

# Daftar Isi

Daftar Isi.....	1
BAB 1 .....	4
1.1 Arsitektur Enterprise Application .....	5
1.2 Karakteristik Enterprise Application .....	7
BAB 2 .....	8
2.1 Pengenalan Framework MVC .....	9
2.2 Komponen Framework MVC .....	10
2.3 Keuntungan Menggunakan Framework MVC .....	12
BAB 3 .....	13
3.1. Pengenalan Codeigniter .....	14
3.2. Keunggulan Codeigniter .....	15
3.3. Alur Kerja Codeigniter .....	15
3.4. Teknologi yang dibutuhkan .....	16
3.5. Instalasi Codeigniter.....	17
3.6. Struktur Folder.....	20
3.7. Konfigurasi.....	22
3.8. Memahami Routing .....	23
3.8.1. Segmen URI .....	24
3.8.2. Contoh Routing .....	24
3.8.3. Membuat Routing Baru .....	25
3.8.4. Membuat Closure .....	27
3.8.5. Membuat Placeholder.....	27
BAB 4 .....	30
4.1. Validasi Form .....	31
4.2. Pengujian Form.....	36
BAB 5 .....	44
5.1. Konfigurasi Database .....	45
5.1.1. Membuat Database .....	46
5.1.2. Memasukkan Data ke Dalam Database .....	49
5.2. Membuat List Data .....	53
5.2.1. Membuat Model.....	53

5.2.3. Membuat View .....	55
5.2.4. Membuat Routes .....	58
5.3. Membuat Add Data .....	59
5.3.1. Edit Controller.....	59
5.3.2. Menambah View add.php.....	60
5.3.3. Membuat Routes Untuk Add dan Store .....	62
5.4. Membuat Edit Data.....	63
5.4.1. Edit Controller.....	63
5.4.2. Menambah View edit.php.....	65
5.4.3. Membuat Routes Untuk Edit dan Update .....	66
5.5. Membuat Hapus Data .....	68
5.5.1. Edit Controller.....	68
5.5.2. Membuat Routes Untuk Delete.....	69
5.6. Membuat Info Detail Data.....	70
5.6.1. Edit Controller.....	70
5.6.2. Menambah View view.php.....	70
5.6.3. Menambah link pada list.php .....	72
5.6.4. Membuat Routes Untuk Detail Data.....	72
BAB 6 .....	74
6.1. Prinsip Manajemen Keamanan.....	75
6.1.1. Confidentiality .....	75
6.1.2. Integrity .....	76
6.1.3. Availability .....	76
6.2 Identification .....	77
6.3. Otentikasi .....	77
6.4. Membuat Form Register .....	78
6.4.1. Membuat Tabel Users .....	78
6.4.2. Membuat Model Users .....	80
6.4.3. Membuat Controller Register.....	81
6.4.4. Membuat View register.php.....	83
6.4.5. Membuat Routes Untuk Index dan Save.....	84
6.5. Membuat Form Otentikasi .....	86
6.5.1. Membuat Controller Login dan Dashboard.....	86

6.5.2. Membuat View Login dan Dashboard.....	88
6.5.3. Membuat file Filters Auth.php .....	90
6.5.4. Membuat Routes Login dan Dashboard.....	91
BAB 7 .....	94
7.1. Pengenalan API.....	95
7.2. Pengenalan REST.....	95
7.3. Design EndPoint RESTful API .....	96
7.4. Membuat Tabel Subject .....	97
7.5. Memasukkan Model Subject .....	98
7.6. Memasukkan Data ke Dalam Tabel Subject .....	99
7.7. Membuat Controller SubjectController .....	101
7.7. Konfigurasi Routes.php.....	104
7.8. Mengaktifkan Cross-Origin Resource Sharing (CORS).....	104
7.9. Pengujian RESTful API Menggunakan Postman.....	106
7.9.1. Instalasi Postman .....	106
7.9.2. Pengujian.....	107
DAFTAR PUSTAKA.....	113

# BAB 1

# PENGENALAN

# ENTERPRISE

# APPLICATION

## Tujuan instruksional:

- Memahami Arsitektur Aplikasi Enterprise

## 1.1 Arsitektur Enterprise Application

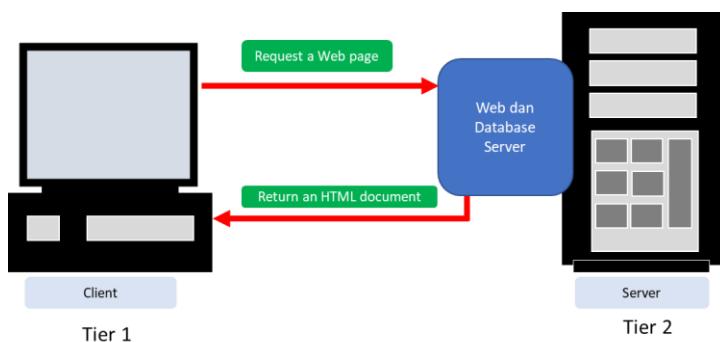
Enterprise mengacu pada suatu organisasi atau individu sebagai suatu kesatuan, yang bekerja bersama-sama untuk mencapai beberapa tujuan umum. Enterprise berkaitan erat dengan B2B (Business to Business) dan B2C (Business to Customer).

Fowler (Patterns of Enterprise Application ) menjelaskan bahwa enterprise application secara teknis memang tidak serumit jenis software lainnya. Sebagai contoh, software untuk industri telekomunikasi yang memiliki fitur untuk berinteraksi dengan perangkat keras dan juga memiliki fitur multithreading yang rumit. Tetapi di sisi lain, enterprise application lebih sulit untuk dikembangkan karena enterprise application seringkali harus bekerja dengan data yang rumit yang berproses dalam aturan bisnis yang rumit. Contoh dari enterprise application adalah sistem penggajian, sistem pencatatan data pasien, sistem tracking pengiriman, sistem analisis biaya dan lain sebagainya. Contoh software yang bukan enterprise application adalah software untuk mengelola bahan bakar pada mobil, word processor, pengendali lift, sistem operasi, games dan lain sebagainya.

Ketika membangun enterprise application, pada umumnya akan identik dengan sekumpulan kode program. Tidak seperti aplikasi biasa lainnya, yang pada dasarnya semua kode program disimpan pada satu tempat, enterprise application justru menyebarkan kode programmnya pada banyak mesin. Pada dasarnya enterprise application dibagi menjadi beberapa tingkatan, yaitu:

- 2-tier application

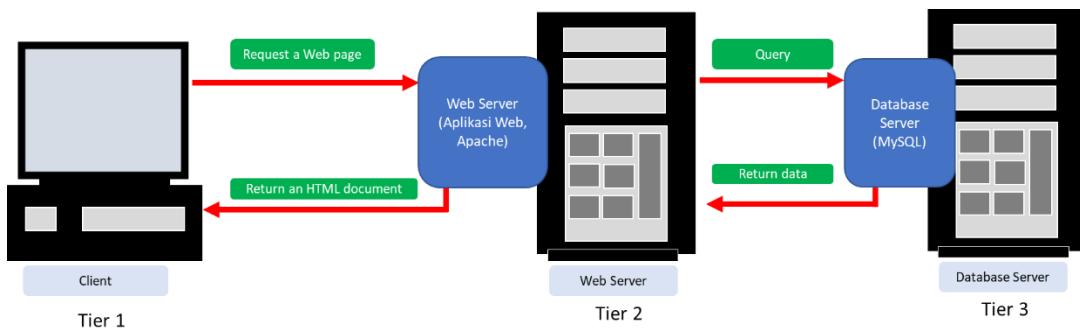
Dalam arsitektur 2-tier, klien berada di tingkat pertama. Server database dan server aplikasi web berada di mesin server yang sama, yang merupakan lapis kedua. Tingkat kedua ini menyajikan data dan menjalankan logika bisnis untuk aplikasi web. Organisasi yang mendukung arsitektur ini biasanya lebih suka mengkonsolidasikan kapabilitas aplikasi dan kapabilitas server database mereka pada satu tingkat. Tingkat kedua bertanggung jawab untuk menyediakan ketersediaan, skalabilitas, dan karakteristik kinerja untuk lingkungan web organisasi.



Gambar 1.1 2-tier application

- 3-tier application

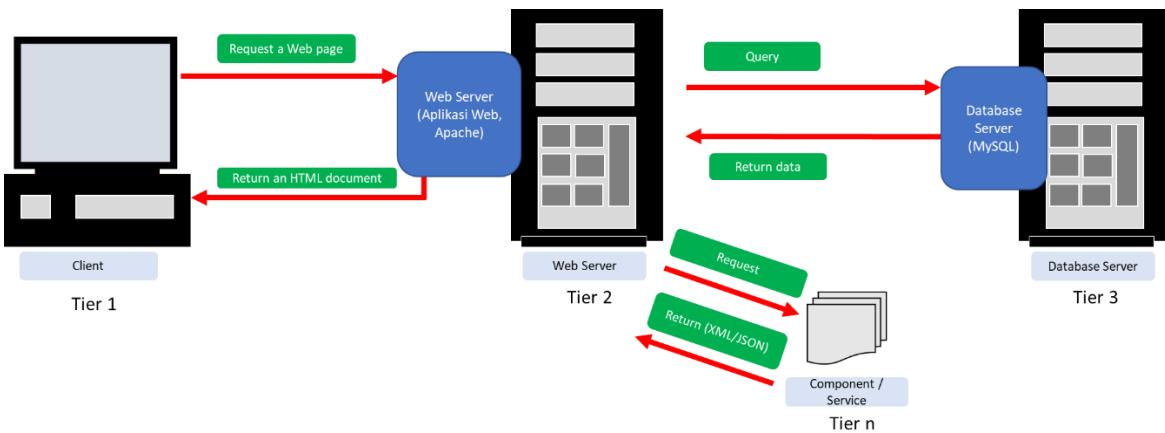
Dalam arsitektur 3-tier, server database tidak berbagi mesin server dengan server aplikasi web. Klien berada di tingkat pertama seperti dalam arsitektur 2-tier. Di tingkat ketiga, server database melayani data. Untuk alasan kinerja, server database biasanya menggunakan prosedur untuk menangani beberapa logika bisnis. Server aplikasi berada di tingkat kedua. Server aplikasi menangani porsi logika bisnis yang tidak memerlukan fungsionalitas yang disediakan server database. Dalam pendekatan ini, komponen perangkat keras dan perangkat lunak dari tingkatan kedua dan ketiga berbagi tanggung jawab atas ketersediaan, skalabilitas, dan karakteristik kinerja lingkungan web.



Gambar 1.2 3-tier Application

- N-tier application

n-tier application memiliki ukuran yang jauh lebih kompleks dan memiliki layer yang berbeda. Dalam arsitektur n-tier, objek aplikasi didistribusikan ke beberapa tingkatan logis.



