Aplicación del patrón de diseño MVC utilizando Yii2 Framework en el desarrollo del módulo de Consulta Externa del Sistema Integral Hospital Rovirosa (SIHR)

Wilver Omar Figueroa Escudero, Miguel Pérez Vasconcelos, Fidelio Castillo Romero, Rosa Gómez Domínguez, Eutimio Sosa Silva

Tecnológico Nacional de México/Instituto Tecnológico de Villahermosa, División de Estudios de Posgrado e Investigación.

Resumen

El propósito de este artículo es mostrar los beneficios que aporta el uso del patrón de diseño arquitectónico Modelo Vista Controlador (MVC) en el desarrollo de software, asimismo se plantea la manera en que se implementa en el desarrollo del módulo de consulta externa del Sistema Integral Hospital Rovirosa (SIHR) para el Hospital Regional de Alta Especialidad Dr. Gustavo A. Rovirosa Pérez, como un modelo que propicia la escalabilidad, robustez y flexibilidad necesaria para un sistema de información.

Abstract

The purpose of this article is to show the benefits of the use the Model View Controller (MVC) architectural design pattern in the development of software, as well as the way in which it is implemented in the development of the external consultation module of the Integral System Rovirosa Hospital (SIHR) for the Regional Hospital of High Specialty Dr. Gustavo A. Rovirosa Pérez, as a model that favors the scalability, robustness and flexibility necessary for an information system.

Palabras claves: Patrón de diseño, Modelo Vista Controlador, Yii2 Framework, Sistema Integral, Hospital Rovirosa.
Keywords: Design Pattern, Model View Controller, Yii2 Framework, Integral System, Rovirosa Hospital.

1. INTRODUCCIÓN

En la actualidad el desarrollo de software juega un papel importante dentro de las empresas que gestionan cada uno de sus procesos a través de un Sistema de Información. La tendencia está orientada hacia la construcción de soluciones informáticas en las que se consideran aspectos fundamentales como lo son: lograr la mayor calidad en el menor tiempo posible, implementar estándares en el diseño de las aplicaciones que permitan mayor reutilización del código para facilitar el mantenimiento de los sistemas desarrollados [4] y ofrecer el soporte necesario a la lógica de negocio establecida por la empresa.

Existe una necesidad creciente de aplicar diferentes modelos que permitan alcanzar los objetivos que demanda el desarrollo de software a la vanguardia tecnológica. Uno de los patrones de diseño más utilizado en grandes proyectos de software es el Modelo Vista Controlador (MVC) [10].

La implementación del MVC no es fácil cuando se desarrollan aplicaciones a gran escala. Todo el proceso que sugiere el patrón conlleva una serie de acciones monótonas, lo cual absorbe demasiado tiempo y esfuerzo de parte de los desarrolladores de software [3]. Yii2 Framework es una herramienta para desarrollo en lenguaje PHP que facilita la aplicación del MVC ya que su estructura está diseñada para funcionar bajo la concepción de este patrón de diseño, pero lo que hace más interesante a esta herramienta es su gran eficiencia, las características y clara documentación que posee, así como la gran comunidad de desarrolladores de software que lo utilizan, destacando entre una diversidad de frameworks que se encuentran actualmente en el mercado como son: Laravel, CodeIgniter y Symfony. Otro aspecto que lo hace aún más interesante es la forma en que promueve la

organización de código bajo la filosofía de escribir código de manera simple y elegante [16] por lo que su utilización en el desarrollo de cualquier proyecto permite agilizar los tiempos de desarrollo a la vez que se genera software de calidad pues una de sus características es que emplea el paradigma de Programación Orientada a Objetos (POO) [13] lo que le permite ser extensible a través de los diversos Widgets de los que se puede disponer.

