Development of the Comprehensive System of the High Specialty Regional Hospital "Dr. Gustavo A. Rovirosa Pérez": Medical emergency module with Yii2 Framework and the MVC design pattern

Camara Ayala Luis Fernando, Gómez Zea José Manuel, Jesús Magaña José Angel, Garrido Vázquez José Ney, Hernández Cadena Alejandro

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

Abstract

In recent years, the use of frameworks for application development has increased, due to the fact that they use a better structure through design patterns and code organization. This article addresses the Yii2 framework in conjunction with the MariaDB database management and the Nginx web server, as well as the artifacts (models, views, controllers and components) resulting from the development of the medical emergencies module of the Rovirosa Hospital Integral System (SIHR) for the Regional Hospital High Specialty "Dr. Gustavo A. Rovirosa Pérez", which is located in the city of Villahermosa, Tabasco, Mexico.

Resumen

En los últimos años se ha incrementado el uso de framework para el desarrollo de aplicaciones, debido a que utilizan una mejor estructura mediante los patrones de diseño y organización de código. En este artículo se aborda el marco de trabajo Yii2 en conjunto con el sistema manejador de base de datos MariaDB y el servidor web Nginx, así como los artefactos (modelos, vistas, controladores y componentes) resultado del desarrollo del módulo de urgencias médicas del Sistema Integral Hospital Rovirosa (SIHR) para el Hospital Regional de Alta Especialidad "Dr. Gustavo A. Rovirosa Pérez", que se encuentra localizado en la ciudad de Villahermosa, Tabasco, México.

Palabras clave: Yii2 Framework, Sistema Integral, Hospital Regional, Modelo Vista Controlador, Hospital Rovirosa.
Keywords: Yii2 Framework, Integral System, Regional Hospital, Model View Controller, Rovirosa Hospital.

1. INTRODUCCIÓN

In recent years the Regional Hospital mentioned above, in its need to cover its deficiency in terms of systems that where its strength is interoperability to carry out an adequate management of the information to display it in a medical consultation and that the doctor responsible for the care be able to use this tool intuitively and this is where the Yii2 Framework and the MVC (Model-View-Controller) design pattern come together in the development of this module being a part of a comprehensive system.

1.1. The MVC design pattern

In Smalltalk-80, input and output are largely stylized. Views must manage screen real estate and display text or graphic forms within that real estate. Controllers must cooperate to ensure that the proper controller is interpreting keyboard and mouse input (usually according to which view contains the cursor). Because the input and output behaviour of most applications is stylized, much of it is inherited from the generic classes -- View and Controller. These two classes, together with their subclasses, provide such a rich variety of behaviours that your applications will usually require little added protocol to accomplish their command input and interactive output behaviour. In contrast, the model cannot be stylized. Constraints on the type of objects allowed to function as models would limit the useful range of applications possible within the MVC paradigm. Necessarily, any object can be a model. A float number could be the model for an airspeed view which might be a sub view of a more complex flight simulator instrument panel view. A String makes a perfectly usable model for an editor application (although a slightly more complex object called a String Holder is usually used for such purposes). Because any object can play the role of model, the basic behaviour required for models to participate in the MVC paradigm is inherited from class Object which is the class that is a superclass of all possible models.

MVC is a pattern in software design commonly used to implement user interfaces, data, and controlling logic. It emphasizes a separation between the software's business logic and display. This "separation of concerns" provides for a better division of labor and improved maintenance. Some other design patterns are based on MVC, such as MVVM (Model-View-Viewmodel), MVP (Model-View-Presenter), and MVW (Model-View-Whatever).

The three parts of the MVC software-design pattern can be described as follows:

1. Model: Manages data and business logic.

2. View: Handle's layout and display.

3. Controller: Routes commands to the model and view parts.

In Figure 1 the elements that make up the MVC are identified, the user makes a request to the controller through a view through an action, the controller makes different queries through business logic, the model

ISSN: 2007-4786
Innovación y Desarrollo Tecnológico
Volumen 13 – Número 2
Abril – Junio 2021

returns the requested information so that if the necessary changes are made in the view, this is responsible for the data that is received in response to the user.