Gambar 1.3 n-tier Application

## 1.2 Karakteristik Enterprise Application

Menurut Martin Fowler, sebagian dari karakteristik Enterprise application adalah:

1. Enterprise applications usually involve persistent data. (Enterprise application bisaanya melibatkan data yang tetap).
2. There's usually a lot of data—a moderate system will have over 1 GB of data organized in tens of millions of records—so much that managing it is a major part of the system. (Terdapat banyak sekali data. Begitu banyak, hingga pengelolaan data adalah bagian utama dari sistem tersebut).
3. Usually many people access data concurrently. (Biasanya digunakan oleh banyak pengguna untuk mengakses data dalam waktu bersamaan).
4. With so much data, there's usually a lot of user interface screens to handle it. (Dengan begitu banyak data, umumnya terdapat banyak sekali antar muka sistem untuk menanganiinya).
5. Usually they need to integrate with other enterprise applications scattered around the enterprise. (Biasanya butuh diintegrasikan dengan enterprise application lain yang terdapat dalam organisasi).
6. Even if a company unifies the technology for integration, they run into problems with differences in business process and conceptual dissonance with the data. (Walaupun perusahaan dapat menyatukan teknologi agar dapat terintegrasi, perusahaan menghadapi masalah perbedaan dalam proses bisnis dan ketidakcocokan konseptual pada data).
7. Complex business "illogic" that makes business software so difficult. (Proses bisnis yang diluar ketentuan logis akan membuat software bisnis menjadi rumit).

## LATIHAN

1. Sebutkan contoh masing-masing aplikasi enterprise untuk setiap tier yang ada di dunia saat ini!!

# BAB 2

# FRAMEWORK

# MODEL VIEW

# CONTROLLER

**Tujuan instruksional:**

- Memahami Framework MVC

## 2.1 Pengenalan Framework MVC

Dalam membangun aplikasi web, seringkali berhadapan dengan tugas-tugas pemrograman seperti berikut :

- Menangkap request dari browser.
- Memproses request, melakukan business logic, dan menampilkan dynamic page yang sesuai.
- Menerima input berupa HTML Form.
- Melakukan validasi HTML Form.
- Memproses input dari Form.
- Melakukan akses database untuk mengambil data, menambah data, atau mengubah data.

Sering kali antara business logic dan user interface digabung dalam satu file yang sama. Hal ini tidak masalah jika diterapkan pada aplikasi yang sederhana. Tapi, jika aplikasi menjadi besar dan kompleks, akan menimbulkan masalah bila terjadi perubahan pada salah satu komponennya. Solusi permasalahan ini adalah menggunakan Framework (Kerangka Kerja) Model-View-Controller (MVC) dalam pengembangannya. Pada bab ini akan dibahas mengenai dasar-dasar konsep arsitektur MVC.

MVC merupakan konsep arsitektur aplikasi yang memisahkan model data, user interface, dan business logic sebagai komponen yang berbeda, sehingga modifikasi komponen menjadi lebih mudah. Pada aplikasi dimana code user interface tergabung dengan business logicnya, jika terjadi perubahan pada user interface atau business logic maka keduanya akan saling mempengaruhi. Misalnya sebuah file php dimana berisi code user interface dan code business logic untuk mengakses database. Ketika ingin mengubah user interface nya, maka business logic-nya juga harus ikut menyesuaikan. Itu berarti harus merombak seluruh code dalam file php tersebut, yang tentunya akan sangat merepotkan.

Disamping itu tidak semua programmer bisa melakukan desain atau sebaliknya, seorang desainer web belum tentu bisa melakukan pemrograman. Jika user interface dan business logic digabung maka diperlukan orang yang bisa melakukan program dan desain. Akan tetapi, jika aplikasi terlalu besar, hal ini tentunya akan sangat menyulitkan.

Perubahan yang terjadi pada komponen yang satu tidak akan terlalu mempengaruhi komponen lainnya secara signifikan. Paling tidak dengan memakai arsitektur MVC, memerlukan lebih sedikit penyesuaian jika terjadi perubahan. Selain itu, seorang desainer bisa focus pada user interface aplikasi dan programmer bisa focus pada business logic-nya.

Untuk aplikasi yang besar, implementasi sebuah pola dapat dipermudah dengan menggunakan third-party framework. Frameworks tersebut menyediakan detail terkait (request, konfigurasi, dan sebagainya) sehingga kita dapat berkonsentrasi pada hal lain yang lebih penting. Frameworks tersebut juga menyediakan fungsi-fungsi tambahan. Penerapan MVC akan sangat membantu maintenance atau pengembangan yang selanjutnya. Misal, jika hanya aplikasi web apalagi dengan halaman yang statis, tidak perlu menggunakan pola MVC ini. Tapi, untuk aplikasi yang besar, misal aplikasi ERP, maka penggunaan konsep MVC akan sangat membantu sekali.

## 2.2 Komponen Framework MVC

Ada tiga layer utama dalam arsitektur web MVC, yang disebut sebagai best-practices dan banyak diadopsi oleh para arsitek aplikasi yaitu Layer Model, View, dan Controller.

### 1. Layer Model

Model adalah tempat dimana seluruh logika bisnis dari suatu bisnis disimpan. Logika bisnis dapat berupa bagaimana cara berinteraksi dengan data, juga bagaimana merepresentasikan data ke dalam tampilan webview. Logika bisnis juga berisi persyaratan proses bisnis jika aplikasi menggunakan pihak ketiga dalam menjalankan bisnis. Singkatnya, Model bertugas untuk mengatur, menyiapkan, memanipulasi dan mengorganisasikan data (dari database) sesuai dengan instruksi dari controller.

Model, biasanya berhubungan dengan data dan interaksi ke database atau web service. Model juga merepresentasikan struktur data dari aplikasi yang bisa berupa basis data maupun data lain, misalnya dalam bentuk file teks, file XML maupun web service. Biasanya di dalam model akan berisi class dan fungsi untuk mengambil, melakukan update dan menghapus data website. Oleh karena itu, model memiliki hubungan dengan perintah-perintah SQL.

### 2. Layer View

View adalah tempat semua elemen antarmuka aplikasi yang akan dilihat oleh end user. View biasanya berisi markup HTML, CSS maupun javascript. View juga bertugas untuk menyajikan informasi kepada user sesuai dengan instruksi dari controller. Oleh karena nya, tampilan view tidak berisi kode perintah yang berhubungan dengan koneksi data. Tampilan view yang ditampilkan pada masing-masing user walaupun dalam keadaan menggunakan fitur dan fungsi yang sama, tetapi memiliki kemungkinan mendapatkan tampilan yang berbeda-beda.

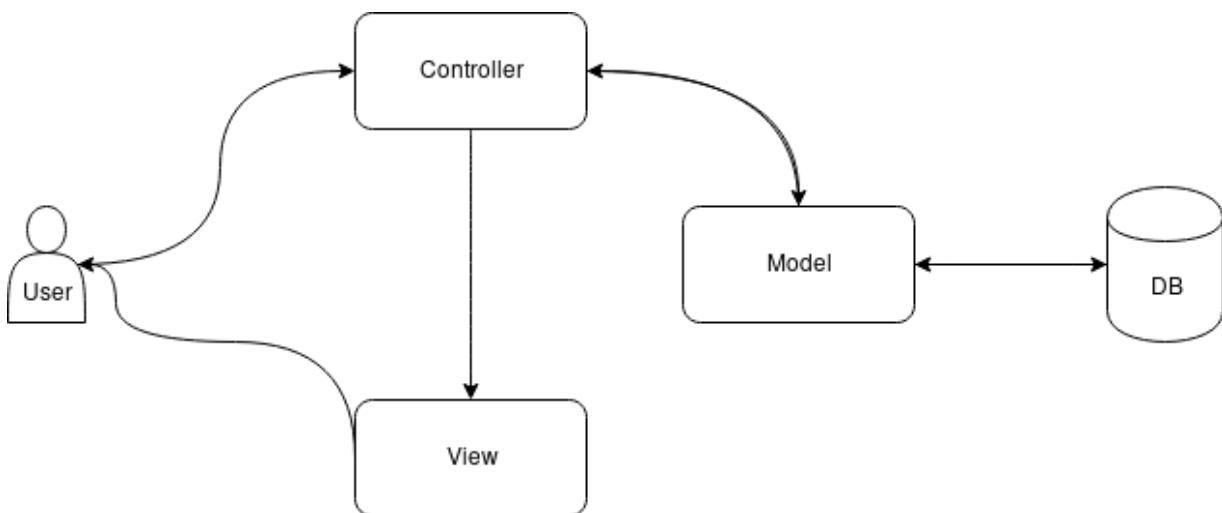
View, merupakan bagian yang menangani presentation logic. Pada suatu aplikasi web bagian ini biasanya berupa file template HTML, yang diatur oleh controller. View berfungsi untuk menerima dan merepresentasikan data hasil dari model dan controller kepada user. View tidak memiliki akses langsung terhadap bagian model.

### 3. Layer Controller

Controller adalah tempat dimana semua elemen yang menjadi penghubung antara model dan view berada. Controller bertugas untuk mengisolasi perintah – perintah yang berhubungan dengan koneksi data karena semuanya telah diserahkan pada tampilan model. Controller juga berperan dalam menjalankan perintah – perintah berupa permintaan dari user melalui view untuk dijalankan dan memanggil model untuk akses database. Namun, terkadang permintaan dari user tidak selalu memerlukan aksi dari model. Misalnya seperti menampilkan halaman form untuk registrasi user. Tugas lain controller juga menyediakan penanganan terhadap kesalahan/error serta menyediakan fungsi untuk validasi terhadap input.

Controller, merupakan bagian yang mengatur hubungan antara bagian model dan bagian view. Pada controller terdapat class-class dan fungsi-fungsi yang memproses permintaan dari View ke dalam struktur data di dalam model. Controller juga tidak boleh berisi kode untuk mengakses basis data Karena tugas megakses data telah diserahkan kepada model. Tugas controller adalah menyediakan berbagai variable yang akan ditampilkan di view, memanggil model untuk melakukan akses ke basis data, menyediakan penanganan kesalahan/error, mengerjakan proses logika dari aplikasi serta melakukan validasi atau cek terhadap input.

Proses dalam MVC diawali dengan controller yang memproses permintaan end user dari view. Kemudian controller akan memanggil model untuk melakukan logika bisnis sesuai dengan permintaan. Model akan memberikan akses terhadap database dan memberikan data yang sesuai. Setelahnya, data yang diperoleh akan diteruskan controller kepada view untuk ditampilkan kepada end user yang sesuai dengan permintaan. Lihat Gambar 2.1.



Gambar 2.1 Model-View-Controller

Berikut adalah alur kerja aplikasi web ketika user mengunjungi salah satu halaman yaitu :

1. Browser berhubungan dengan server untuk akses halaman.
2. Request (permintaan) browser ditangani oleh bagian Controller dari kode kita.
3. Controller akan melakukan pemanggilan ke Model untuk mendapatkan data yang relevan, dan kemudian mempersiapkan data tersebut untuk ditampilkan.
4. Controller memberikan data yang diperlukan kepada view.
5. View menampilkan data dan berbagai elemen antarmuka tambahan yang diperlukan.

Dengan menggunakan MVC pattern, banyak framework yang telah dibuat. Salah satunya adalah Framework MVC dalam PHP yaitu Code Igniter. Arsitektur MVC secara sederhana dirancang dan diadaptasi dalam penggunaan Web-Application.

