Code Creator Application Design to Create CRUD Module on Codeigniter Framework

I Wayan Candra Winetra[1], Edhy Sutanta[2*], I Wayan Julianta Pradnyana[2]

[1]Department of Electrical Engineering, Politeknik Negeri Bali, Indonesia
[2]Department of Informatics, Institut Sains & Teknologi AKPRIND Yogyakarta, Indonesia
**Corresponding author: edhy_sst@akprind.ac.id*

ABSTRACT
Codeigniter is a robust PHP framework to help developers build their web applications. This research aims to create a code creator application to automatically generate the CRUD (Create-Read-Update-Delete) module in the Codeigniter framework project to help the developers accelerate the making of the web project, especially in making the back-end module. CRUD module is created using a prototyping model approach, programming architecture using MVC (Model-View-Controller), and the Java language to develop the code creator application. The research conclusion is that the application can produce a CRUD module that works correctly, with printing and export to excel features. Still, the applications cannot handle the relations between tables that will need future work.

Keywords: *Code Creator, Codeigniter, CRUD Module, Framework*

1. INTRODUCTION

Web application development has become an inevitability. Almost all large companies have a web page to manage their resources or reach a broader market. *Codeigniter* is a robust PHP framework created for developers to build dynamic web applications to respond to the rapid development of web-based information systems. The variety of simple and elegant toolkits in *Codeigniter* that are easy to use expected can facilitate developers to develop web applications that are effective, efficient, and guaranteed in terms of security. Even though the Codeigniter framework can help developers develop web-based applications, areas can improve. For example, making CRUD modules for master data or the back-end section usually requires developing web-based applications. The more master data to create, the more time it takes to work on it. The CRUD module on the back-end usually has only a standard function to create, read, update and delete data. The process of each CRUD module of the table data is almost the same, but due to each data's different fields and types, this module still has to be made one by one, which takes time to work.

This research provides a solution by creating a code creator application that can automatically create the CRUD module in the Codeigniter framework without typing the code. Furthermore, the CRUD module code will automatically generate by describing each data table on the application form. It will help developers develop their back-end modules on their web applications.

2. METHODS

Codeigniter Framework

CodeIgniter is a PHP framework developed by Alice Lab and does not require additional configuration. We do not have to use the command line. It is extraordinarily light and usually provides a rich set of libraries for essential works and provides a simple interface and logical design to access this library [1]. According to their web sentence on www.codeigniter.com, this framework is a robust PHP framework with a tiny footprint, built for developers who need a simple and elegant toolkit to create full-featured web applications. Data flows throughout the system on *Codeigniter* are shown in Figure 1. It shows that the *Codeigniter* framework works based on the MVC concept.

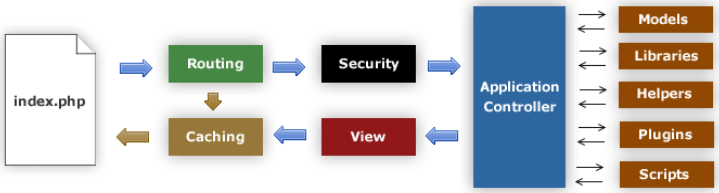

Figure 1. Codeigniter data flow flowchart (www.codeigniter.com, 2022)

Prototyping System Development Model

This research uses a prototyping system development model to develop the code creator application. Prototyping is the rapid development and testing of new applications working models (prototypes) through interaction and repetitive processes commonly used by information system experts and business experts [2]. By prototyping, the software designers and implementers can get valuable feedback from users early in the project [3].

MVC Architecture

MVC programming architecture applies a three-way factoring, whereby objects of different classes take over the operations. The models are those components of the system application that do the work (simulation of the application domain). They are kept quite distinct from views that display aspects of the models. Controllers send messages to the model and provide the interface between the model with its associated views and the interactive user interface devices. Each view is closely associated with a controller, each having precisely one model, but a model may have many view/controller pairs [4]. The Codeigniter framework uses the MVC architecture in its application development scheme. Therefore, every CRUD module created must consist of a model, view, and controller.

JAVA Programming

Java is a programming language developed by James Gosling and a group of Engineers at Sun Microsystems of the USA in 1991 [5]. Java became a computing platform first released by Sun Microsystems in 1995. Java is fast, secure, and reliable. As a result, Java is everywhere, from laptops to datacenters, game consoles to scientific supercomputers, and cell phones to the Internet (java.com). This study uses the Java language to develop the code creator application, wherewith many file access libraries, and object-oriented concepts will greatly facilitate application development.