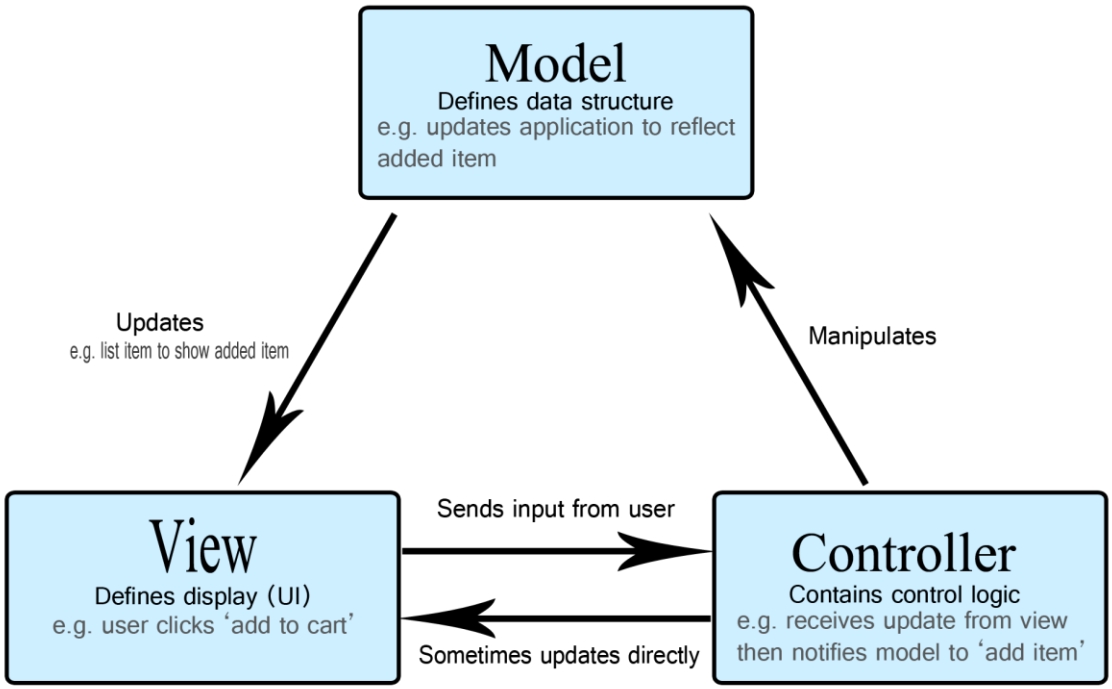

Figure 1. MVC Design Pattern

2. MariaDB

MariaDB Server is one of the most popular open-source relational databases. It's made by the original developers of MySQL and guaranteed to stay open source. It is part of most cloud offerings and the default in most Linux distributions.

It is built upon the values of performance, stability, and openness, and MariaDB Foundation ensures contributions will be accepted on technical merit. Recent new functionality includes advanced clustering with Galera Cluster 4, compatibility features with Oracle Database and Temporal Data Tables, allowing one to query the data as it stood at any point in the past.

3. Nginx

Nginx es un servidor web/proxy inverso ligero de alto rendimiento y un proxy para protocolos de correo electrónico. Es software libre y de código abierto, licenciado bajo la Licencia BSD simplificada; también existe una versión comercial distribuida bajo el nombre de Nginx Plus. Es multiplataforma, por lo que corre en sistemas tipo Unix y Windows.

Igor Sysoev creates NGINX to solve the C10K problem, revolutionizing how servers operate in high-performance contexts. Now, more than half of the Internet's busiest websites use NGINX.

4. Yii2 Framework

4.1. What is Yii

Yii is a high performance, component-based PHP framework for rapidly developing modern Web applications. The name Yii (pronounced Yee or [ji:]) means "simple and evolutionary" in Chinese. It can also be thought of as an acronym for Yes, It Is!

4.2. What is Yii Best for?

Yii is a generic Web programming framework, meaning that it can be used for developing all kinds of Web applications using PHP. Because of its component-based architecture and sophisticated caching support, it is especially suitable for developing large-scale applications such as portals, forums, content management systems (CMS), e-commerce projects, RESTful Web services, and so on.

4.3. How does Yii Compare with other frameworks?

If you're already familiar with another framework, you may appreciate knowing how Yii compares:

- Like most PHP frameworks, Yii implements the MVC (Model-View-Controller) architectural pattern and promotes code organization based on that pattern.
- Yii takes the philosophy that code should be written in a simple yet elegant way. Yii will never try to over-design things mainly for the purpose of strictly following some design pattern.
- Yii is a full-stack framework providing many proven and ready-to-use features: query builders and Active Record for both relational and NoSQL databases; RESTful API development support; multi-tier caching support; and more.
- Yii is extremely extensible. You can customize or replace nearly every piece of the core's code. You can also take advantage of Yii's solid extension architecture to use or develop redistributable extensions.
- High performance is always a primary goal of Yii.
-

4.4. Request Handling Overview

Each time when a Yii application handles a request, it undergoes a similar workflow.

a) A user makes a request to the entry script web/index.php.
b) The entry script loads the application configuration and creates an application instance to handle the request.
c) The application resolves the requested route with the help of the request application component.
d) The application creates a controller instance to handle the request.
e) The controller creates an action instance and performs the filters for the action.
f) If any filter fails, the action is cancelled.
g) If all filters pass, the action is executed.
h) The action loads a data model, possibly from a database.
i) The action renders a view, providing it with the data model.

ISSN: 2007-4786

j) The rendered result is returned to the response application component.
k) The response component sends the rendered result to the user's browser.