## 2.3 Keuntungan Menggunakan Framework MVC

Dengan menggunakan MVC dalam membangun sebuah aplikasi, seorang programmer akan dimudahkan dengan beberapa keuntungan :

- a. Programmer tidak perlu membuat kodingan-kodingan yang berulang-ulang. Cukup melakukan reuse terhadap program yang telah ada.
- b. Source code secara otomatis akan mengikuti struktur file yang ada di framework tersebut sehingga memudahkan manajemen source code
- c. Kebanyakan perusahaan sekarang membutuhkan programmer yang bisa menggunakan framework MVC, artinya peluang kerja semakin besar dibanding yang ngoding dari awal
- d. Setiap bagian memiliki tanggung jawab masing-masing. Ingin mengubah query yang digunakan agar menjadi lebih cepat? Langsung saja edit kode pada bagian Model. Designer juga dapat bekerja pada bagian view jika ingin mengubah tampilan, tanpa perlu takut merusak keseluruhan kode. Pastinya hal seperti ini akan mempercepat dan mempermudah pengembangan web kita.

### LATIHAN

1. Sebutkan 3 contoh framework MVC yang menggunakan PHP

# BAB 3

## PENGENALAN CODEIGNITER

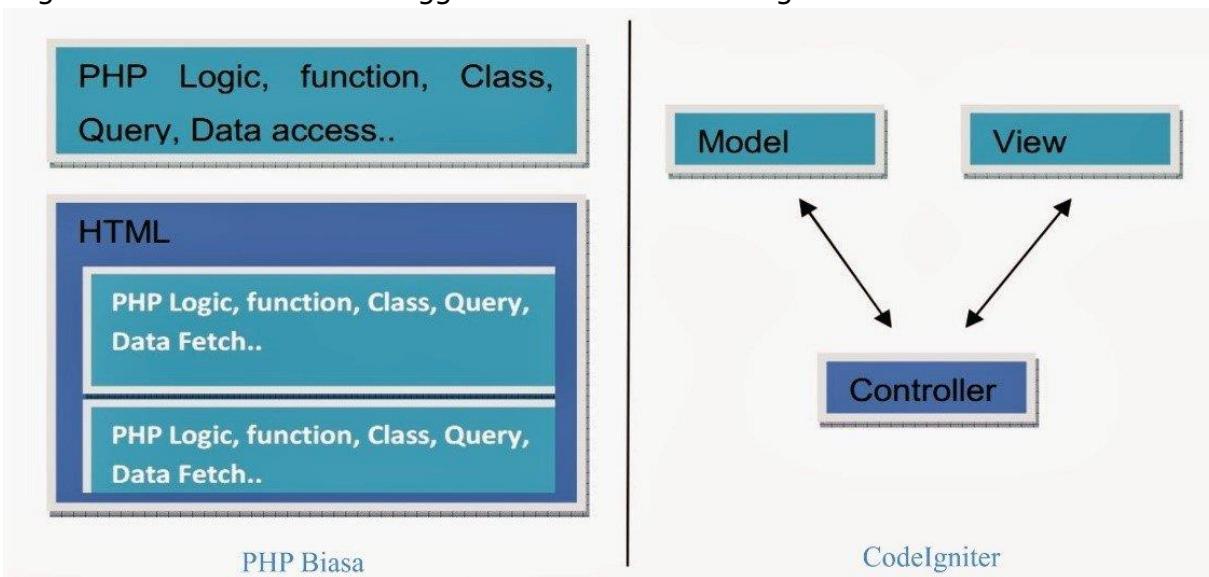
### Tujuan instruksional:

- Memahami cara kerja Framework MVC CodeIgniter

### 3.1. Pengenalan Codeigniter

CodeIgniter adalah salah satu MVC Framework populer yang digunakan untuk mengembangkan situs web menggunakan bahasa pemrograman PHP. CodeIgniter bersifat open-source sehingga setiap orang bebas untuk mengembangkannya. CodeIgniter dikenal memiliki banyak fitur dan kelebihan tersendiri berupa meningkatkan kecepatan dari situs web yang dikembangkan. Library yang dimiliki CodeIgniter sangat banyak dan membantu membangun struktur pemrograman yang rapi dari segi kode maupun struktur file phpnnya. CodeIgniter juga dapat mencegah berbagai serangan terhadap situs web.

CodeIgniter menjadi sebuah framework PHP dengan model MVC (Model, View, Controller) untuk membangun website dinamis. CodeIgniter juga memiliki dokumentasi yang super lengkap disertai dengan contoh implementasi kodenya. Dokumentasi yang lengkap inilah yang menjadi salah satu alasan kuat mengapa banyak orang memilih CodeIgniter sebagai framework pilihannya. CodeIgniter pertama kali dikembangkan pada tahun 2006 oleh Rick Ellis. Dengan logo api yang menyala, CodeIgniter dengan cepat "membakar" semangat para web developer untuk mengembangkan web dinamis dengan cepat dan mudah menggunakan framework PHP yang satu ini. Untuk lebih memahami, dapat melihat diagram perbandingan antara penggunaan PHP dengan cara native dan cara menggunakan framework CodeIgniter.



Gambar 3.1 Perbandingan PHP Biasa dengan CodeIgniter

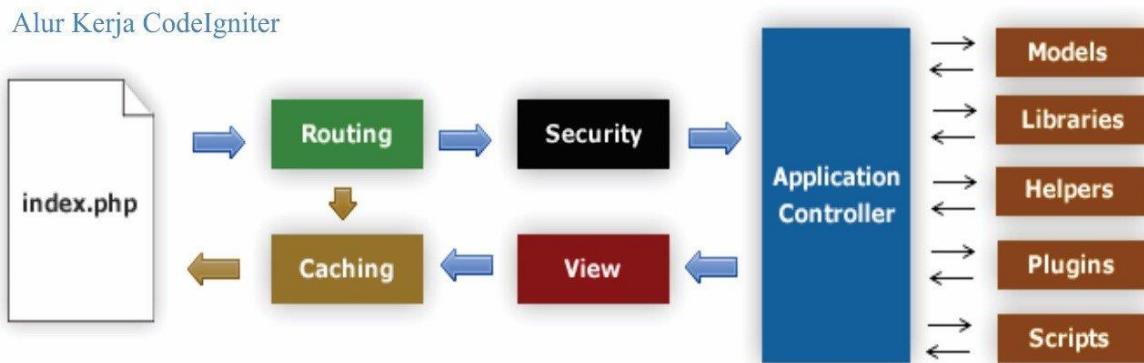
Dalam modul ini, kita menggunakan versi CodeIgniter 4. Adapun kebutuhan dalam menjalankan frawework ini adalah versi minimal PHP 7.2 +.

### 3.2. Keunggulan Codeigniter

CodeIgniter sangat ringat, terstruktur dan mudah dipelajari. Adapun beberapa keunggulan lainnya yang ditawarkan oleh codeigniter adalah sebagai berikut:

- Codeigniter adalah framework PHP yang bersifat open-source.
- Memiliki dokumentasi yang lengkap. Dokumentasi ini berisi berisi pengantar, tutorial, bagaimana panduan penggunaan, serta referensi dokumentasi untuk komponen-komponennya.
- Menyediakan pustaka (Library), fungsi, dan helper yang lengkap.
- Kinerja aplikasi yang dibuat menggunakan framework codeigniter berjalan sangat cepat.
- Codeigniter menggunakan pola desain Model-View-Controller (MVC) sehingga satu file tidak terlalu berisi banyak kode. Hal ini menjadikan kode lebih mudah dibaca, dipahami, dan dipelihara di kemudian hari.
- Scalable and less configuration
- Codeigniter memiliki security yang handal seperti xss filtering, session, encryption, dan lain-lain.
- Codeigniter mendukung banyak RDBMS (Relational Database Management System).
- Codeigniter pada dasarnya menganut Clean URL dan mendukung SEO (Search Engine Optimazation).
- Codeigniter memiliki forum dan komunitas yang besar dan tersebar sehingga membantu para pengembang web untuk memecahkan permasalahan yang dihadapi

### 3.3. Alur Kerja CodeIgniter



Gammbar 3.2 Alur Kerja CodeIgniter

2. Index.php: Index.php disini berfungsi sebagai file pertama dalam program yang akan dibaca oleh program.

3. Router: Router akan memeriksa HTTP request untuk menentukan hal apa yang harus dilakukan oleh program.
4. Cache File: Apabila dalam program sudah terdapat "cache file" maka file tersebut akan langsung dikirim ke browser. File cache inilah yang dapat membuat sebuah website dapat dibuka dengan lebih cepat. Cache file dapat melewati proses yang sebenarnya harus dilakukan oleh program codeigniter.
5. Security: Sebelum file controller di load keseluruhan, HTTP request dan data yang disubmit oleh user akan disaring terlebih dahulu melalui fasilitas security yang dimiliki oleh codeigniter.
6. Controller: Controller akan membuka file model, core libraries, helper dan semua resources yang dibutuhkan dalam program tersebut.
7. View: Hal yang terakhir akan dilakukan adalah membaca semua program yang ada dalam view file dan mengirimkannya ke browser supaya dapat dilihat. Apabila file view sudah ada yang di "cache" maka file view baru yang belum ter-cache akan mengupdate file view yang sudah ada.

### 3.4. Teknologi yang dibutuhkan

Tools yang digunakan dalam belajar CodeIgniter :

1. Web Server dan Database

Web server merupakan salah satu kebutuhan yang digunakan oleh website yang mempunyai kapasitas penyimpanan yang besar dan juga untuk trafik yang besar. Web server berfungsi untuk mentransfer seluruh aspek pemberkasan dalam sebuah halaman web termasuk yang dalam berupa teks, video, gambar dan lainnya.

Salah satu contoh dari Web Server adalah Apache. Apache merupakan web server yang paling banyak digunakan di Internet. Adapun contoh lain dari Web Server yaitu :

- a. Apache Tomcat
- b. Internet Information Services (IIS)
- c. Glassfish Web Server

Database adalah sekumpulan data yang dikelola sedemikian rupa berdasarkan ketentuan tertentu yang saling terkait. Adanya database membantu pengguna dalam memperoleh kemudahan mencari, menyimpan dan menghapus data. Adapun beberapa contoh dari database yaitu :

- a. MySQL / MariaDB
- b. Microsoft SQL Server
- c. Oracle
- d. PostgreSQL

Pada modul ini, kita akan menggunakan XAMPP yang merupakan perangkat lunak open source yang mendukung banyak sistem operasi dan merupakan kompilasi dari beberapa program. Fungsinya adalah sebagai server yang berdiri sendiri, yang terdiri atas program Apache HTTP Server, MySQL database (sekarang MariaDB), dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Untuk mengunduh perangkat lunak ini gunakan tautan: <https://www.apachefriends.org/download.html>

## 2. CodeIgniter

Berkas codeIgniter yang dapat diunduh di internet.