Use Case Diagram

Figure 2 shows the use case diagram. Actors can create projects, add tables, import tables, download the Codeigniter framework file, create code corresponding to the projects and tables, and automatically run the code in the Codeigniter framework. The user must create to download the Codeigniter framework and create the CRUD code. Adding or importing tables is an option, but at least the user must create a project. However, to create a complete CRUD system, we must add tables to the project, which can be done by creating or importing a table from an existing project or local database. The complete process sequence is to create a new project, add tables by creating or importing a table or a combination, download the Codeigniter framework, create code, and finally run the code.

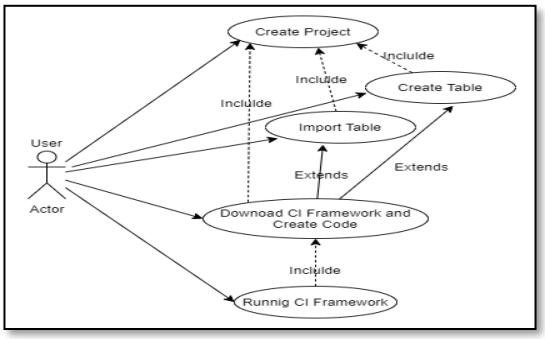

Figure 2. Use case diagram

Activity Diagram

The activity diagram shows three segments of the application process: creating projects and tables, downloading the *Codeigniter* framework and creating CRUD code, and running the generated CRUD code. In the first part, after successfully creating the project, the user can immediately start creating code in the Codeigniter framework. However, the complete process of setting up the project is to add a table which can be done by creating or importing a table. The second part is creating CRUD code in the Codeigniter framework. If the Codeigniter file does not exist, the download process will run, but the process will directly create the CRUD code module in the Codeigniter framework if it already exists. The third part is running the generated code, and the process begins with creating a database on MySql DBMS. If it fails to make the database, the application will display a message to create the database manually and run the code manually. If creating a database is successful or already exists, the following process is running the migration table, the seed menu, and the PHP Spark. Finally, if the process is complete, the browser will be automatically opened, and the web page on localhost will run. Figure 3 shows the activity diagram of the application process.

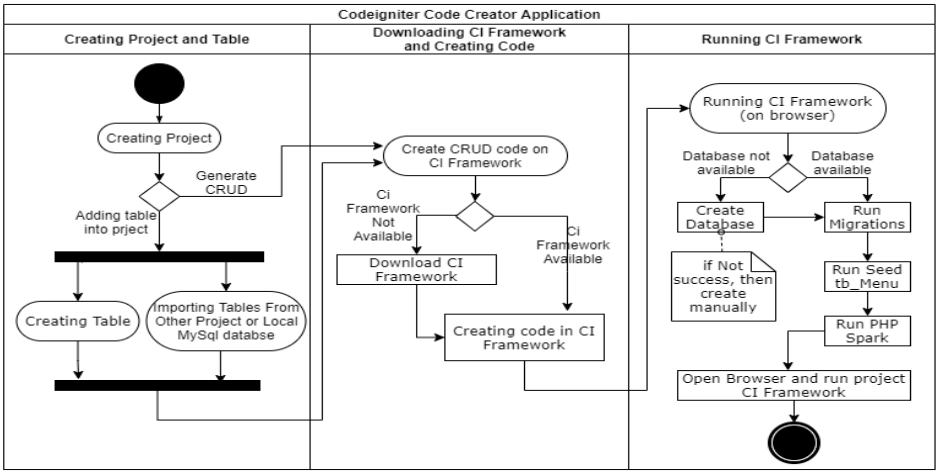

Figure 3. Activity diagram

Class Diagram

The code generator application will consist of three main classes: the project class, the table class, and the field class. The project class contains the name, version, storage location, main menu title, database hostname, database name to be created, database username, database password, project author, author email, and an ArrayList of the table. These tables are master data tables that will create in the project. Next is the table class that the project will own. This class contains the class's name to create the file name, package name, model name, and variable naming on the CRUD module. The table will also have a caption that will be the module title, table's name that will use to create the table name on the database, a column for the number of columns in the table, and a field table that the class used as a column in the table. The field class contains information about the table columns such as name, type, length, validation, primary, auto increment, search, column title, editor type, and Combobox items. Figure 4 shows the class diagram used in the system experiment.