The following diagram shows how an application handles a request.

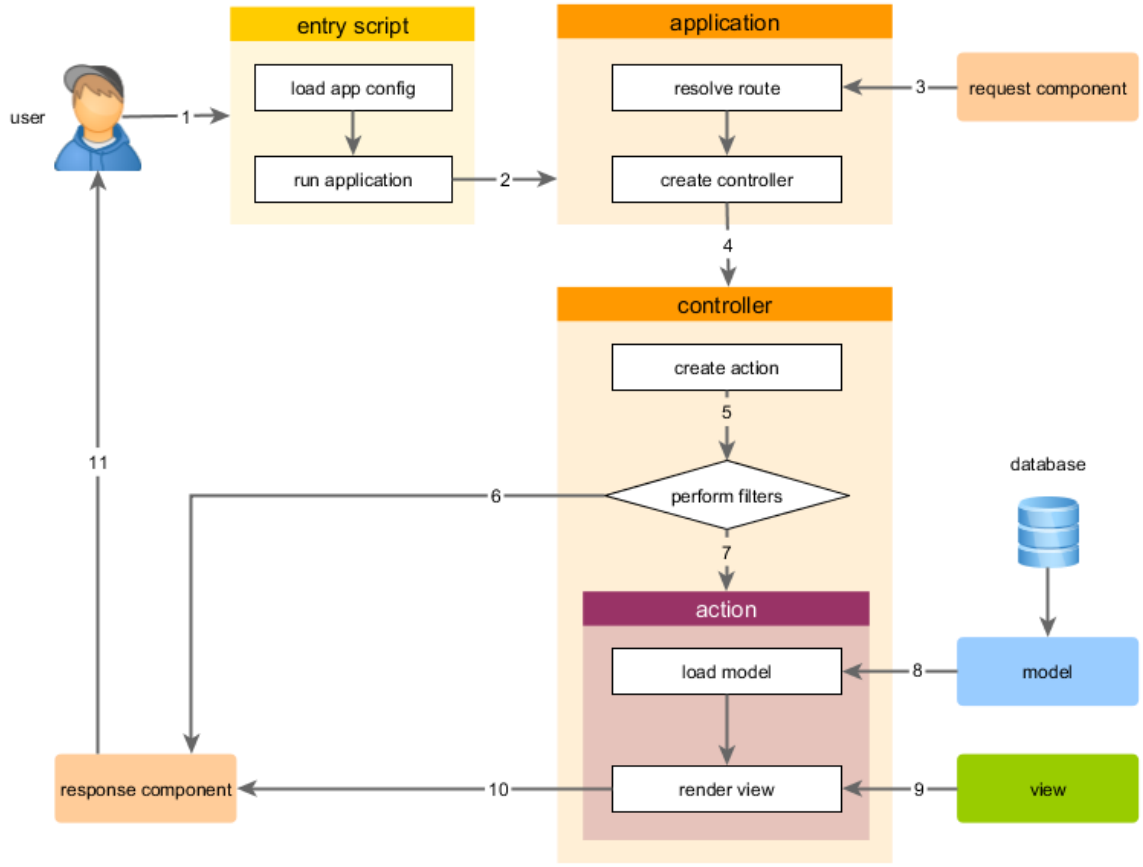

Figure 2. yii2 request handling on the client and server side

5. Medical emergency module with Yii2 Framework and the mvc design pattern
5.1. Yii2 Framework and module generation.

The most important directories and files in your application are (assuming the application's root directory is basic):

```
basic/                      application base path
composer.json               used by Composer, describes package information
    config/                 contains application and other configurations
        console.php         the console application configuration
        web.php the Web application configuration
    commands/               contains console command classes
    controllers/            contains controller classes
    models/                 contains model classes
    runtime/                contains files generated by Yii during runtime, such as logs and cache files
    vendor/                 contains the installed Composer packages, including the Yii framework itself
    views/                  contains view files
    web/                    application Web root, contains Web accessible files
```

assets/	contains published asset files (JavaScript and CSS) by Yii
index.php	the entry (or bootstrap) script for the application
yii	the Yii console command execution script

The following image illustrates the structure of a Yii2 basic project mentioned above on server-side using an ftp program for manage like FileZilla or WinSCP.