Este artículo propone una visión generalizada del patrón de diseño Modelo Vista Controlador (MVC) y su implementación a través de Yii2 Framework en el desarrollo del módulo de Consulta Externa del Sistema Integral Hospital Rovirosa (SIHR) en la ciudad de Villahermosa, Tabasco, México.

2. EL PATRÓN DE DISEÑO MVC

La expresión "patrón de diseño" fue mencionada por Cristopher Alexander en 1977, estaba enfocada a la conformación de un lenguaje de patrones a partir de soluciones que la arquitectura y el urbanismo visualizaban de los problemas arquitectónicos más habituales. La finalidad era compartir experiencias y conocimientos utilizados, donde su aplicación fuese flexible, adaptándose al contexto y situación específica para lo que se necesitara [1].

Para Mor y Winters [9] un patrón de diseño es: «una descripción semi-estructurada del método de un experto para resolver un problema recurrente que incluye una descripción del problema y del contexto en que será aplicado. Los patrones de diseño tienen el objetivo explícito de externalizar el conocimiento para permitir la acumulación y la generalización de soluciones y para hacer posible que todos los miembros de una comunidad o grupo de diseño participen en los debates relacionados con el diseño».

Desde esta perspectiva, un patrón de diseño es una forma de estructurar soluciones para problemas comunes y recurrentes que pueden implementarse en el desarrollo de aplicaciones. Es el resultado de conjuntar conocimiento y vivencias de las mejores alternativas de solución, que de manera estructurada faciliten al usuario identificar las problemáticas más habituales y encontrar soluciones escalables.

Desde los años 90, se empezó a diversificar el uso de los patrones de diseño hacia otras áreas muy distantes de la Arquitectura, campo para el que originalmente fueron concebidos, hacia la ingeniería de software [7], en donde permiten describir a objetos comunicantes y clases personalizadas para resolver un problema de diseño general en un contexto particular [5].

El Modelo Vista Controlador (MVC), es un patrón de diseño arquitectónico que define la manera en que se encuentran organizados los componentes de presentación, es muy utilizado en el desarrollo de aplicaciones; muestra una separación entre los tres elementos que lo componen: el modelo, la vista y el controlador [6].

El modelo es el responsable de la lógica del negocio ya que representa la información almacenada en archivos o en la base de datos con la cual opera la aplicación, gestiona las validaciones y las operaciones básicas sobre los datos [11].

La vista es una representación separada de los datos del modelo, define como se deben de mostrar los datos de la aplicación y contiene elementos de la interfaz de usuario como textos, formularios de entrada [14] videos, música y documentos [11].

ISSN: 2007-4786

A través de la vista se logra la interacción entre el usuario y la aplicación, siendo la forma en que se logra obtener datos y enviarlos para procesarlos.

El controlador es un intermediario entre el usuario y aplicación que captura, interpreta y actúa en función de las acciones que son ejecutadas a través de la vista y regresa una respuesta acorde al estado actual del sistema. Representa el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el modelo. Cuando se entera de que se realizó un cambio en la información del modelo o alteraciones en la vista actúa dando significado a la petición [4].

En la Figura 1 se ilustra los elementos que componen el MVC, el usuario realiza una petición al controlador a través de la vista mediante una acción, el controlador consulta los datos de la lógica del negocio, el modelo regresa la información solicitada para que se realicen los cambios necesarios en la vista, siendo esta última la responsable de mostrar la respuesta al usuario.

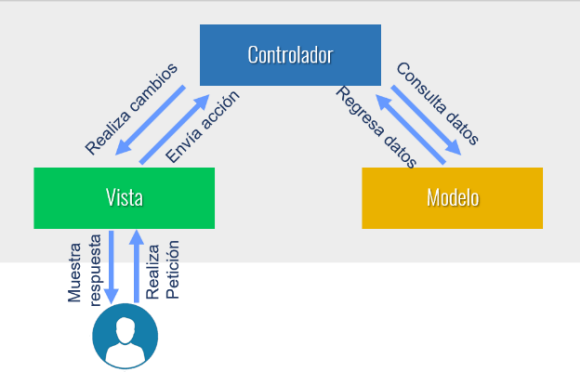

Figura 1. Elementos que integran el MVC

2.1 Ventajas

La implementación del patrón de diseño MVC ofrece múltiples prestaciones que justifican su utilización.