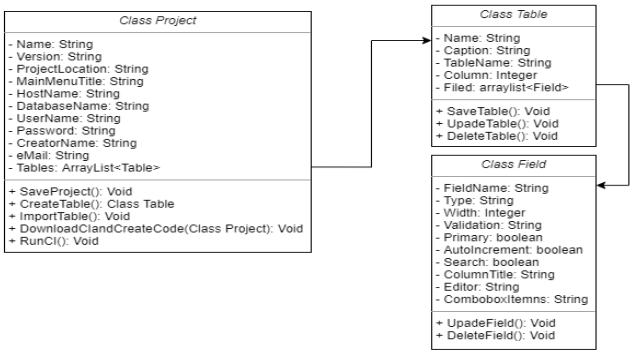

Figure 4. Class diagram

Codeigniter Framework File Structure

The Codeigniter Framework file structure consists of four main folders: the app folder, public folder, system folder, and writable folder. Inside the root framework folder, there are seven files where only two are modified and used. The env file is the environment setting (base URL and database settings) and the spark file, which will run the Codeigniter framework with the PHP spark command. The critical part of the folder to be modified is the app and public folders. Inside the app folder, there are config folders, database migrations and seeds, controllers, models and views (MVC components) in which we will automatically generate CRUD code according to the project. The public folder contains the index.php file, which is the index of the web project. This folder is also the place to store the uploaded files if there is a file upload process and store the asset files, which contain the libraries used by the web page.

Files Modified and Files Created by The Application

Modifying files and generating the code in the specified Codeigniter framework required to create the CRUD system and its menu page in the project. Here are the modified and created files made by the code creator application:

1. Modify the env file

env file located in the root folder is the file to set the Codeigniter framework environment. In this file, the app.baseURL line will modify by changing the base URL to: app.baseURL = http://localhost:8080/. This baseURL setting ensures the web with the Codeigniter framework runs by calling the address http://localhost:8080/ in the browser to run the framework index page. Other part of this file that must be modified is the database section, as follows:

```
database.default.hostname = localhost
database.default.database = dbname
database.default.username = root
database.default.password =
database.default.DBDriver = MySQLi
```

This section sets up the database connection, with database host, database name, username, password, and the database driver, which is set to the MySQL by writing the MySQLi code.

2. Creating .htaccess files

The .htaccess file is created in the root folder of the Codeigniter framework to redirect the web URL, with the following contents:

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php/$1 [L]
```

RewriteEngine On is the command to enable rewrite mode on the apache server. This mode provides the way to modify incoming URL requests dynamically, based on regular expression rules that allow arbitrary mapping URLs to internal URL structures in the way that the developer wants.

RewriteCond %{REQUEST_FILENAME} !-f and !-d is to redirect every request of a certain file type to the index. %{REQUEST_FILENAME} represents the name of the requested file. At the same time, the !-f command means that if the file with the name specified in the browser does not exist, or !-d, i.e., the directory in the browser does not exist, then proceed will be passed to the rewrite rules below this command, which is the rewrite rules to index.php. RewriteRule ^(.*)$ index.php/$1 [L], the command means that if the request matches ^(.+)$ (matches any URL except the server root), it will be rewritten as index.php?url=$1, which means rewritten the request as index.php?URL = olle.

3. Modify the app.php file

This file is in the app/Config/App.php folder. In this file, the application will modify the public $baseURL line (on lines 25-26) by changing the code to: public $baseURL = 'http://localhost:8080/';. $baseURL redirects the call of the index page from the Codeigniter framework to localhost with port 8080 on the browser.

4. Creating CSS file for web administrator design

The next step is to create the web page design with the CSS file in the public/assets/css/style.css folder. All pages and CRUD modules will use this style.css to make all web page uniforms.

5. Creating file migration

Migration consists of files used to create tables in the database. The location of the migration file is in the app\Database\Migrations folder, where the file name format is *yyyyMMddhhmmss_add_tablename.php*, following the Codeigniter framework migration naming form. In this migration, the application must create the file with the *yyyyMMddhhmmss_add_main_menu.php* name to create the mainmenu table in the database. This table will have an id field with autoincrement type, menu_name for the name of the displaying menu on the web page, and a menu link that contains a link to call the page. The most important is the timestamp as the initial name of this file used by the Codeigniter framework to check the updating migrations on the database.

6. Creating seed for mainmenu table

After creating the table mainmenu on migration, the application will create the seed file on the app\Database\Seeds folder to fill the mainmenu table with data representing the menu of the CRUD module on the web page. This seed file will be named mainmenu.php. This seed file will be named mainmenu.php with the code containing the id, name, and the URL of the menu for each table as follows:

```
<?php namespace App\Database\Seeds;
use CodeIgniter\Database\Seeder;
class Mainmenu extends Seeder{
  public function run(){
    $menuData = [
   [
      'id' => 0,
      'menu_name' => 'Class_Name',
      'menu_link' => '/class_name' ], ];
     foreach ($menuData as $item) {
        // inserting data into database table menu
        $this->db->table('main_menu')->insert($item); } } }
```

$menuData is an array of all menus representing the tables added to the project. Data with id=>0 is created because the id on the mainmenu table is an autoincrement. The menu_name added to naming each CRUD module that will appear as the caption of the menu on the web page. The menu_link is the URL of the module to call the home page of each CRUD module.

7. Creating model

The model is created in the app\Models folder, named ModelClassName.php. This model contains the table name that the model uses, the primary key definitions, and the permissions to changes to the data on the table's fields, as in the example:

```
<?php
```

```
namespace App\Models;
use CodeIgniter\Model;
class ModelClassName extends Model{
  protected $table = 'tb_name';
  protected $primaryKey = 'field as primary';
  protected $useTimestamps = false;
  protected $allowedFields = ['field1',' field1',…' field n']; }
```

8. Creating views

The application creates this page in the \app\Views folder to design how the web page will look. A folder will create for each CRUD module following the class name of each module. Inside each created folder will create the home.php, insert.php, update.php, export.php, and print.php file. The home.php file on the root folder \app\Views must be modified because the index.php in the public folder will call this page. The home.php on this root serves as the main page of the CRUD system that the code creator application will create. Here is the code from home.php on the root:

```
<?= $this->extend('template/layout') ?>
<?= $this->section('content') ?>
<div class="text-center mt-5">
   <h3 class="text-center">Welcome to Undagi Code Dashboard</h3>
</div>
<div class="row m-5 pt-5">
  <?php foreach ($menu as $item) : ?>
      <div class="col-sm-3 mb-3">
  <div class="card bg-white rounded shadow">
     <div class="card-body">
       <h5 class="card-title"><i class="fas fa-folder-open fa-4x text-warning "></i></h5>
       <p class="card-text">Manage your data on <?= $item['menu_name'] ?></p>
       <a href="<?= site_url($item['menu_link']) ?>" class="btn btn-outline-success"><?=
                    $item['menu_name'] ?></a>
     </div>
          </div>
       </div>
   <?php endforeach ?>
</div>
<?= $this->endSection() ?>
```

9. Creating controllers

A controller is created for each CRUD by the application. The application will create the controller page in the app\Controllers folder to control the model and the view. A controller is created for each CRUD by the application. This controller also contains codes to create, read, update and delete data from the database. Furthermore, the application also changed the BaseController.php in the app\Controllers folder adding the helper and the affected rows of the database table. This controller is the default controller in the Codeigniter framework. Here is the code of the BaseController.php file:

```
<?php
namespace App\Controllers;
use CodeIgniter\Controller;
class BaseController extends Controller{
    protected $helpers = ['number', 'form', 'url', 'text', 'date'];
    public function initController(\CodeIgniter\HTTP\RequestInterface $request,
      \CodeIgniter\HTTP\ResponseInterface $response, \Psr\Log\LoggerInterface $logger)
    { parent::initController($request, $response, $logger); }
   public function dbAffectedRows(){
```

```
$db = \Config\Database::connect();
return $db->affectedRows(); }
```

10. Index.php in the public folder

The index.php page, which is in the public folder of the Codeigniter framework, does not have to change because the application has modified and created the file in the Codeigniter framework environment to link the pages with this index page.

3. RESULTS AND DISCUSSION

After developing the application by following the steps, the experiment tests the application by creating a web project. Figure 5 shows the results of the experiment. The experiment shows that the application is running well, and it can create a project and add a table to the project. Furthermore, the application successfully copies the Codeigniter framework and creates the CRUD code inside the framework.

The first is to activate the apache server and MySQL DBMS on the localhost and create a MySQL database according to the project. Running the code in the Codeigniter framework can be done manually or directly from the application. The sequence of running the framework is by opening the command prompt, going to the framework folder, running the migrations, sedding the mainmenu table, and running the PHP spark service command by typing the command as below:

```
PHP splash migrate <enter>
PHP spark dB: seed mainmenu
PHP spark service
```

as the spark service runs, open the browser and point the address to the http://localhost:8080/, the CRUD system will run on the browser. Running the framework from the run menu on the application will automatically open the command prompt and run all the sequence commands. This menu will also activate the browsers and point the address to http://localhost:8080/.

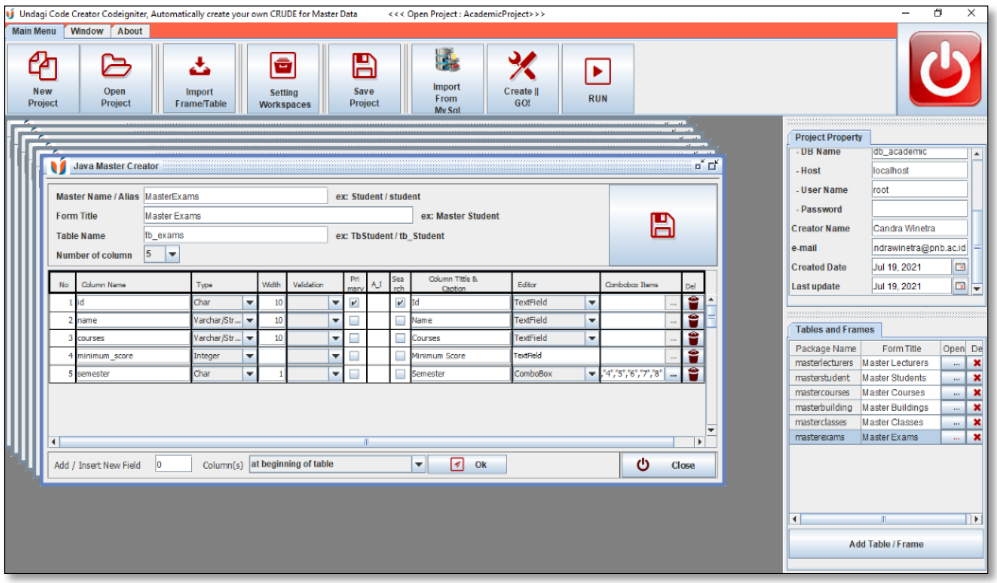

Figure 5. The application interface screenshot

4. CONCLUSION

From the conducted experiments, below are the conclusions of the research:

1. Applications can be made according to the design and run as expected to create the CRUD code system in the Codeigniter framework.

2. The Applications can automatically generate the CRUD systems quickly in the Codeigniter framework without writing the code. The speed of the generating process depends on the number of tables involved in the CRUD system. Generally, it should not take more than 5 minutes to create 100 tables. The resulting CRUD system can work perfectly on the localhost to manage the tables in a MySQL database with insert, delete, update, print and export to excel files facilities.
3. Although the application successfully created and ran the CRUD system, the application has not been able to work on relations between tables on the database.

Further development on the research and applications is needed to deal with the relational database problems so that CRUD in the Codeigniter framework can connect between two or more tables. With better application development, creating the CRUD system, especially for the back-end web user, can be done thoroughly without typing the code and with a faster time to reduce the total time needed to build a complete web system.

REFERENCES

[1] R. Jahagirdar and Y. Puranik, "A Review on Codeigniter. International Journal of Trend in Scientific Research and Development," vol. 2(4), pp. 1124–1129, 2018.
[2] A. Susanto and Meiryani, "System Development Method with The Prototype," *Res. Int. J. Sci. Technol.*, vol. 8(7), pp. 141–144, 2019.
[3] A. Saxena and P. Upadhyay, "Waterfall vs. Prototype: Comparative Study of SDLC," *Imp. J. Interdiscip. Res.*, vol. 2(6), 2016.
[4] G. . Krasner and S. Pope, "A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System," *J. Object-oriented Program.*, vol. 1(3), 1988.
[5] O. Ikedilo, P. Osisikankwu, and Madubuike C, "A Critical Evaluation of Java as a Good Choice for Introductory Course," *Int. J. Res.*, vol. 2(12), pp. 847–853, 2015.