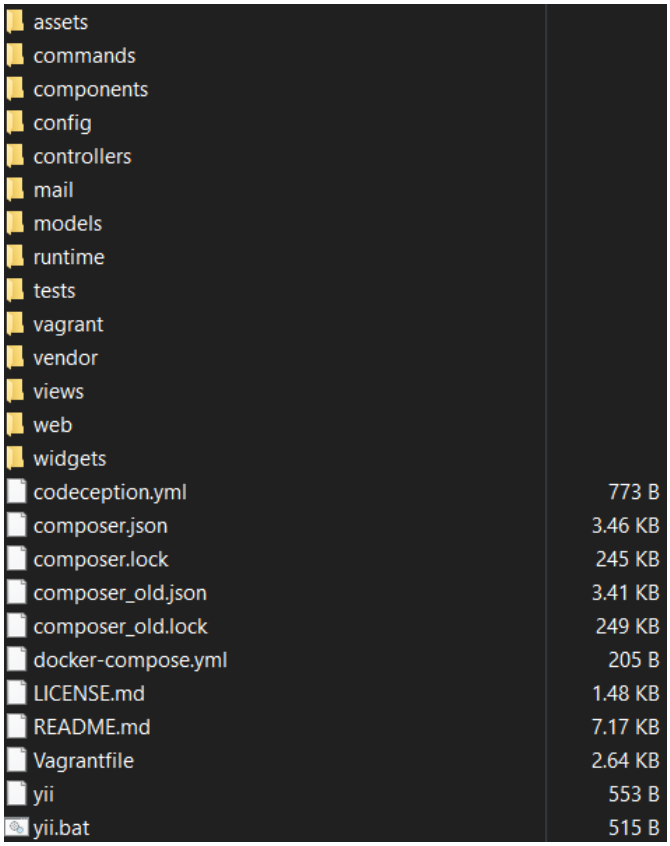

Figure 3. Yii2 basic template structure on server side

In the model's folder that is illustrated in figure 2, there are the files that will be generated with the Gii Model Generator tool by accessing a database and selecting the table to create the model, also these are the brain of

the application since these are all aspects of the table that has been generated as a model, a model is illustrated below.

```php
<?php

namespace app\models;

use Yii;

class UrgenciaEstancia extends \yii\db\ActiveRecord {

    public static function tableName() {
    }

    public function rules() {
    }

    public function attributeLabels() {
    }
}
```

Figure 4. EmergencyStay model

In this model called Urgency Stay are the aspects that were generated with the Model Generator, which are table Name, rules, attribute Labels and foreign keys in case the table contains them, however, these can exist both physically and logically, Yii will interpret them as if they existed, in the same way in the model other types of methods that go according to the need of the model can be added, having generated the model in the same way the controller can be created with the basic actions that are require, in the same way you can generate a blank controller and program the actions that are needed saving space for the code in comment mode that is generated, the following figures illustrates a controller generated with CRUD Generator and Controller Generator, and the folder that stores the controllers is called controllers and the way to declare that the controller is in the correct folder is done by using namespace "app\controllers" in figure 4 is illustrated.

```php
<?php

namespace app\controllers;

use Yii;
use app\models\UrgenciaEstancia;
use app\models\search\UrgenciaEstanciaSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;

class UrgenciasController extends Controller {
    public function actionIndex() {
    }

    public function actionView($id) {
    }

    /* SE INGRESA AL PACIENTE */
    public function actionCreate($id) {
    }

    /* SE ACTUALIZA EL REGISTRO */
    public function actionUpdate($id) {
    }


    /* BUSQUEDA DE OBJETOS POR MODELO */
    protected function findModel($id) {
    }

    /* ELIMINACIÃ"N DE REGISTROS */
    public function actionDelete($id) {
    }

}
```

Figure 5. Emergency controller generated with basic actions

```php
<?php

namespace app\controllers;

use Yii;
use app\models\UrgenciaEstancia;
use app\models\search\UrgenciaEstanciaSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;

class UrgenciasController extends Controller {

}
```

Figure 6. Emergency controller generated without basic actions (blank)

The views folder stores the files that are automatically generated by the CRUD generator or created manually depending on the needs of the programmer or the project itself, these files can be six or less if they are unchecked in the CRUD generator and more without being programmed manually, some of them are manually programmed as interconsultation emergency, the views folder containing the urgency folder and the actions illustrated in figure seven and eight contains the actions is illustrated below.

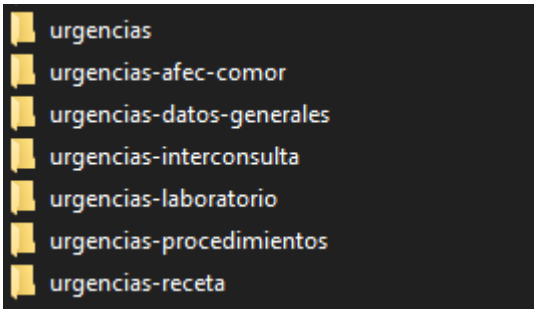

Figure 7. folders generated with gii crud and manually for urgency-dependent use

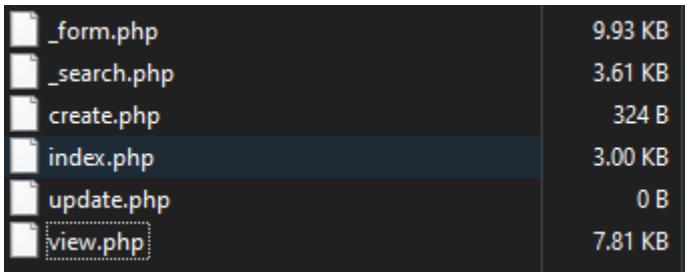

Figure 8. basic actions that are generated by default

These files can be modified depending on the need at the time of programming and while it is in development, since when the project is ready for deployment it is very difficult to make changes within the files and more if it is in use, due to the times. in which this is needed it can only be edited at night in case it is not occupied.