El MVC es un patrón de diseño probado con el que se pueden desarrollar aplicaciones rápidamente, de forma modular, mantenible, así como separar su funcionalidad en modelos, vistas y controladores que hacen que la aplicación sea muy ligera, facilitando la forma en que se visualiza su estructura, de esta manera posibilita el trabajo colaborativo a través del cual los diseñadores y desarrolladores de software trabajan en conjunto. Esta separación en capas permite efectuar cambios en una parte de la aplicación sin que las demás se vean afectadas[2] esto lo hace deseable para proyectos de grandes dimensiones.

De acuerdo con Camarena(et al.) [3] y la Junta de Andalucía [8] las ventajas de utilizar el MVC son:

a) Permitir la sustitución de las interfaces de usuario.
b) Generar componentes de las interfaces.
c) Diseñar vistas simultáneas del mismo modelo.
d) Aplicar fácilmente cambios de las interfaces.
e) Sus vistas muestran información actualizada siempre.

f) Cualquier cambio que afecte al modelo, no afecta el mecanismo de comunicación y de actualización entre modelos.
g) Las modificaciones a las vistas no afectan al modelo.
h) MVC está demostrando ser un patrón de diseño bien elaborado puesto que las aplicaciones que lo implementan presentan una extensibilidad y una mantenibilidad únicas comparadas con otras aplicaciones basadas en otros patrones.

Las ventajas de utilizar este patrón de diseño arquitectónico son muchas y varían de acuerdo al tipo y tamaño de proyecto en que se pretenda implementar.

2.2 Desventajas

Hay aspectos que tienen que considerarse al momento en que se plantea utilizar el MVC. Uno de ellos es que para desarrollar una aplicación bajo la concepción de este patrón es necesario una mayor dedicación en los tiempos iniciales del desarrollo que tiempo después se ve compensada en la etapa de mantenimiento de la aplicación, dando como resultado una aplicación MVC mucho más mantenible, extensible y modificable que una aplicación que no lo implementa [8].

MVC hace uso de una arquitectura inicial que es necesaria ya que permite modificar y comunicar los módulos de una aplicación lo que aumenta la complejidad [3], esto genera un esfuerzo de creación de las capas con las que funcionará. De igual forma, está orientado a objetos por lo que su implementación es sumamente costosa y difícil en lenguajes que no siguen este paradigma [8].

A través de estos factores vemos reflejado algo que es muy constante en el desarrollo de software como lo es la creación de componentes que conciban este marco de desarrollo, la respuesta contundente es un Framework que haga esa tarea por nosotros y que incluya un conjunto estandarizado de conceptos, prácticas y criterios que faciliten la tarea de crear la estructura en la que se basa el MVC.

3. APLICACIÓN DEL MVC

3.1 El MVC en Yii2 Framework

Yii2 Framework es un entorno de trabajo colaborativo que permite desarrollar software bajo la concepción del paradigma de programación orientado a objetos.

La guía definitiva para Yii 2.0 [15] lo define como: «un marco PHP basado en componentes de alto rendimiento para desarrollar rápidamente aplicaciones web modernas que puede utilizarse para desarrollar todo tipo de aplicaciones web con PHP. Debido a su arquitectura basada en componentes y su sofisticado soporte de almacenamiento en caché, es especialmente adecuado para desarrollar aplicaciones a gran escala como portales, foros, sistemas de gestión de contenido (CMS), proyectos de comercio electrónico y servicios web RESTful».