## 3. Composer

Berkas codeIgniter yang dapat diunduh di internet. Composer adalah package-manager (di level aplikasi) untuk bahasa pemrograman PHP yang menyediakan format standar untuk mengelola dependensi PHP dan pustaka-pustaka yang diperlukan. Composer dikembangkan oleh Nils Adermann dan Jordi Boggiano. Kita bisa mengunduh composer di alamat <https://getcomposer.org/download/>

## 4. Text Editor

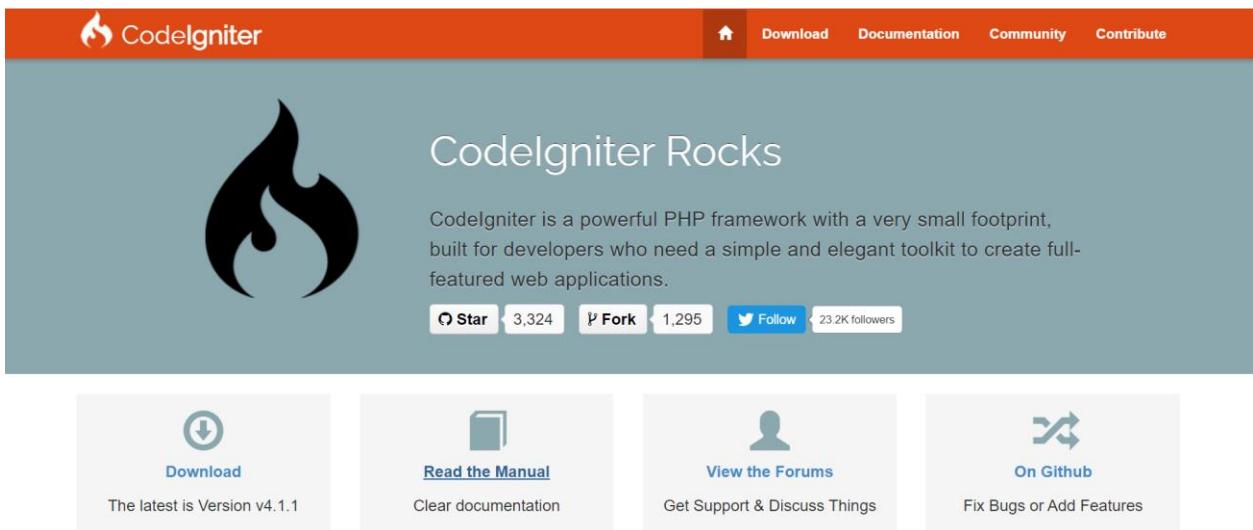
Text Editor adalah suatu perangkat lunak yang digunakan untuk membuat, mengubah atau mengedit program komputer dari bahasa pemrograman tertentu. Text Editor memiliki fitur-fitur sangat kecil dan sederhana. Namun ada juga beberapa text editor kini sudah menawarkan fungsi luas dan kompleks. Berikut contoh dari aplikasi text editor yang dapat digunakan :

- a. Notepad++
- b. Sublime-Text
- c. Atom
- d. Vim
- e. Visual Studio Code

Pada modul ini, kita akan menggunakan Visual Studio Code. Aplikasinya bisa diunduh di halaman <https://code.visualstudio.com/Download>

## 3.5. Instalasi Codeigniter

CodeIgniter dapat diunduh dari <https://codeigniter.com/> dengan memilih menu Download.



Gambar 3.3 Tampilan halaman CodeIgniter

Dalam CodeIgniter 4, kita juga bisa melakukan instalasi dengan cara berbeda yaitu dengan menggunakan bantuan Composer. Kehadiran composer membantu programmer dalam membuat kode menjadi lebih terstruktur dan rapi.

Langkah-langkah instalasi:

1. Buka Command Prompt dan ketikkan perintah composer. Hasilnya seperti berikut:

```
C:\ Administrator: Command Prompt
Microsoft Windows [Version 10.0.18363.1379]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\ASUS>composer
   / \
  /  \
 / \  \
/ \ / \  \
\ / \ / \  \
 \ / \ / \  \
  \ / \ / \  \
   \ / \ / \  \
    \ / \ / \  \
     \ / \ / \  \
      \ / \ / \  \
       \ / \ / \  \
        \ / \ / \  \
         \ / \ / \  \
          \ / \ / \  \
           \ / \ / \  \
            \ / \ / \  \
             \ / \ / \  \
              \ / \ / \  \
               \ / \ / \  \
                \ / \ / \  \
                 \ / \ / \  \
                  \ / \ / \  \
                   \ / \ / \  \
                    \ / \ / \  \
                     \ / \ / \  \
                      \ / \ / \  \
                       \ / \ / \  \
                        \ / \ / \  \
                         \ / \ / \  \
                          \ / \ / \  \
                           \ / \ / \  \
                            \ / \ / \  \
                             \ / \ / \  \
                              \ / \ / \  \
                               \ / \ / \  \
                                \ / \ / \  \
                                 \ / \ / \  \
                                  \ / \ / \  \
                                   \ / \ / \  \
                                    \ / \ / \  \
                                     \ / \ / \  \
                                      \ / \ / \  \
                                       \ / \ / \  \
                                        \ / \ / \  \
                                         \ / \ / \  \
                                          \ / \ / \  \
                                           \ / \ / \  \
                                            \ / \ / \  \
                                             \ / \ / \  \
                                              \ / \ / \  \
                                               \ / \ / \  \
                                                \ / \ / \  \
                                                 \ / \ / \  \
                                                  \ / \ / \  \
                                                   \ / \ / \  \
                                                    \ / \ / \  \
                                                     \ / \ / \  \
                                                      \ / \ / \  \
                                                       \ / \ / \  \
                                                        \ / \ / \  \
                                                         \ / \ / \  \
                                                          \ / \ / \  \
                                                           \ / \ / \  \
                                                            \ / \ / \  \
                                                             \ / \ / \  \
                                                              \ / \ / \  \
                                                               \ / \ / \  \
                                                                \ / \ / \  \
                                                                 \ / \ / \  \
                                                                  \ / \ / \  \
                                                                   \ / \ / \  \
                                                                    \ / \ / \  \
                                                                     \ / \ / \  \
                                                                      \ / \ / \  \
                                                                       \ / \ / \  \
                                                                        \ / \ / \  \
................................................................
```

Composer version 2.0.9 2021-01-27 16:09:27

Jangan lupa untuk cek versi php yang digunakan karena CodeIgniter 4 bekerja dengan versi PHP 7.2+. Kita bisa menggunakan perintah **php -v** untuk melihat versi PHP

```
C:\ Administrator: Command Prompt
C:\Users\ASUS>php -v
PHP 7.4.10 (cli) (built: Sep 1 2020 16:52:39) ( ZTS Visual C++ 2017 x64 )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies

C:\Users\ASUS>
```

Dari gambar di atas terlihat bahwa versi php yang digunakan adalah versi 7.4.10.

2. Bukalah folder htdocs dari hasil instalasi XAMPP menggunakan perintah dalam command prompt. Contoh:

Administrator: Command Prompt

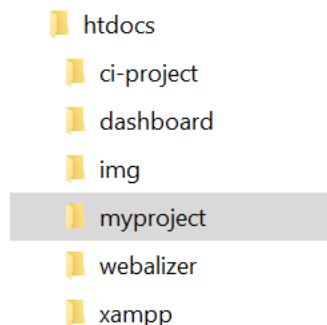
```
C:\Users\ASUS>cd D:\xampp\htdocs
C:\Users\ASUS>d:
D:\xampp\htdocs>
```

3. Kita akan membuat projek baru bernama myproject. Lakukan instalasi dengan menggunakan perintah seperti di bawah:

Administrator: Command Prompt

```
D:\xampp\htdocs>composer create-project codeigniter4/appstarter myproject --no-dev
Creating a "codeigniter4/appstarter" project at "./myproject"
Installing codeigniter4/appstarter (v4.1.1)
  - Installing codeigniter4/appstarter (v4.1.1): Extracting archive
Created project in D:\xampp\htdocs\myproject
Loading composer repositories with package information
Updating dependencies
Lock file operations: 41 installs, 0 updates, 0 removals
  - Locking codeigniter4/framework (v4.1.1)
  - Locking doctrine/instantiator (1.4.0)
  - Locking fakerphp/faker (v1.13.0)
  - Locking kint-php/kint (3.3)
  - Locking laminas/laminas-escaper (2.7.0)
```

Setelah selesai maka kita bisa melihat projek tersebut di bawah direktori htdocs pada Explorer.



Untuk memastikan apakah codeIgniter telah terinstall dengan baik, kita akan buka halaman web tersebut. Buka kembali command prompt kemudian masuk ke dalam folder **myproject**.

Administrator: Command Prompt

```
D:\xampp\htdocs>cd myproject
```

CodeIgniter 4 hadir dengan server pengembangan lokal, memanfaatkan server web bawaan PHP dengan perutean CodeIgniter. Kita dapat menggunakan skrip **serve** untuk menjalankan server ini, dengan baris perintah berikut di direktori utama:

```
c:\ Administrator: Command Prompt - php spark serve
```

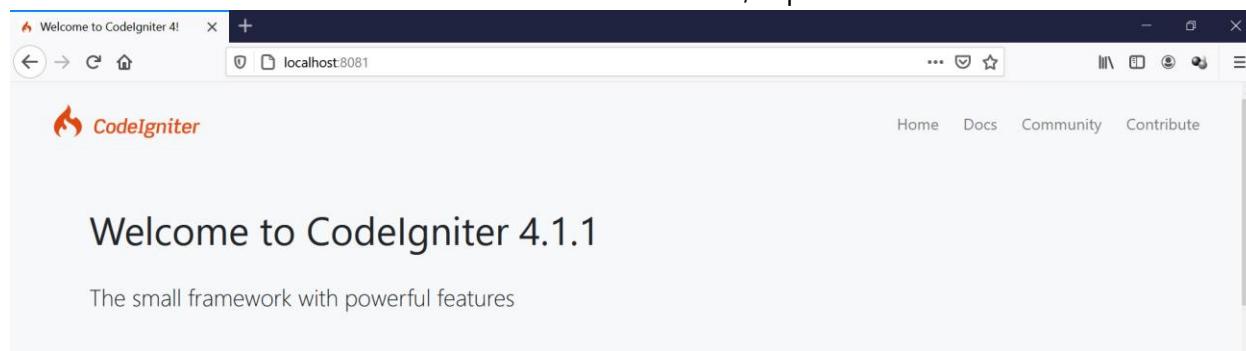
```
D:\xampp\htdocs\myproject>php spark serve
```

Port default adalah 8080. Jika sudah terpakai oleh server lain, sesuaikan port dengan konfigurasi server masing-masing. Kita bisa menggunakan perintah :

```
c:\ Administrator: Command Prompt - php spark serve --port 8081
```

```
D:\xampp\htdocs\myproject>php spark serve --port 8081
```

Setelah itu buka browser dan masukkan alamat localhost, seperti di bawah:



Gambar 3.4 Tampilan Awal CodeIgniter

### 3.6. Struktur Folder

Struktur direktori Web yang menggunakan CodeIgniter 4 terdiri atas 5 direktori yaitu app, public, tests, vendor, dan writable.