Figure 9. Action create

Figure nine illustrates the _form.php file that contains everything necessary to generate the create or update action depending on the case, this file is highly modified during the development stage so this file is only a shell of what can become at the moment of being finalized, for the creation of the action create the next code is necessary on file create.php or update.php depends on the case of use:

```
<?php

use yii\helpers\Html;

$this->title = 'Ingreso a Urgencias: '. $paciente->nombre;
$this->params['breadcrumbs'][] = ['label' => 'Urgencias'];
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="urgencia-estancia-create">
   <?= $this->render('_form', compact('model')) ?>
</div>
```

In the controller the create action can be modified depending on the situation, when using catalogs it is necessary to carry out the searches in the controller and not in the view, since if they are left in the view it can be a hole or direct access to the model that is needs at that time and this can be changed by third parties, so in the create file they are only called in the form of variables and this method is added to the create file with this form compact ('model', 'first variable', ' second variable ',' n variable ')

Figure 10. Action view

In figure ten the attributes of the table that are necessary in the view are specified, in the same way it can contain a link as illustrated in the image that contains the update and delete button respectively, this is only a file generated by default depending on the case this file can change many times.

In the view file it is composed in the same way as the create file, however it does not render a form, this final file displays depending on the data that was required at the time of development since the data that is displayed is the model itself Some data may appear in the forms of 1 and 0, but it can be changed using the foreign key that has been created in the model.

The following code is an example of how it can be used from the controller to the form passing through the create file so that the use is self-explanatory.

Controller create action

```
  public function actionCreate($curp, $id) {
    $model = new UrgenciaEstancia();
    $modelPaciente = Paciente::findOne(['pac_id' => $id]);
    $model->urgest_fkPaciente = $modelPaciente->pac_id;
    $model->urgest_proceso = 0;
    $aev = new UrgenciaEvento;
          if ( $model->load(Yii::$app->request->post()) ) {
      if($model->save()) {
          $aev->urgeve_fkUrgencia = $model->urgest_id;
        $aev->save();
        return $this->redirect(['set-folio', 'id' => $model->urgest_id]);
      }
    }
    return $this->render('create', [
      'model' => $model,
      'paciente' => $modelPaciente,
      'tipoUrgencia' => ArrayHelper::map(CatTipoUrgencias::find()->all(), 'tipurg_id', 'tipurg_nombre'),
      'servicio' => ArrayHelper::map(CatServicio::find()->all(), 'serv_id', 'serv_descripcion'),
    ]);
  }
```

Create file code

```
<?php
use yii\helpers\Html;
$this->title = 'Ingreso a Urgencias: '. $paciente->nombre;
```

```
$this->params['breadcrumbs'][] = ['label' => 'Urgencias'];
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="urgencia-estancia-create">
  <?= $this->render('_form', compact('model', 'tipoUrgencia', 'servicio')) ?>
</div>
form file code
<?php
use yii\helpers\Html;
use yii\helpers\Url;
use yii\web\JsExpression;
use kartik\form\ActiveForm;
use kartik\date\DatePicker;
use kartik\time\TimePicker;
use kartik\select2\Select2;
use kartik\depdrop\DepDrop;
?> <div class="box box-primary">
  <?php $form = ActiveForm::begin([
    'options' => [
      'autocomplete' => 'off',
      'class' => 'disable-submit-buttons'
    ],
  ]); ?>
  <div class="box-body">
    <div class="row">
      <div class="col-md-2">
        <?= $form->field($model, 'urgest_atencionPreHospitalaria')->widget(Select2::classname(), [
          'data' => Yii::$app->params['sino'],
          'language' => 'es',
          'options' => [
            'id' => 'sel-atencionPrehospitalaria',
            'prompt' => 'SELECCIONE',
            'required' => true,
            'onchange' => 'campoTraslado(this)'
          ],
          'pluginLoading' => false,
          'pluginOptions' => [
            'allowClear' => true
          ],
        ]); ?>
      </div>
      <div class="col-md-2" id="datos-atencion" style="display:none;">
        <?= $form->field($model, 'urgest_tiempoTraslado')->widget(TimePicker::classname(), [
          'id' => 'tiempoTraslado',
          'pluginOptions' => [
            'showSeconds' => false,
            'showMeridian' => false,
            'minuteStep' => 1,
            'secondStep' => 5,
          ],
        ]) ?>
      </div>
      <div class="col-md-2">
        <?= $form->field($model, 'urgest_fechaIngreso')->widget(DatePicker::classname(), [
          'options' => [
            'autocomplete' => 'off',
          ],
          'removeButton' => false,
```