Yii2 implementa el patrón de diseño arquitectónico MVC y promueve la organización del código en función de dicho patrón por lo que mediante esta herramienta se garantiza su total aplicación, por esta razón no tenemos que preocuparnos por la tarea de crear una estructura que gestione una aplicación basada en el paradigma MVC, ni por la tareas de mantenimiento que resulten difíciles ya que todo esto se realiza de manera rápida y sencilla dedicando mayor tiempo a las soluciones enfocadas al proceso de negocio de un sistema. Es extremadamente extensible ya que una de sus características es que puede implementar Widgets lo cuales

«son bloques de código reutilizables que se usan en las vistas para crear elementos de interfaz de usuario complejos y configurables, de forma orientada a objetos» [13] y mediante los cuales se representa la ideología de la orientación a objetos.

La forma en que se estructuran las aplicaciones en Yii está basada en el patrón arquitectónico modelo-vista-controlador (MVC). Los modelos representan datos, lógica de negocios y reglas; las vistas son representaciones de salida de modelos; y los controladores toman la entrada y la convierten en comandos para modelos y vistas[12].

En las aplicaciones que se desarrollan con esta herramienta se consideran las siguientes entidades [12]:

- *Scripts de entrada*: son scripts PHP a los que los usuarios finales pueden acceder directamente. Son responsables de iniciar un ciclo de manejo de solicitudes.
- *Aplicaciones*: son objetos de acceso global que administran los componentes de la aplicación y los coordinan para cumplir con las solicitudes.
- *Componentes de la aplicación*: son objetos registrados con aplicaciones y brindan diversos servicios para satisfacer solicitudes.
- *Módulos*: son paquetes independientes que contienen MVC completos por sí mismos. Una aplicación se puede organizar en términos de múltiples módulos.
- *Filtros*: representan el código que debe invocarse antes y después del manejo real de cada solicitud por parte de los controladores.
- *Widgets*: son objetos que se pueden incrustar en vistas. Pueden contener lógica de controlador y pueden reutilizarse en diferentes vistas.

En la Figura 2 se muestra la estructura estática de una aplicación en Yii2 Framework.

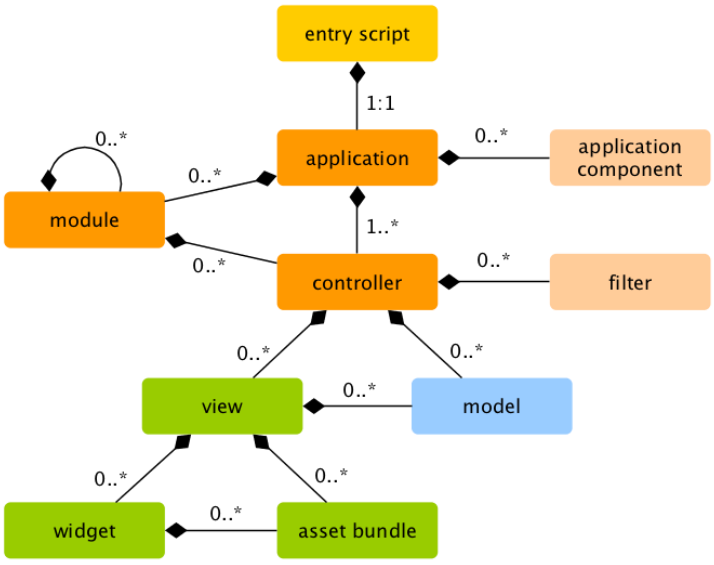

Figura 2. Estructura de una aplicación en Yii2 Framework [12]

Para desarrollar software bajo la concepción del patrón de diseño MVC Yii2 Framework implementa *Gii* un potente generador automático de código que a partir de tablas que se encuentran en una base de datos crea el modelo, la vista y el controlador. Inicialmente se tiene que generar un modelo (model), después la vista y el

controlador; es mediante la creación de un *CRUD* (Create, Read, Update, Delete) que es posible crear, leer, actualizar y eliminar registros de una tabla, así como, visualizarlos a través de la vista (view) que es dirigida por un controlador (controller).