1. Direktori app: merupakan direktori yang pada dasarnya menyimpan aplikasi yang sedang kita buat.
2. Direktori public: merupakan direktori yang berisi file yang bisa diakses oleh publik, seperti file index.php, robots.txt, favicon.ico, ads.txt, dll;
3. Direktori tests: merupakan direktori yang berisi kode untuk melakukan pengujian menggunakan framework PHPUnit.
4. Direktori vendor: merupakan direktori yang berisi library atau pustaka yang dibutuhkan aplikasi termasuk kode inti dari sistem CodeIgniter.
5. Direktori writeable: merupakan direktori yang berisi file yang ditulis oleh aplikasi. Nantinya, kita bisa pakai untuk menyimpan file yang di-upload, logs, session, dll.

Selain direktori, struktur CI 4 memiliki beberapa file yaitu:

- env adalah file yang berisi variabel environment yang dibutuhkan oleh aplikasi.
- .gitignore adalah file yang berisi daftar nama file dan folder yang akan diabaikan oleh Git.
- builds adalah script untuk mengubah versi codeigniter yang digunakan. Ada versi release (stabil) dan development (labil).
- composer.json adalah file JSON yang berisi informasi tentang proyek dan daftar library yang dibutuhkannya. File ini digunakan oleh Composer sebagai acuan.
- composer.lock adalah file yang berisi informasi versi dari libraray yang digunakan aplikasi.
- LICENSE.txt adalah file yang berisi penjelasan tentang lisensi Codeigniter;
- phpunit.xml.dist adalah file XML yang berisi konfigurasi untuk PHPunit.
- README.md adalah file keterangan tentang codebase CI. Ini biasanya akan dibutuhkan pada repo github atau gitlab.
- spark adalah program atau script yang berfungsi untuk menjalankan server, generate kode, dll.

Direktori app memiliki beberapa direktori dan file di dalamnya. Direktori dan file ini banyak digunakan ketika membangun aplikasi web.

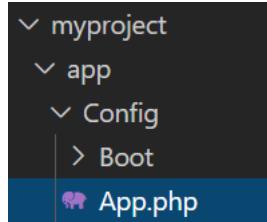
- Config, merupakan direktori yang menyimpan informasi mengenai konfigurasi aplikasi seperti autoload, database, routes dan lainnya.
- Controllers, , merupakan direktori menyimpan controller - controller aplikasi yang dapat digunakan untuk menyusun aktivitas program
- Database adalah direktori yang berisi kode untuk migrasi database dan membuat sample data.
- Filters adalah direktori yang berisi kode untuk filter atau middleware;
- Helpers, merupakan direktori yang berisi kode untuk fungsi-fungsi helper;
- Language adalah folder yang berisi kamus bahasa untuk aplikasi;
- Libraries, merupakan folder untuk menyimpan library
- Models, merupakan direktori untuk menyimpan models yang akan mendefinisikan tabel dari database yang dapat kita gunakan oleh Controller yang kita buat untuk mengakses database.
- ThirdParty, merupakan folder untuk menyimpan fungsi fungsi tambahan dalam cara kerja codeigniter.
- Views, merupakan folder untuk menyimpan tampilan dari aplikasi yang kita buat.
- .htaccess adalah file yang berisi konfigurasi akses untuk web server;
- Common.php adalah file yang berisi definisi fungsi untuk menindih fungsi core dari CI.
- index.html adalah file *placeholder* untuk mencegah *direct access*.

## 3.7. Konfigurasi

### A. Mengatur URL Utama

Maksud dari mengatur URL utama adalah konfigurasi Base URL dimana akan digunakan untuk mendeklarasikan URL aplikasi yang akan kita buat, berikut ini caranya:

1. Buka file /app/Config/App.php



2. Lalu ubah bagian berikut dengan alamat URL yang Anda inginkan.

Contoh:

```
public $baseURL = 'http://localhost:8081/';
```

### B. Konfigurasi TimeZone

Masih dalam file App.php kemudian edit \$appTimezone menjadi seperti berikut:

```
public $appTimezone = 'Asia/Jakarta';
```

### C. Aktifkan Token CSRF

Cross-site Request Forgery (CSRF) adalah sebuah serangan yang menggunakan injeksi script. Kode ini bisa berupa kode javascript, link, atau gambar dengan memanfaatkan token autentikasi. Fitur proteksi form dengan csrf protection sudah disediakan dalam CodeIgniter. Untuk mengaktifkannya, buka file Filters.php pada folder app/config kemudian aktifkan csrf seperti berikut:

```
public $globals = [
    'before' => [
        // 'honeypot',
        'csrf',
    ],
    'after'  => [
        'toolbar',
        // 'honeypot',
    ],
];
```

### D. Ubah env Production Jadi Development

Pada saat pengembangan projek sebelum disimpan dalam produksi, kita aktifkan mode debugging agar ketika terjadi kesalahan maka akan muncul pesan kesalahan tersebut

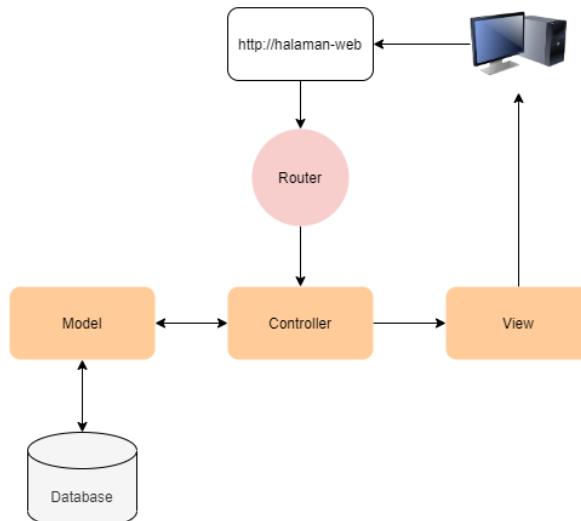
untuk memudahkan kita dalam perbaikan. Caranya adalah dengan mengubah file **env** menjadi **.env** (ditambah titik didepan tulisan env). Kemudian buka .env dan ubah kode di bawah ini:

```
# CI_ENVIRONMENT = production
menjadi
CI_ENVIRONMENT = development
```

### 3.8. Memahami Routing

Dalam sebuah framework, kita harus memahami tentang konsep "routing" atau proses menentukan rute. Routing ini adalah fitur pada Codeigniter bertugas untuk menentukan controller dan method/fungsi yang akan dieksekusi. Kode untuk membuat routing terdapat di file app/Config/Routes.php. File index.php adalah file entri point yang akan dieksekusi pertamakali saat aplikasi dibuka.

Pada saat kita membuka browser, kemudian memasukkan halaman web yang dituju artinya kita melakukan request. Request yang diterima akan diserahkan ke Router. Lalu Router akan menentukan Controller yang akan meresponnya.



Gambar 3.5 Routing pada CodeIgniter

Untuk lebih memahami konsep routing, perhatikan URL dibawah ini :

[www.myproject.com/profil](http://www.myproject.com/profil)

URL di atas merupakan contoh, dimana URL utamanya adalah [www.myproject.com](http://www.myproject.com). Setelah nama web, kita melihat ada tulisan profil, artinya ketika kita mengakses URL maka web akan memberikan response tampilan tertentu berdasarkan dari URL yang dimasukkan oleh pengguna dalam browser. Router akan mengatur jika ada request URL dari pengguna.

### 3.8.1. Segmen URI

Secara singkat bisa dikatakan URI (*Uniform Resource Identifier*) segment adalah bagian dari sebuah URI/URL yang di pisahkan oleh (/) setelah BASE URL. URI yang membantu kita dalam mengambil data melalui url codeigniter. Cara penyebutan uri segment pada codeigniter sendiri misalnya segment 1, segment 2, segment 3 dan seterusnya.

Perhatikan URL dibawah ini :

[www.myproject.com/siswa/profil/7](http://www.myproject.com/siswa/profil/7)

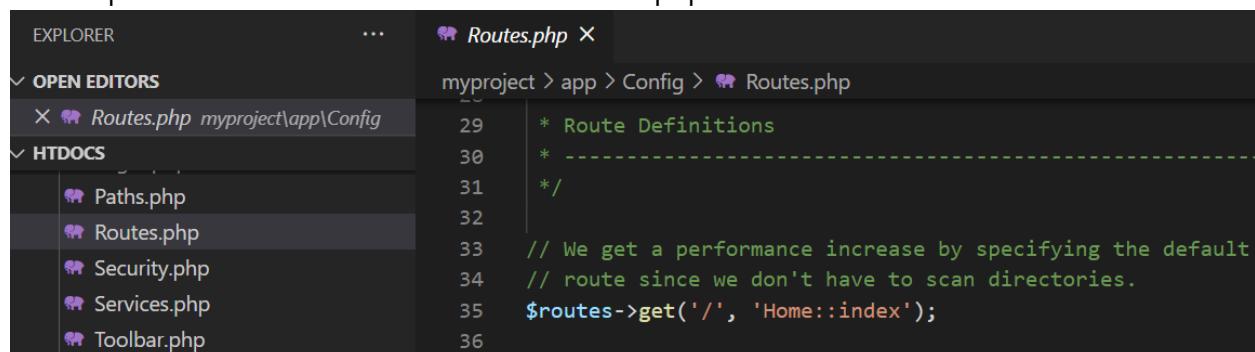
Kita akan coba membagi URL di atas menjadi beberapa bagian sesuai dengan URI Segment, menjadi seperti berikut ini :

- [www.myproject.com](http://www.myproject.com) adalah URL utama dari web
- siswa adalah Segment 1 yang merupakan nama controller
- profil adalah Segment 2 yang merupakan nama methodnya
- 7 adalah Segment 3 yang merupakan nilai parameter pertama dari method pada segment 2

Jadi jika diartikan, URL diatas akan mengakses method profil pada controller bernama siswa, serta mengirimkan nilai angka 12 sebagai nilai dari parameter di method profil.