ISSN: 2007-4786
Innovación y Desarrollo Tecnológico
Volumen 13 – Número 2
Abril – Junio 2021

```
        'type' => DatePicker::TYPE_COMPONENT_APPEND,
        'pluginOptions' => [
          'orientation' => 'bottom left',
          'autoclose' => true,
          'todayHighlight' => true,
          'format' => 'yyyy-mm-dd',
          'endDate' => 'Y-m-d'
        ]
      ]); ?>
    </div>
    <div class="col-md-2">
      <?= $form->field($model, 'urgest_horaIngreso')->widget(TimePicker::classname(), [
        'pluginOptions' => [
          'showSeconds' => false,
          'showMeridian' => false,
          'minuteStep' => 1,
          'secondStep' => 5,
        ],
      ]) ?>
    </div>
    <div class="col-md-2">
      <?= $form->field($model, 'urgest_numVez')->widget(Select2::classname(), [
      'language' => 'es',
      'data' => Yii::$app->params['ageCita'],
      'options' => [
        'placeholder' => 'SELECCIONE',
        'required' => true
      ],
      'pluginLoading' => false,
      'pluginOptions' => [
        'allowClear' => true
      ],
    ]); ?>
    </div>
  </div>
  <div class="row">
    <div class="col-md-3">
      <?= $form->field($model, 'urgest_fkTipoUrgencia')->widget(Select2::classname(), [
        'language' => 'es',
        'data' => $tipoUrgencia,
        'options' => [
          'placeholder' => 'SELECCIONE',
          'required' => true,
          'id' => 'cat-urgencia-id'
        ],
        'pluginLoading' => false,
        'pluginOptions' => [
          'allowClear' => true
        ],
      ]); ?>
    </div>
    <div class="col-md-3">
      <?= $form->field($model, 'urgest_fkClasificacionUrgencia')->widget(DepDrop::classname(), [
        'type' => DepDrop::TYPE_SELECT2,
        'options' => [
          'id' => 'cat-clasificador-id',
          'required' => true,
        ],
```


ISSN: 2007-4786
Innovación y Desarrollo Tecnológico
Volumen 13 – Número 2
Abril – Junio 2021

```
            'pluginOptions' => [
              'depends' => ['cat-urgencia-id'],
              'placeholder' => 'SELECCIONE',
              'url' => Url::to(['/urgencias-catalogos/cat-clasificador'])
            ]
          ]); ?>
        </div>
        <div class="col-md-3">
          <?= $form->field($model, 'urgest_tipoReferencia')->widget(Select2::classname(), [
            'language' => 'es',
            'data' => Yii::$app->params['tipoReferencia'],
            'options' => [
              'placeholder' => '=== SELECCIONE ===',
              'id' => 'sel-tipoReferencia',
              'required' => true,
              'onchange' => 'campoReferencia(this)'
            ],
            'pluginLoading' => false,
            'pluginOptions' => [
              'allowClear' => true
            ],
          ]); ?>
        </div>
      </div>
      <div class="row">
        <div class="col-md-12">
          <?= $form->field($model, 'urgest_causaIngreso')->textInput(['maxlength' => true, 'onkeyup' =>
'javascript:this.value=this.value.toUpperCase();', 'required' => true]) ?>
        </div>
      </div>
      <div class="row" id="unidadReferencia" style="display: none;">
        <div class="col-md-12">
          <?= $form->field($model, 'urgest_cluesRegistro')->widget(Select2::classname(), [
            'options' => ['placeholder' => 'Buscar Unidad Médica'],
            'pluginLoading' => false,
            'pluginOptions' => [
              'allowClear' => true,
              'minimumInputLength' => 5,
              'language' => [
                'errorLoading' => new JsExpression("function () { return 'Esperando por resultados...'; }"),
              ],
              'ajax' => [
                'url' => Url::to(['/urgencias-catalogos/cat-clues']),
                'dataType' => 'json',
                'data' => new JsExpression('function(params) { return {q:params.term}; }')
              ],
            ],
          ]); ?>
        </div>
      </div>
      <div class="row" id="datosReferencia" style="display: none;">
        <div class="col-md-3">
          <?= $form->field($model, 'urgest_fechaEnvia')->widget(DatePicker::classname(), [
            'options' => [
              'autocomplete' => 'off',
            ],
            'removeButton' => false,
            'type' => DatePicker::TYPE_COMPONENT_APPEND,
```