3.2 Aplicación de MVC en el desarrollo del módulo de Consulta Externa del Sistema Integral Hospital Rovirosa (SIHR)

Yii2 Framework es un conjunto de componentes que nos permiten agilizar el proceso de desarrollo de una aplicación web, debido a que su curva de aprendizaje es muy corta facilita su utilización en proyectos grandes y que a futuro deben de ser extensibles, sin dejar de lado que el mantenimiento necesario en todo sistema es relativamente fácil. Esta fue la razón principal por la que se decidió desarrollar el módulo de Consulta Externa del Sistema Integral Hospital Rovirosa (SIHR) con el apoyo de esta herramienta tecnológica.

El módulo de Consulta Externa del Sistema Integral Hospital Rovirosa (SIHR) forma parte de un desarrollo a nivel modular que administrará la prestación del servicio de consulta médica a todos los pacientes que acudan en cada una de las especialidades que se ofrecen, es uno de los tantos procesos que se administran dentro del Hospital Regional de Alta Especialidad Dr. Gustavo A. Rovirosa Pérez, esto hizo evidente la necesidad de contemplar la necesidad que existirá en un futuro con el desarrollo de otras funcionalidades que formarán parte del sistema integral que de manera gradual se le adicionarán nuevos módulos, trayendo consigo nuevas mejoras que justificarán aún más su utilización. Todo este horizonte de oportunidades ocasionó un análisis detallado de la situación en la que se concluyó que el desarrollo a realizar involucraría a diversos desarrolladores de software que harían uso de la misma base de datos, determinándose utilizar el patrón de diseño arquitectónico Modelo Vista Controlador (MVC) en el desarrollo del módulo debido a que su implementación a través de Yii2 Framework simplifica la tarea de crear una estructura que soporte este modelo y a su vez, permita visualizar el proceso de consulta externa.

Para modelar el proceso de la consulta externa mediante el patrón de diseño MVC se utilizó la estructura que provee Yii2 Framework misma que contempla una división de funcionalidades a través de carpetas en las que se encuentran contenidos una diversidad de componentes. En la Figura 3 se ilustra las carpetas en las que se encuentran los *modelos* (models), las *vistas* (views) y los *controladores* (controllers).

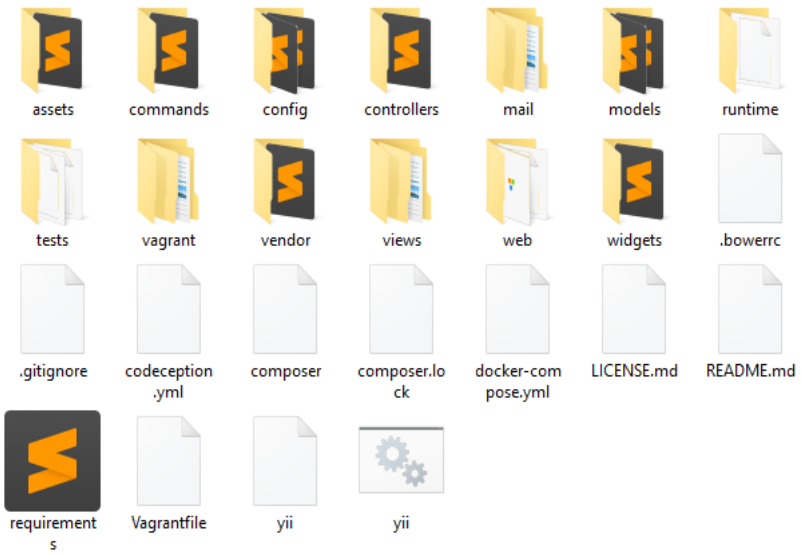

Figura 3. Estructura de carpetas básicas de una aplicación en Yii2 Framework

Para crear los *modelos* se utilizó el generador de código integrado *Gii* que viene contenido dentro del template básico de Yii2 Framework. Esta aplicación nos permitió generar el código a partir de tablas de la base de datos, guardándolos en la carpeta *models* de manera automatizada. Para generar un modelo se realiza la selección de una tabla de la base de datos, colocamos un nombre que lo identificará y en caso de necesitar alguna de las demás opciones se realiza la selección. En la Figura 4 se muestra los elementos que requiere la aplicación para generar el modelo a partir de cada tabla.