### 3.8.2. Contoh Routing

Sekarang kita ambil contoh sederhana halaman default ketika instalasi CodeIgniter yang menampilkan halaman welcome. Buka file Routes.php.



```

EXPLORER           ...
OPEN EDITORS       ...
  Routes.php myproject\app\Config
HTDOCS
  Paths.php
  Routes.php
  Security.php
  Services.php
  Toolbar.php

Routes.php X
myproject > app > Config > Routes.php
29  * Route Definitions
30  *
31  */
32
33 // We get a performance increase by specifying the default
34 // route since we don't have to scan directories.
35 $routes->get('/', 'Home::index');
36

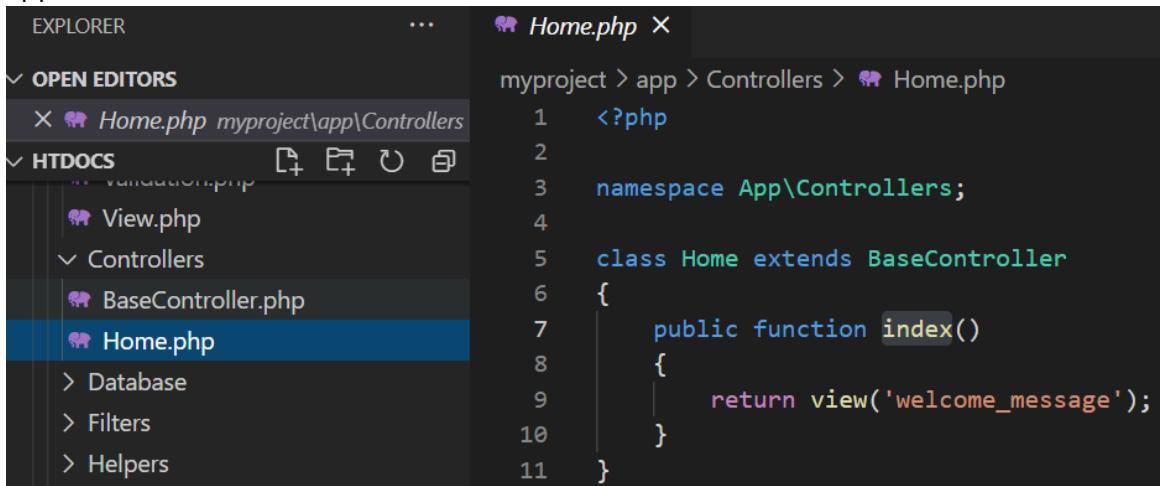
```

Pada file Routes.php, kita bisa mendefinisikan rute untuk aplikasi. Coba perhatikan bagian ini:

```
$routes->get('/', 'Home::index');
```

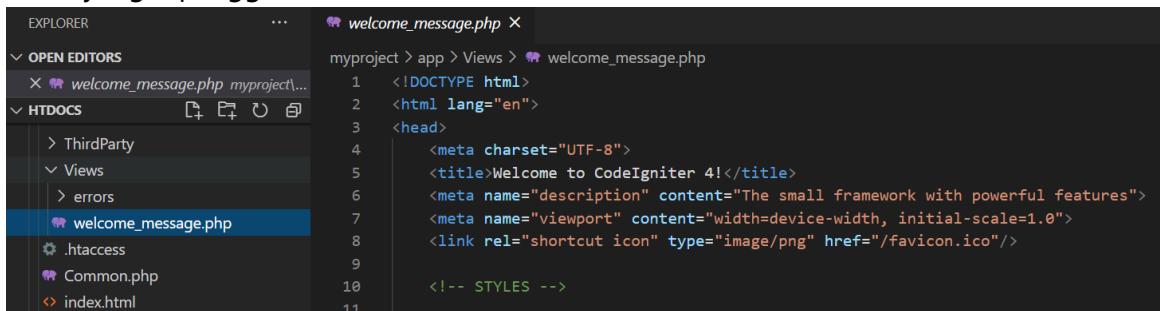
Perintah routing di atas memiliki arti :

1. Get adalah tipe request untuk mengakses halaman web
2. Tanda '/' adalah ketika pengguna web mengakses root url (misal http://localhost:8080/ ) dalam URL Browser
3. Ketika pengguna mengakses root halaman web, maka proses akan dialihkan ke controller bernama Home dan fungsi bernama index. Controller Home berada di folder app/Controller



```
myproject > app > Controllers > Home.php
1  <?php
2
3  namespace App\Controllers;
4
5  class Home extends BaseController
6  {
7      public function index()
8      {
9          return view('welcome_message');
10     }
11 }
```

4. Perhatikan fungsi index() dalam class Home. Fungsi ini mengembalikan view bernama welcome\_message.
5. Buka folder app/Views, maka kita akan melihat file bernama welcome\_message.php. File inilah yang dipanggil oleh controller.



```
myproject > app > Views > welcome_message.php
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Welcome to CodeIgniter 4!</title>
6      <meta name="description" content="The small framework with powerful features">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <link rel="shortcut icon" type="image/png" href="/favicon.ico"/>
9
10     <!-- STYLES -->
11 
```

6. View welcome\_message bertugas untuk menampilkan tampilan dari sebuah halaman web ke browser di komputer pengguna.

### 3.8.3. Membuat Routing Baru

Berikutnya kita akan membuat routing dengan controller yang berbeda. Pertama, kita buat terlebih dahulu controller bernama Siswa menggunakan composer menggunakan perintah di bawah ini:

Administrator: Command Prompt

```
D:\xampp\htdocs\myproject>php spark make:controller Siswa
```

Sebuah controller Siswa akan muncul dalam folder app/Controllers. Tambahkan fungsi profil ke dalam controller.

```
<?php

namespace App\Controllers;

use App\Controllers\BaseController;

class Siswa extends BaseController
{
    public function index()
    {
        //
    }

    public function profil() {
        return view('profil');
    }
}
```

Kemudian buat file view baru dalam folder app/Views bernama profil.php. Isikan kodennya seperti di bawah:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Profil Siswa</title>
</head>
<body>
    Ini adalah halaman profil siswa
</body>
</html>
```

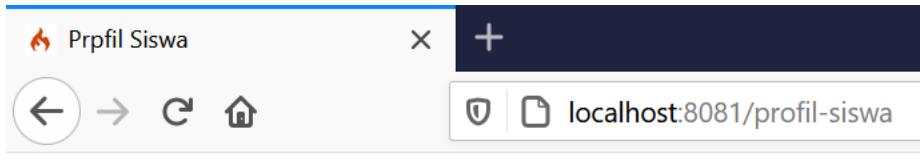
Berikutnya kita tambahkan kode untuk routing baru ke dalam Routes.php

```
$routes->get('/profil-siswa', 'Siswa::profil');
```

Sekarang, buka browser dan ketikkan alamat

<http://localhost:8081/profil-siswa>

Hasilnya terlihat seperti di bawah ini:



Gambar 3.6 Tampilan halaman profil siswa

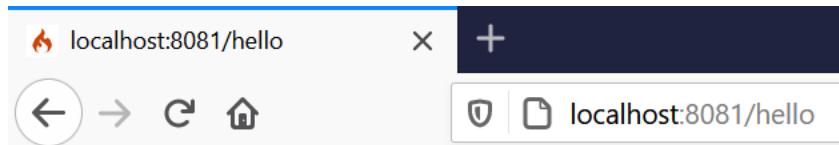
### 3.8.4. Membuat Closure

Kita juga bisa menambahkan fungsi ke dalam router. Penambahan fungsi ini sering disebut closure. Jadi kita bisa membuat fungsi yang berisi perintah php, dimana akan dijalankan ketika routing diakses, sebagai contoh kita akan menambahkan routing dengan nama hello, silahkan buka file routing didalam folder : app/Config/Routes.php. Berikutnya tambahkan code seperti berikut ini dibagian routing.

```
$routes->get('/hello', function () {
    echo "Hello World";
});
```

Buka kembali browser dan masukkan alamat menggunakan <http://localhost:8081/hello>

Hasil:



Hello World

Gambar 3.6 Tampilan halaman Hello World

### 3.8.5. Membuat Placeholder

Placeholder berfungsi untuk menambahkan parameter pada route. Parameter ini akan digunakan sebagai masukan ke dalam parameter method di controller. Selain dapat menambahkan parameter, kita juga dapat menentukan jenis data yang dikirimkan melalui parameter tersebut, apakah berbentuk angka, atau huruf, atau dapat berupa bentuk data lainnya, sehingga kita dapat mengatur value parameter yang dikirimkan nantinya. Ada beberapa jenis placeholder di Codeigniter 4 yang sering digunakan.

1. **(:any)** menerima semua jenis karakter.
2. **(:num)** hanya menerima angka.
3. **(:alpha)** hanya karakter alphabet.
4. **(:alnum)** kombinasi alphabet dan angka.

Sebagai contoh kita akan membuat method baru di dalam controller Siswa, method baru kita beri nama dataSiswa, sehingga kurang lebih controller Siswa menjadi seperti berikut ini :

```
<?php

namespace App\Controllers;

<?php

namespace App\Controllers;

use App\Controllers\BaseController;

class Siswa extends BaseController
{
    public function index()
    {
        //
    }

    public function profil() {
        return view('profil');
    }

    public function dataSiswa($nama, $umur)
    {
        echo "Nama Siswa : $nama <br/>";
        echo "Umur : $umur";
    }
}
```

Method dataSiswa memiliki 2 parameter, yaitu nama dan umur, parameter ini akan mengambil nilai dari route. Berikutnya kita buat routing baru dengan nama data-siswa, yang akan mengakses method dataSiswa didalam controller Siswa. Buka file app/Config/Routes.php lalu tambahkan perintah routing dibawah ini :

```
$routes->get('data-siswa/(:alpha)/(:num)', 'Siswa::dataSiswa/$1/$2');
```

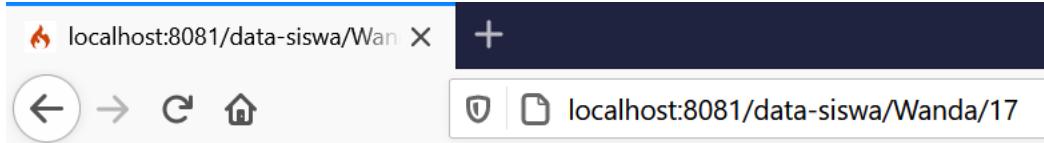
Dalam routing tersebut, kita berikan URL data-siswa, dimana akan mengakses method dataSiswa di dalam Controller Siswa. Dalam routes juga, kita berikan 2 parameter, dimana parameter yang dikirimkan melalui URL ini akan dijadikan nilai parameter pada method dataSiswa pada Controller Siswa.

Pada bagian routes kita menyebutkan jenis parameter (placeholder) pertama adalah (:alpha) yang berarti nilainya harus berupa huruf, dan di bagian parameter kedua (placeholder) adalah (:num) yang berarti nilainya harus berupa angka.

Jalankan server dengan perintah php spark serve lalu akses URL :

localhost:8080/data-siswa/Wanda/17

Hasilnya adalah sebagai berikut :



Nama Siswa : Wanda

Umur : 17

Gambar 3.7 Tampilan halaman nama dan umur

## LATIHAN

1. Buatlah sebuah halaman web yang menampilkan biodata:
  - a. Nama
  - b. Tempat dan Tanggal Lahir
  - c. Alamat
  - d. No Telepon
  - e. Jenis Kelamin
  - f. Pendidikan

Sesuaikan dengan data diri masing-masing

# BAB 4

# FORM DAN VALIDASI

## **Tujuan instruksional:**

- Memahami teknik membuat form
- Memahami pentingnya validasi
- Membuat form beserta validasi di dalamnya

Sebelum menjelaskan pendekatan CodeIgniter untuk validasi data, mari kita gambarkan skenario yang ideal:

1. Form ditampilkan.
2. Pengguna mengisinya dan mengirimkannya.
3. Jika pengguna mengirimkan sesuatu yang tidak valid, atau mungkin melewatkkan item yang diperlukan, form ditampilkan ulang yang berisi data sebelumnya bersama dengan pesan kesalahan yang menjelaskan masalah tersebut.
4. Proses ini berlanjut sampai pengguna mengirimkan form yang valid.

Di sisi penerima, skrip harus:

1. Periksa data yang diperlukan.
2. Verifikasi bahwa data adalah data yang sesuai tipenya dan memenuhi kriteria. Misalnya, jika nama mata ajar dikirimkan, maka harus divalidasi terlebih dahulu agar hanya berisi karakter yang diizinkan. Nama mata ajar tidak boleh berupa nilai yang sudah ada sebelumnya dan seterusnya.
3. Sanitasi data untuk keamanan.
4. Pra-format data jika diperlukan, seperti adanya pemangkasan data atau dalam bentuk HTML dan seterusnya)
5. Siapkan data untuk dimasukkan ke dalam database.

Terlihat tidak ada yang terlalu rumit dari proses di atas hanya biasanya memerlukan sejumlah kode untuk mengimplementasikannya. Untuk menampilkan pesan kesalahan, biasanya dibuat dalam struktur kontrol kode HTML.

## 4.1. Validasi Form

Untuk menerapkan validasi pada form, kita setidaknya memerlukan tiga hal:

1. Halaman yang berisi form.
2. Halaman yang berisi pesan "berhasil" untuk ditampilkan setelah pengiriman data berhasil.
3. Metode pengontrol untuk menerima dan memproses data yang dikirimkan.

Sekarang kita buat contoh dengan menggunakan form pendaftaran anggota.

### Form Input Data

Gunakan editor teks, buatlah View baru yang berisi sebuah form bernama form\_modul.php dan letakkan file tersebut di folder **app/Views/** dan ketikkan kode di bawah lalu simpan

