PATRÓN OBSERVER

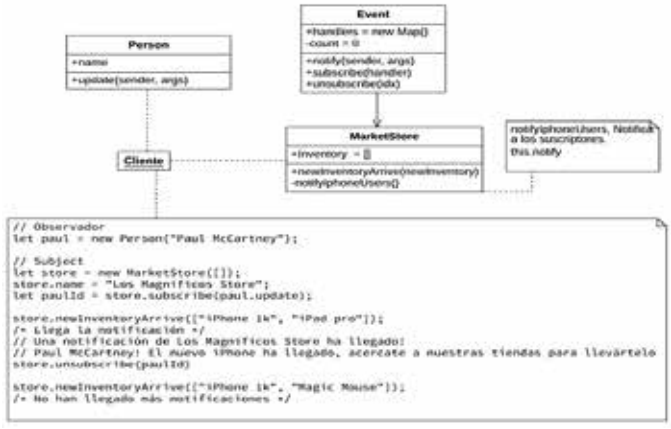

Fig. 6. Patrón Observer – Diagrama de clases
Fuente: Elaboración Propia

En este patrón se encuentran varios componentes que interactuarán entre sí.

1. El Subject, en el ejemplo son las clases MarketStore cuyo padre es Event, es el encargado de mantener la lista de los observers u observadores, y enviarles notificaciones cuando el estado cambia. Cualquier número de observadores puede observar un Subject.

2. Los Observers, en el ejemplo es la clase Person, básicamente es el objeto que tiene la función que será invocada cuando el evento del Subject ocurra.

El patrón de diseño observer es usado cuando el cambio del estado de un objeto requiere cambiar otros objetos.

Este patrón debe ser usado cuando se requiere observar algunos objetos importantes en una aplicación, pero por tiempo limitado o en específicos casos, debido a que cuenta con un sistema de suscripción dinámico. Los suscriptores u observadores pueden entrar e irse de la lista cuando ellos lo requieran y en tiempo de ejecución.

Este patrón permite aplicar el principio de diseño Open/Close Principle, en el que se pueden introducir nuevas clases de suscriptores (Observers) sin cambiar el código del publicador (Subject). Este patrón es de gran utilidad cuando se requiere partir una aplicación grande

en pequeñas partes, que no esté tan acoplada, permitiendo que el código sea reutilizable.

CONCLUSIONES

Fue posible comprobar que la información sobre los patrones de diseño es amplia y basta, hay un sin número de guías y libros en diversos lenguajes de programación. Al principio puede ser un poco abrumador tratar de entender patrones, en especial si se intenta aplicarlos a Javascript, que es un lenguaje tan dinámico. Por tal razón, fue importante elaborar una selección de patrones que encajara en este lenguaje de programación permitiendo usar características propias de la última especificación. Las palabras reservadas class y extends se han vuelto populares en frameworks relevantes para la web actual (angular y react), y en donde el código para escribir páginas web hace un par de años era dominado por librerías como jQuery y mucho código spaghetti.

La creación de una estructura unificada para que todos los laboratorios tengan una serie de características unificadas tiene como propósito que el estudiante pueda entender lo que se intenta solucionar con ese patrón. Esto ayudaría a que siempre que necesitara buscar algún apartado en específico, lo pudiera ubicar en cada laboratorio. Para poder crear esta estructura se definió un enfoque centrado en explicar el problema que usualmente se encuentra en dos apartados: "Porque necesitamos el patrón …" y "El problema en un ejemplo", descritos de una forma sencilla y fácil de entender. Finalmente, se presenta un diagrama con la estructura del patrón contextualizado en el ejemplo que se resolvió anteriormente para presentar por último las conclusiones.

La elaboración de actividades como foros sobre temas, trabajos y actividades aplicadas para trabajo independiente con apoyo de guías didácticas estructuradas de acuerdo a los lineamentos del departamento UNIAJCVirtual, para aplicar los patrones aprendidos fue creada con la idea de que podría presentarse una situación similar en la vida diaria como desarrollador de software. El aprendizaje se logra con la práctica (y más en la programación), y el poder proveer una actividad con un ejercicio diferente al del ejemplo ayudará a una mejor compresión de los patrones. La intención principal es que, en el proceso de aprendizaje de patrones, el estudiante pueda reforzar conceptos de programación básicos y OPP, como la herencia o composición, que raramente se podrán aplicar de una mejor forma en otro contexto.