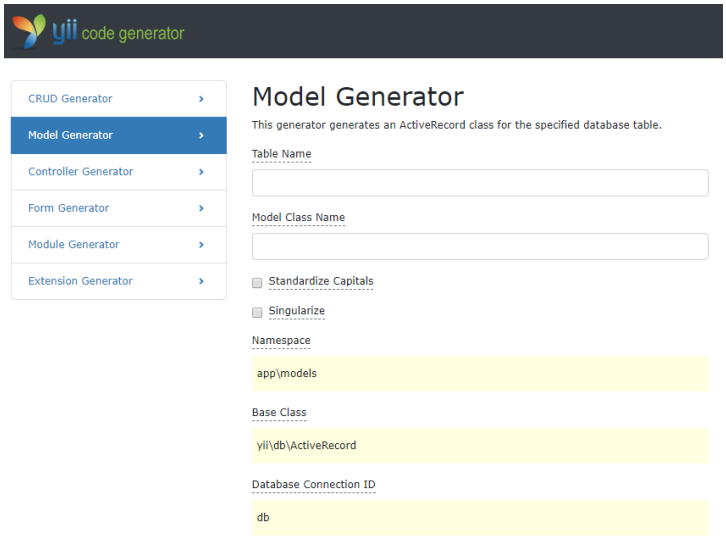

Figura 4. Generador de modelo Gii

Para el desarrollo del módulo de consulta externa se realiza la generación de los modelos necesarios para su funcionamiento. En la Figura 5 se muestra la estructura básica del proyecto.

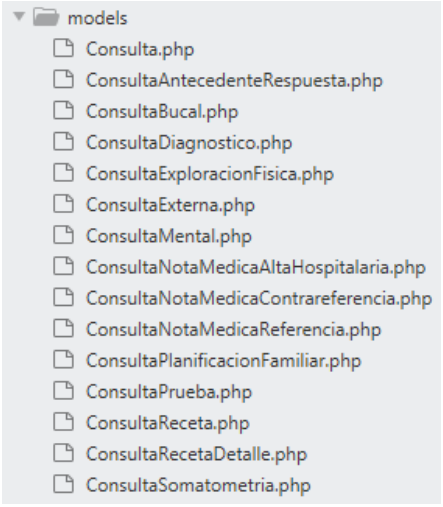

Figura 5. Estructura básica del módulo de consulta externa

Un modelo en Yii2 se ve representado en un archivo .php y este permite representar las tablas de nuestra base de datos como una clase.

En la Figura 6 se observa la clase consulta y alguna de las funciones que lo integran.

```php
<?php
namespace app\models;

class Consulta extends \yii\db\ActiveRecord
{
    public static function tableName()
    {
        return 'consulta';
    }
    public function rules()
    {
    }

    public function attributeLabels()
    {
    }

    public function getConFkPaciente()
    {
        return $this->hasOne(Paciente::className(), ['pac_id' => 'con_fkPaciente']);
    }
```

Figura 6. Ejemplo modelo Consulta

En la parte de la elaboración de la vista y la generación del controlador se utilizó el CRUD generator de Gii con el que se creó toda la estructura necesaria para visualizar y controlar las acciones sobre los registros de los modelos. Para cada modelo la herramienta crea la siguiente estructura:

\models
    \modelo.php
    \modeloSearch.php
\views
    \subcarpeta
        _form.php
        _search.php
        \create.php
        \index.php
        \update.php
        \view.php
\controllers
    \controlador.php

Dentro de la carpeta *models* se encuentran todos los modelos de la aplicación, *modelo.php* y *modeloSearch.php* contienen los métodos que sustentan la lógica de negocio y la búsqueda de todos los registros existentes en la tabla del modelo de referencia de la base de datos, respectivamente.