```
<!DOCTYPE html>
<html lang="en">

<head>
```

```

<title>Form Modul</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<!-- CSS only -->
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<!-- JS, Popper.js, and jQuery -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</head>

<body>
    <div class="container">
        <div class="card">
            <div class="card-header bg-primary text-white">Tambah Data Modul</div>
            <div class="card-body">
                <?php if (!empty(session()->getFlashdata('error'))): ?>
                    <div class="alert alert-danger alert-dismissible fade show" role="alert">
                        <h4>Periksa Entrian Form</h4>
                        </hr />
                        <?php echo session()->getFlashdata('error'); ?>
                        <button type="button" class="close" data-dismiss="alert" aria-label="Close">
                            <span aria-hidden="true">&times;</span>
                        </button>
                    </div>
                <?php endif; ?>
                <form action="= base_url('modul/save'); ?" method="post">
                    <?= csrf_field(); ?>
                    <div class="form-group">
                        <label for="kode">Kode</label>
                        <input type="text" name="kode" class="form-control" value="= old('kode'); ?" id="kode">
                    </div>
                    <div class="form-group">
                        <label for="modul">Nama Modul</label>
                        <input type="text" name="modul" class="form-control" id="modul" value="= old('modul'); ?">
                    </div>
            </div>
        </div>
    </div>
</body>

```

```

<div class="form-group">
    <label for="SKS">SKS</label>
    <input type="number" name="skks" class="form-control" id="skks">
</div>
<button type="submit" class="btn btn-primary">Save</button>
</form>
</div>
</div>
</body>

</html>

```

Bagian tampilan form ini kita menggunakan CDN (Content Devlivery Network) seperti framework css bootstrap, jquery dan javascriptnya. Oleh karena menggunakan CDN, maka untuk menjalankan halaman ini harus selalu terhubung ke internet. Sekarang perhatikan pada tag <form>. Kita menambahkan kode:

```
action="= base_url('modul/save'); ?&gt;"</pre

```

Kode ini memiliki arti ketika tombol save di klik oleh pengguna maka entri form akan di proses ke dalam method save pada controller Modul. Di dalam tag <form> kita menuliskan perintah :

```
<?= csrf_field(); ?>
```

Perintah diatas digunakan untuk memberikan token CSRF (Cross-Site Request Forgery) di bagian form untuk mengamankan data yang dikirim melalui form ini.

Berikutnya kita buat beberapa kontrol input pada bagian form, seperti halnya kita membuat form di HTML, kita perlu memberikan value pada attribute name di masing – masing kontrol.

Untuk mengakses tampilan form, kita bisa mengaktifkan local development server terlebih dahulu dengan perintah "php spark serve" lalu bisa mengakses url seperti pada gambar di bawah:



Gambar 4.1 Tampilan form tambah data modul

## Controller

Berikutnya kita akan membuat validasi form di codeigniter 4, validasi form ini sangat penting untuk mengatur data yang dimasukkan melalui form oleh pengguna aplikasi, sehingga ketika data dikirimkan dan akan dimasukkan ke dalam database sesuai dengan struktur tabel yang ada. Topik mengenai database ada di bab berikutnya. Kali ini kita hanya akan membahas bagaimana membuat validasi form terlebih dahulu. Gunakan editor teks, buatlah Controller baru bernama Modul menggunakan composer.

 Administrator: Command Prompt - php spark serve  
D:\xampp\htdocs\myproject>php spark make:controller Modul

Untuk membuat validasi form, kita ketikkan kode di dalam method save pada controller Modul, sehingga method save menjadi seperti berikut ini :

```
<?php

namespace App\Controllers;

use App\Controllers\BaseController;

class Modul extends BaseController
{
    public function index()
    {
        return view('form_modul');
    }
    public function save()
    {
        if (!$this->validate([
            'kode' => [
                'rules' => 'required',
                'errors' => [
                    'required' => '{field} Harus diisi'
                ]
            ],
            'modul' => [
                'rules' => 'required',
                'errors' => [
                    'required' => '{field} Harus diisi'
                ]
            ],
            'sks' => [
                'rules' => 'required|greater_than[0]|less_than[10]',
                'errors' => [
                    'required' => '{field} Harus diisi',
                    'greater_than' => '{field} Wajib lebih besar dari 0',
                    'less_than' => '{field} Wajib kurang dari 10'
                ]
            ]
        ]))
        return redirect()->to('/modul');
    }
}
```

```
        'required' => '{field} Harus diisi'
    ]
]
]) {
    session()->setflashdata('error', $this->validator->listErrors());
    return redirect()->back()->withInput();
} else {
    print_r($this->request->getVar());
}
}
}
```

Perintah if digunakan untuk melakukan cek dari penggunaan method \$this->validate. Method validate ini berfungsi untuk melakukan validasi data. Pada method validate kita isi parameternya dengan array multidimensi, dimana bisa kita isi dengan rules (aturan) dan message error (pesan error). Misal jika kita lihat kode di atas :

```
'kode' => [
    'rules' => 'required',
    'errors' => [
        'required' => '{field} Harus diisi'
    ]
],
'modul' => [
    'rules' => 'required',
    'errors' => [
        'required' => '{field} Harus diisi'
    ]
],
'sks' => [
    'rules' => 'required|greater_than[0]|less_than[10]',
    'errors' => [
        'required' => '{field} Harus diisi'
    ]
]
```

Dari kode di atas kita melihat ada rules 'required' (harus diisi) pada kode, modul, dan sks. Artinya ketiga input tersebut harus diisi oleh pengguna, jika tidak diisi akan menampilkan pesan error "{field} Harus diisi". Nilai {field} ini akan bernilai sesuai dengan data inputannya. Untuk sks, kita menambahkan validasi dimana nilai sks harus lebih besar dari 0 dan kurang dari 10.

Jika terdapat rules yang bernilai false (inputan form tidak sesuai aturan) maka akan menjalankan perintah:

```
session()->setflashdata('error', $this->validator->listErrors());
```

Kita kirimkan error dengan menggunakan session flashdata, flashdata adalah istilah pengiriman data lewat session, tapi hanya satu kali pakai, biasanya digunakan untuk mengirimkan pesan / alert. Dalam contoh ini kita gunakan untuk mengirimkan pesan error dari form validation, setflashdata memiliki 2 parameter yaitu :

```
nama flashdata : error  
value flashdata : , $this->validator->listErrors()
```

Pada contoh diatas kita membuat flashdata dengan nama error yang berisi list error dari form validation. Kemudian perintah :

```
return redirect()->back()->withInput();
```

berfungsi untuk redirect ke halaman sebelumnya (form input), yang artinya adalah halaman form\_modul, dengan menyertakan inputan yang sudah dientrikan sebelumnya (withInput). Berikutnya kita menggunakan perintah print\_r yang digunakan untuk menampilkan hasil inputan form.

```
print_r($this->request->getVar());
```

Kode di atas akan dijalankan jika input form sudah sesuai rules. Kode ini berlaku untuk inputan dengan tipe method post ataupun get. Jika kita ingin mengakses inputan form bagian tertentu, bisa menuliskan value name dibagian input di method getVar, semisal dari form diatas kita ingin mengakses inputan modul, maka bisa menggunakan perintah :