ISSN: 2007-4786
Innovación y Desarrollo Tecnológico
Volumen 13 – Número 2
Abril – Junio 2021

```
            'pluginOptions' => [
              'orientation' => 'bottom left',
              'autoclose' => true,
              'todayHighlight' => true,
              'format' => 'yyyy-mm-dd'
            ]
          ]); ?>
        </div>
        <div class="col-md-3">
          <?= $form->field($model, 'urgest_medicoEnvia')->textInput(['maxlength' => true, 'style' => 'text-transform: uppercase;', 'onkeyup' => 'javascript:this.value=this.value.toUpperCase();']) ?>
        </div>
        <div class="col-md-3">
          <?= $form->field($model, 'urgest_motivoReferencia')->widget(Select2::classname(), [
            'language' => 'es',
            'data' => Yii::$app->params['motivoReferencia'],
            'options' => [
              'placeholder' => '=== SELECCIONE ===',
              'id' => 'sel-motivoReferencia',
              'onchange' => 'campoMotivoReferencia(this)'
            ],
            'pluginLoading' => false,
            'pluginOptions' => [
              'allowClear' => true
            ],
          ]) ?>
        </div>
        <div class="col-md-3" id="otroMotivo" style="display:none;">
          <?= $form->field($model, 'urgest_otroMotivoReferencia')->textInput(['maxlength' => true, 'style' => 'text-transform: uppercase;', 'onkeyup' => 'javascript:this.value=this.value.toUpperCase();']) ?>
        </div>
        <div class="col-md-3">
          <?= $form->field($model, 'urgest_servicioReferencia')->widget(Select2::classname(), [
            'language' => 'es',
            'data' => Yii::$app->params['servicioReferencia'],
            'options' => [
              'placeholder' => '=== SELECCIONE ===',
            ],
            'pluginLoading' => false,
            'pluginOptions' => [
              'allowClear' => true
            ],
          ]) ?>
        </div>
      </div>
    </div>
    <div class="box-footer" align="center">
      <?= Html::submitButton('<span class="fa fa-save"></span> Guardar', ['class' => 'btn btn-app btn-flat bg-olive', 'data' => ['disabled-text' => 'Guardando'] ]); ?>
    </div>
    <?php ActiveForm::end(); ?>
</div>
```

Depending on what is needed in each data entry, the form is variable, the following figure illustrates the code of the create and form file.

ISSN: 2007-4786
Innovación y Desarrollo Tecnológico
Volumen 13 – Número 2
Abril – Junio 2021

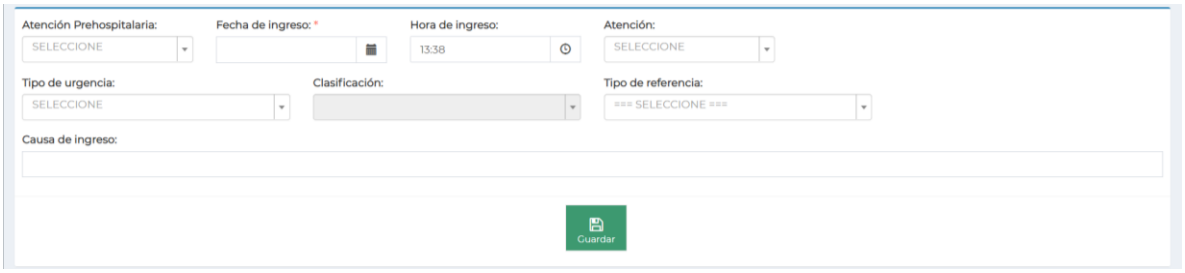

Figure 11. create urgency record

Table one shows the name of the repositories used in this module and the preferred method of installation, adding to the file "composer.json" and then updating so that the extensions are installed or through composer installing one by one.