En la carpeta *views* es donde se guardan los archivos que presentarán toda la información a partir de los modelos, dentro de ella se crea una subcarpeta en la que se encuentran varios archivos que permiten realizar operaciones sobre un registro, *_form.php* representa el formulario a través del cual se puede crear y/o actualizar, *_search.php* se utiliza para realizar una búsqueda, *create.php* permite a través del formulario realizar la creación de un nuevo registro, *index.php* es la página principal desde donde se administran todas la acciones que se realizan sobre una tabla, *update.php* gestiona la actualización sobre un registro y el *view.php* muestra todos los datos contenidos en el mismo.

La carpeta *controllers* es donde se ubican los controladores de todo el módulo de consulta externa, *controlador.php* es el controlador de un modelo. En la Figura 7 se muestra el código del controlador de consulta

ISSN: 2007-4786

externa generado con Gii, en él se pueden observar alguno de los métodos que administran las acciones de las vistas.

```php
class ConsultaExternaController extends Controller
{
    public function behaviors()
    {
        return [
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['POST'],
                ],
            ],

        ];
    }

    public function actionIndex()
    {
        $searchModel = new ConsultaSearch();
        $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }

    public function actionView($id)
    {
        return $this->render('view', [
            'model' => $this->findModel($id),
        ]);
    }

    public function actionViewReceta($id)
    {
        return $this->render('viewReceta', [
            'modelReceta' => $rs = ConsultaReceta::findOne(['rec_id'=>$id]),
            'modelRecetaDetalle' => ConsultaRecetaDetalle::find()
            ->where('recdet_fkReceta=:inicio', [':inicio' => $rs->rec_id])->all(),
        ]);
    }
```

Figura 7. Controlador de Consulta Externa

De lo anterior, se determina que por cada modelo existe una carpeta con sus vistas y un controlador que será el encargado de administrar las peticiones que realice el usuario sobre cada una de las acciones que muestre la vista.

De esta manera, el Modelo Vista Controlador tiene su aplicación a través de Yii2 Framework en el desarrollo del módulo de Consulta Externa del Sistema Integral Hospital Rovirosa, con lo cual se agiliza el tiempo y esfuerzo que representa la generación de código de manera manual para dedicar mayor tiempo a otras actividades igual de importantes y que forman parte medular del desarrollo de este proyecto.

La aplicación del MVC al desarrollo de este proyecto permitió que los otros dos desarrolladores que forman parte del proyecto a nivel modular del Sistema Integral Hospital Rovirosa se integraran fácilmente, de esta manera se logra conjuntar esfuerzos para este desarrollo de software.

4. CONCLUSIONES

El patrón de diseño arquitectónico MVC es una forma de estructurar las funciones básicas de un proyecto de software, de manera que se encuentren perfectamente ordenadas para que nuestra aplicación sea fácil de entender por otros desarrolladores cuando el tamaño es muy grande y el desarrollo se realiza a nivel modular, permitiendo que otros programadores entiendan la forma en que fue construido y la finalidad para la que fue creado nuestro sistema, para que en lo sucesivo, nuestro proyecto conserve un código con la claridad necesaria para los mantenimientos o modificaciones que se pudiesen requerir en un futuro.

ISSN: 2007-4786

El uso del patrón MVC se encuentra muy difundido, tenemos una diversidad de posibilidades para incluir en nuestros proyectos de software a herramientas que facilitan la generación del código básico para nuestra aplicación, de esta manera los desarrolladores conocen nuevos mecanismos útiles y ágiles para poder desarrollar sistemas basados en patrones. Es importante utilizar dichas herramientas, como una alternativa viable, rápida y confiable en el desarrollo de software de calidad.