RECOMENDACIONES Y TRABAJOS FUTUROS

Incentivar la publicación de cursos de programación

- Curso de Programación Orientada a Objetos: explique los pilares: abstracción, polimorfismo, encapsulación y herencia. Esta recomendación es dada debido a que muchas veces llegan estudiantes de Homologación interesados en reforzar sus conocimientos en estas áreas.
- Curso de Principios de Diseño de Software: explique términos como la reutilización de código, extensibilidad y los principios SOLID:
    - Single Responsibility Principle
    - Open/Closed Principle
    - Liskov Substitution Principle
    - Interface Segregation Principle
    - Dependency Inversion Principle

REFERENCIAS BIBLIOGRÁFICAS

Angular. (2020). Angular. La plataforma del desarrollador moderno. https://angular.io/

Bedoya Escobar, R. A. (2020). Entorno virtual para la formación de tecnólogos e ingenieros de sistemas en patrones de diseño de software (Trabajo de grado). Institución Universitaria Antonio José Camacho, Cali, Colombia.

Blackboard. (2020). Blackboard. Enseñanza y aprendizaje. https://www.blackboard.com/es-lac

Chamilo-Asociación Chamilo. (2020) Chamilo. https://chamilo.org/en/

Deperlioglu, O., & Köse, U. (2013). The effectiveness and experiences of blended learning approaches to computer programming education. Computer Applications in Engineering Education, 21. https://doi.org/10.1002/cae.20476

Edmodo. (2020). Edmodo. https://newedmodo.. com/?go2url=%2Fhome

Educativa. (2020). Educativa: Líderes de Iberoamérica en sistemas para formación. https://www.educativa.com/

Gamma, E., Richard Helm, Ralph Johnson, & John Vlissides (Eds.). (1995). Design patterns: Elements of reusable object-oriented software. Addison-Wesley.

Ganduxé, M. (2018). ¿Qué es el e-learning? eLearning Actual. https://elearningactual. com/e-learning-significado/

GNU General Public License. (2007, junio 29). GNU General Public License. http://www. gnu.org/licenses/gpl-3.0.txt

JavaScript. (s. f.). Documentación web de MDN. https://developer.mozilla.org/es/docs/ Web/JavaScript

Moodle. (2020). Moodle—Open-source learning platform. https://moodle.org/

Moreira, M. A., Santos, M. a B. S. N., & Vargas, E. F. (2010). Buenas Prácticas De Aulas Virtuales En La Docencia Universitaria Semipresencial. Teoría de la Educación. Educación y Cultura en la Sociedad de la Información, 11(1), 7-31.

Open edX. (2021). Open EdX. https://open.edx.org/

Pappas, C. (2015). The Top Open-Source Learning Software. ELearning Industry. https://elearningindustry.com/top-open-source-learning-management-systems

React. (2021). React – A JavaScript library for building user interfaces. https://reactjs.org/

Easy LMS. (2021). Significado de LMS: ¿Qué es un LMS? Definición explicada de LMS. https://www.easy-lms.com/es/centro-de-conocimiento/centro-de-conocimiento-lms/que-es-un-lms/item10182

Wang, F. L., Fong, J., Choy, M., & Wong, T.-L. (2007). Blended teaching and learning of computer programming. Proceedings of the 6th international conference on Advances in web-based learning, 606-617.

AUTORES

Ramiro Andrés Bedoya Escobar: ingeniero de Sistemas de la UNIAJC. Correo electrónico: imramiro.edu@gmail.com

Tania Isadora Mora Pedreros: ingeniera de Sistemas y Magíster en Tecnología Educativa y Competencias Digitales. Docente Tiempo Completo Facultad de Ingenierías, UNIAJC. Correo electrónico: timora@admon.uniajc.edu.co