Repository name	add to file "composer.json"	Install extensions using composer
kartik-v/yii2-krajee-base	"kartik-v/yii2-krajee-base": "^2.0"	composer require kartik-v/yii2-krajee-base "dev-master"
kartik-v/yii2-grid	"kartik-v/yii2-grid": "^3.3"	composer require kartik-v/yii2-grid "dev-master"
kartik-v/yii2-widget-select2	"kartik-v/yii2-widget-select2": "dev-master"	composer require kartik-v/yii2-widget-select2 "dev-master"
kartik-v/yii2-widget-datepicker	"kartik-v/yii2-widget-datepicker": "dev-master"	composer require kartik-v/yii2-widget-datepicker "dev-master"
kartik-v/yii2-widget-depdrop	"kartik-v/yii2-widget-depdrop": "^1.0"	composer require kartik-v/yii2-widget-depdrop "^1.0"
kartik-v/yii2-widget-fileinput	"kartik-v/yii2-widget-fileinput": "^1.0"	composer require kartik-v/yii2-widget-fileinput "^1.0"
fortawesome/font-awesome	"fortawesome/font-awesome": "~4.7"	composer require fortawesome/font-awesome "~4.7"
kartik-v/yii2-widget-timepicker	"kartik-v/yii2-widget-timepicker": "dev-master"	composer require kartik-v/yii2-widget-timepicker "dev-master"
kartik-v/yii2-detail-view	"kartik-v/yii2-detail-view": "dev-master"	composer require kartik-v/yii2-detail-view "dev-master"
wbraganca/yii2-dynamicform	"wbraganca/yii2-dynamicform": "*"	composer require wbraganca/yii2-dynamicform "*"
kartik-v/yii2-tabs-x	"kartik-v/yii2-tabs-x": "@dev"	composer require kartik-v/yii2-tabs-x "@dev"
kartik-v/yii2-widget-datetimepicker	"kartik-v/yii2-widget-datetimepicker": "dev-master"	composer require kartik-v/yii2-widget-datetimepicker "dev-master"
symfony/dom-crawler	"symfony/dom-crawler": "2.8"	composer require symfony/dom-crawler "2.8"
symfony/css-selector	"symfony/css-selector": "2.8"	composer require symfony/css-selector "2.8"
kartik-v/yii2-mpdf	"kartik-v/yii2-mpdf": "dev-master"	composer require kartik-v/yii2-mpdf "dev-master"
kartik-v/yii2-export	"kartik-v/yii2-export": "dev-master"	composer require kartik-v/yii2-export "dev-master"
faryshta/yii2-disable-submit-buttons	"faryshta/yii2-disable-submit-buttons": "~2.0.0"	composer require faryshta/yii2-disable-submit-buttons "~2.0.0"
vova07/yii2-imperavi-widget	"vova07/yii2-imperavi-widget": "*"	composer require vova07/yii2-imperavi-widget "*"

Table 1. Components and Repositories used on this project.

6. CONCLUSIONS

Using the MVC design pattern in the medical emergencies module of the Comprehensive System of the High Specialty Regional Hospital "Dr. Gustavo A. Rovirosa Pérez", is one of the best practices to keep the code unified and structured, in addition to becoming understandable by other programmers, since the MVC allows it in a way that is understood for modifications or updates within what required in the medical emergency module.

Having used Yii2 Framework in the development of this module, allowed to use the MVC design pattern, automate the coding process, speed up development times and understand the process to generate a record of a medical emergency, as well as improve a medical consultation in response times of the system

REFERENCIAS

[1] Application Structure: Application Structure Overview. (n.d.). Retrieved from The Definitive Guide to Yii 2.0: https://www.yiiframework.com/doc/guide/2.0/en/structure-overview

[2] Application Structure: Controllers. (n.d.). Retrieved from The Definite Guide to Yii 2.0: https://www.yiiframework.com/doc/guide/2.0/en/structure-controllers

[3] Beck, K., & Cunningham, W. (n.d.). Using Pattern Languages for Object-Oriented Programs. Retrieved from http://c2.com/doc/oopsla87.html

[4] Bragança, W. (n.d.). Yii2 Dynamic form. Retrieved from Wanderson Bragança: https://wbraganca.com/yii2extensions/yii2-dynamicform/installation

[5] Foundation, M. (n.d.). About MariaDB Server. Retrieved from MariaDB Server: The open source relational database: https://mariadb.org/about/

[6] Guevara, A. (n.d.). Github - Faryshta Yii2 Disable Buttons. Retrieved from Github: https://github.com/Faryshta/yii2-disable-submit-buttons

[7] Handling Requests: Handling Overview. (n.d.). Retrieved from The Definitive Guide to Yii 2.0: https://www.yiiframework.com/doc/guide/2.0/en/runtime-overview

[8] Introduction: About Yii. (n.d.). Retrieved from The Definitive Guide to Yii 2.0: https://www.yiiframework.com/doc/guide/2.0/en/intro-yii

[9] Lloyd, C. (n.d.). Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC) by Steve Burbeck, Ph.D. Retrieved from WayBackMachine: http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html

[10] MVC - MDN Web Docs Glosary: Definitions of Web-related terms. (n.d.). Retrieved from https://developer.mozilla.org/en-US/docs/Glossary/MVC

[11] Sysoev, I. (n.d.). About NGINX. Retrieved from NGINX: https://www.nginx.com/about/

[12] Visweswaran, K. (n.d.). Krajee Yii Extensions A collection of extensions & modules for Yii Framework 2.0. Retrieved from Krajee Yii Extensions: https://demos.krajee.com/

[13] Wilver Omar Figueroa Escudero. (n.d.). Aplicación del patrón de diseño MVC utilizando Yii2 Framework en el desarrollo del módulo de Consulta Externa del Sistema Integral Hospital Rovirosa (SIHR). Retrieved from Innovación y Desarrollo Tecnológico: https://iydt.files.wordpress.com/2021/01/4_2_aplicacion-del-patron-de-diseno-mvc-utilizando-yii2-framework-en-el-desarrollo-del-modulo-de-consulta-externa-del-sistema-integral-hospital-rovirosa.pdf

Author's email: ***iscluisfdoayala@gmail.com***