El utilizar Yii2 Framework en el desarrollo de este proyecto permitió generar la estructura del patrón de diseño MVC, reducir considerablemente el tiempo de programación y automatizar el proceso de codificación.

REFERENCIAS

[1] Alexander, C., Ishikawa, S., & Silverstein, M. (1977). A Pattern Language. Towns, Buildings, Construction. New York: Oxford University Press.

[2] Cake Software Foundation Inc. Entendiendo el Modelo-Vista-Controlador. Obtenido de https://book.cake php.org/1.3/es/The-Manual/Beginning-With-CakeP HP/Understanding-Model-View-Controller.html

[3] Camarena, J., Trueba, A., Martínez, M., & López, M. (2012). Automatización de la codificación del patrón modelo vista controlador (MVC) en proyectos orientados a la Web. Ciencia Ergo Sum, 19(3), 239-250.

[4] Fernández, Y., & Díaz, Y. (2012). Patrón Modelo-Vista-Controlador. Telem@tica, 11(1), 47-57.

[5] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Design Patterns: Elements of Reusable ObjectOriented Software. Reading, MA: AddisonWesley.

[6] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (2003). Introducción. Patrones de Diseño. Elementos de software orientado a objetos reutilizable. Madrid, España: Pearson Educación, S.A.

[7] Gros, B., Escofet, A., & Marimón, M. (2016). Los patrones de diseño como herramientas para guiar la práctica del profesorado. Revista Latinoamericana de Tecnología Educativa, 15(3), 12-14.

[8] Junta de Andalucia. Patrón Modelo Vista Controlador. Obtenido de http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/122

[9] Mor, Y., & Winters, N. (2007). 'Design approaches in technology enhanced learning'. Interactive Learning Environments, 15(1), 61-75. doi:https://doi.org/10 .1080/10494820601044236

[10] Morales, L., Rengifo, S., & Reyes, D. (2010). Aplicación del Patrón de Diseño MVC, en el Desarrollo de la Plataforma del Sistema General de Seguridad Social en Salud (SGSSS) de Barranquilla. Investigación y Desarrollo en TIC, 1, 79-81. Obtenido de https://revistas.unisimon.edu.co/index.php/identic/article/view/2470

[11] Naranjo, H., & Jiménez, E. (2017). Utilización de la arquitectura Modelo - Vista - Controlador (MVC) en el desarrollo de una aplicación web de catálogos privados. Repositorio Interno Universidad Técnica de Ámbato, 1-11. Obtenido de http://redi.uta.edu.ec/jspui/handle/123456789/37301

[12] YiiFramework. Estructura de la aplicación: Descripción general de la estructura de la aplicación | La guía definitiva para Yii 2.0 | Marco PHP Yii. Obtenido de https://www.yiiframework.com/doc/guide/2.0/es/structure-overview

[13] YiiFramework. Estructura de una aplicación: Widgets | Guía Definitiva de Yii 2.0 | Yii PHP Framework. Obtenido de https://www.yiiframework.com/doc/guide/2.0/es/structure-widgets

[14] YiiFramework. Fundamentos: Modelo-Vista-Controlador (MVC). Obtenido de https://www.yiiframework.com/doc/guide/1.1/es/basics.mvc

[15] YiiFramework. Introducción: Acerca de Yii | La guía definitiva para Yii 2.0 | Marco PHP Yii. Obtenido de https://www.yiiframework.com/doc/guide/2.0/en/intro-yii

[16] YiiFramework. Introducción: Qué es Yii. Obtenido de https://www.yiiframework.com/doc/guide/1.1/es/quickstart.what-is-yii

Correo electrónico autor: *wilver.omar.figueroa.escudero@gmail.com*