```
$this->request->getVar(modul)
```

## 4.2. Pengujian Form

Setelah kita konfigurasi bagian validasi form pada method save di controller Modul, berikutnya kita akan coba melakukan pengujian. Sekarang kita buka halaman awal untuk form modul :

localhost:8080/modul

Maka tampilannya adalah seperti berikut ini :

A screenshot of a web browser window titled 'Form Modul'. The address bar shows 'localhost:8081/modul'. The main content area is titled 'Tambah Data Modul'. It contains three input fields: 'Kode', 'Nama Modul', and 'SKS', each with a corresponding text input box. Below the input fields is a blue 'Save' button.

Gambar 4.2 Tampilan awal pengujian form

Keterangan :

Jika kita tidak mengisi data, lalu menekan tombol Save, maka hasilnya adalah sebagai berikut:

A screenshot of a web browser window titled 'Form Modul'. The address bar shows 'localhost:8081/modul'. A pink rectangular box at the top contains the text 'Periksa Entri Form' and a bulleted list of validation errors: '• kode Harus diisi', '• modul Harus diisi', and '• sks Harus diisi'. Below this box are the same three input fields ('Kode', 'Nama Modul', 'SKS') and a 'Save' button as shown in Figure 4.2.

Gambar 4.3 Tampilan hasil validasi

Jika kita tidak mengisi data kode dan modul, dan memasukkan nilai 0 pada SKS lalu kita tekan tombol Save, maka hasilnya adalah sebagai berikut :

Gambar 4.4 Tampilan hasil validasi bagian 2

Jika kita isi data dengan benar lalu kita tekan tombol Save, maka hasilnya adalah sebagai berikut:

Gambar 4.5 Tampilan data yang tersimpan

Dari gambar, terlihat kita sudah berhasil menangkap data yang dikirimkan melalui form.

Beberapa rules lain yang bisa kita tambahkan ke dalam aplikasi seperti tabel di bawah ini:

Tabel 4.1 Rules

Rule	Parameter	Description	Example
alpha	Tidak ada	Menghasilkan nilai salah jika terdapat karakter selain huruf	
alpha_space	Tidak ada	Menghasilkan nilai salah jika terdapat karakter selain huruf atau spasi	
alpha_dash	Tidak ada	Menghasilkan nilai salah jika terdapat karakter selain huruf, angka, garis bawah, atau tanda pisah	

alpha_numeric	Tidak ada	Menghasilkan nilai salah jika terdapat karakter selain huruf dan angka	
alpha_numeric_space	Tidak ada	Menghasilkan nilai salah jika terdapat karakter selain huruf dan angka atau spasi	
alpha_numeric_punct	Tidak ada	Menghasilkan nilai salah jika terdapat karakter selain huruf, angka , spasi, atau karakter tanda baca seperti : ~ (tilde), ! (exclamation), # (number), \$ (dollar), % (percent), & (ampersand), * (asterisk), - (dash), _ (underscore), + (plus), = (equals),   (vertical bar), : (colon), . (period).	
decimal	Tidak ada	Menghasilkan nilai salah jika terdapat karakter selain desimal. Validasi ini juga menerima masukan tanda angka + atau -	
differs	Ada	Menghasilkan nilai salah jika nilai masukan tidak berbeda dengan nilai pada parameter	differs[field_name]
exact_length	Ada	Menghasilkan nilai salah jika masukan tidak sama dengan nilai dalam parameter	exact_length[5] atau exact_length[5,8,12]
greater_than	Ada	Menghasilkan nilai salah jika masukan kurang atau sama dengan nilai dalam parameter atau tidak bernilai numerik	greater_than[8]
greater_than_equal_to	Ada	Menghasilkan nilai salah jika masukan kurang	greater_than_equal_to[5]

		dari nilai paramater atau tidak bernilai numerik	
hex	Tidak ada	Menghasilkan nilai salah jika masukan berisi apa pun selain karakter heksadesimal	
if_exist	Tidak ada	Jika aturan ini ada, maka validasi akan menghasilkan false jika tidak ada yang dihasilkan tanpa melihat nilainya	
in_list	Ada	Fails if field is not within a predetermined list.	in_list[red,blue,green]
integer	Tidak ada	Fails if field contains anything other than an integer.	
is_natural	Tidak ada	Fails if field contains anything other than a natural number: 0, 1, 2, 3, etc.	
is_natural_no_zero	Tidak ada	Fails if field contains anything other than a natural number, except zero: 1, 2, 3, etc.	
is_not_unique	Ada	Checks the database to see if the given value exist. Can ignore records by field/value to filter (currently accept only one filter).	is_not_unique[table.field,where_field,where_value]
is_unique	Ada	Checks if this field value exists in the database. Optionally set a column and value to ignore, useful when updating records to ignore itself.	is_unique[table.field,ignore_field,ignore_value]
less_than	Ada	Fails if field is greater than or equal to the parameter value or not numeric.	less_than[8]

less_than_equal_to	Ada	Fails if field is greater than the parameter value or not numeric.	less_than_equal_to[8]
matches	Ada	The value must match the value of the field in the parameter.	matches[field]
max_length	Ada	Fails if field is longer than the parameter value.	max_length[8]
min_length	Ada	Fails if field is shorter than the parameter value.	min_length[3]
not_in_list	Ada	Fails if field is within a predetermined list.	not_in_list[red,blue,green]
numeric	Tidak ada	Fails if field contains anything other than numeric characters.	
regex_match	Ada	Fails if field does not match the regular expression.	regex_match[/regex/]
permit_empty	Tidak ada	Allows the field to receive an empty array, empty string, null or false.	
required	Tidak ada	Fails if the field is an empty array, empty string, null or false.	
required_with	Ada	The field is required when any of the other required fields are present in the data.	required_with[field1,field2]
required_without	Ada	The field is required when all of the other fields are present in the data but not required.	required_without[field1,field2]
string	Tidak ada	A generic alternative to the alpha* rules that confirms the element is a string	
timezone	Tidak ada	Fails if field does match a timezone per	

		<code>timezone_identifiers_list</code>	
		<code>t</code>	
valid_base64	Tidak ada	Fails if field contains anything other than valid Base64 characters.	
valid_json	Tidak ada	Fails if field does not contain a valid JSON string.	
valid_email	Tidak ada	Fails if field does not contain a valid email address.	
valid_emails	Tidak ada	Fails if any value provided in a comma separated list is not a valid email.	
valid_ip	Tidak ada	Fails if the supplied IP is not valid. Accepts an optional parameter of 'ipv4' or 'ipv6' to specify an IP format.	valid_ip[ipv6]
valid_url	Tidak ada	Fails if field does not contain a valid URL.	
valid_date	Tidak ada	Fails if field does not contain a valid date. Accepts an optional parameter to matches a date format.	valid_date[d/m/Y]
valid_cc_number	Ada	Vерifies that the credit card number matches the format used by the specified provider. Current supported providers are: American Express (amex), China Unionpay (unionpay), Diners Club CarteBlance (carteblanche), Diners Club (dinersclub), Discover Card (discover),	

		Interpayment (interpayment), JCB (jcb), Maestro (maestro), Dankort (dankort), NSPK MIR (mir), Troy (troy), MasterCard (mastercard), Visa (visa), UATP (uatp), Verve (verve), CIBC Convenience Card (cibc), Royal Bank of Canada Client Card (rbc), TD Canada Trust Access Card (tdtrust), Scotiabank Scotia Card (scotia), BMO ABM Card (bmoabm), HSBC Canada Card (hsbc)	
--	--	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

### Latihan

Buatlah aplikasi web yang menampilkan Form Registrasi berisi:

- Nama
- Tempat dan Tanggal Lahir
- Alamat
- No Telepon
- Jenis Kelamin
- Pendidikan

Ketika Form disubmit, hasil data yang dimasukkan akan ditampilkan dan jika data tidak dimasukkan dengan benar maka form akan memberikan info kesalahan yang dibuat

# BAB 5

# MEMBUAT CRUD

# PADA

# APLIKASI WEB

### Tujuan instruksional:

- Mampu membuat halaman untuk CRUD menggunakan Code Igniter

CRUD adalah singkatan dari Create, Read, Update, dan Delete. Proses ini sangat berkaitan dengan pengambilan atau transaksi data dari atau ke database. CRUD merupakan salah satu inti dari sebuah pemrograman karena di dalam suatu program biasanya mencakup operasi untuk Create/menambah data, Read/Menampilkan data, Update/mengedit suatu data dan Delete /menghapus data. Untuk menggunakan operasi tersebut, kita harus terhubung dengan database. Hal ini menjadi krusial apabila berhubungan dengan sistem informasi perusahaan karena data yang diproses biasanya merupakan data transaksi.

## 5.1. Konfigurasi Database

CodeIgniter mendukung banyak jenis database seperti MySQL/MariaDB, PostGreSQL, Oracle dan lain-lain. Dukungan database dari CodeIgniter berupa penyediaan beberapa driver database yang sekaligus juga memiliki fungsi keamanan, caching dan active record. Agar dapat melakukan koneksi dengan database, yang harus dilakukan adalah melakukan konfigurasi database pada file app/Config/Database.php seperti berikut :

```
public $default = [
    'DSN'      => '',
    'hostname' => 'localhost',
    'username' => 'root',
    'password' => '',
    'database' => 'nama_database',
    'DBDriver' => 'MySQLI',
    'DBPrefix' => '',
    'pConnect' => false,
    'DBDebug'  => (ENVIRONMENT !== 'production'),
    'charset'   => 'utf8',
    'DBCollat'  => 'utf8_general_ci',
    'swapPre'   => '',
    'encrypt'   => false,
    'compress'  => false,
    'strictOn'  => false,
    'failover'  => [],
    'port'      => 3306,
];
```

Selain cara di atas, kita juga bisa menggunakan cara lain yaitu dengan menggunakan file env yang terdapat pada root project. Jika kita ingin menggunakan file tersebut, maka ubah file env menjadi .env. Apabila sudah diubah, silakan buka file tersebut dan kemudian temukan kode berikut:

```
# database.default.hostname = localhost
```

```
# database.default.database = ci4
# database.default.username = root
# database.default.password = root
# database.default.DBDriver = MySQLi
```

Kita bisa atur informasi yang ada di dalamnya seperti nama hostname, database, username dan password untuk koneksi ke database, serta drivernya. Jika ingin menggunakannya, hapuslah tanda tagar # di setiap baris kodanya sehingga menjadi seperti ini:

```
database.default.hostname = localhost
database.default.database = ci4
database.default.username = root
database.default.password = root
database.default.DBDriver = MySQLi
```

Kemudian masukkan data yang sesuai ke masing-masing baris.

### 5.1.1. Membuat Database

Buatlah database baru, misal kita beri nama dbproject di dalam MySQL. Misal kita beri nama dbproject. Kemudian buka file Database.php pada folder app/Config dan edit bagian database menjadi seperti berikut:

```
'database' => 'dbproject',
```

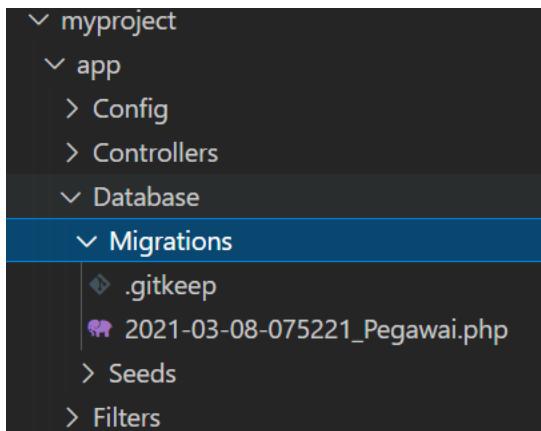
Setelah kita tentukan nama databasenya, langkah selanjutnya adalah membuat tabel untuk database tersebut. Dalam codeigniter 4 kita bisa menggunakan fitur migrasi (*migration*). Fitur ini ada di berbagai macam framework php modern, biasa digunakan untuk membuat dan memodifikasi tabel yang ada pada database. Jadi, kita tidak perlu menjalankan syntax SQL dari terminal atau editor khusus untuk membuat tabel. Cukup dengan satu perintah, maka perubahan di database dapat dilakukan tanpa harus mengganggu tabel dan data yang sudah ada.

Bukalah command prompt atau terminal. Masuk ke direktori projek kemudian buat tabel pegawai menggunakan perintah berikut:

Administrator: Command Prompt

```
D:\xampp\htdocs\myproject>php spark migrate:create Pegawai
CodeIgniter v4.1.1 Command Line Tool - Server Time: 2021-03-08 01:52:21 UTC-06:00
File created: APPPATH\Database\Migrations\2021-03-08-075221_Pegawai.php
```

Selanjutnya akan terbentuk file baru dengan nama 2021-XXXX\_Pegawai.php pada folder App/Database/Migrations



Buka file tersebut kemudian edit menjadi seperti berikut:

```

<?php

namespace App\Database\Migrations;
use CodeIgniter\Database\Migration;

class Pegawai extends Migration {
    public function up() {
        $this->forge->addField([
            'id_pegawai' => [
                'type' => 'INT',
                'unsigned' => TRUE,
                'auto_increment' => TRUE
            ],
            'nip' => [
                'type' => 'VARCHAR',
                'constraint' => 10,
                'null' => FALSE
            ],
            'nama_pegawai' => [
                'type' => 'VARCHAR',
                'constraint' => 30,
                'null' => FALSE,
            ],
            'alamat' => [
                'type' => 'TEXT',
                'null' => FALSE
            ],
            'telp' => [
                'type' => 'VARCHAR',
                'constraint' => 20,
                'null' => FALSE
            ],
        ],
    }
}

```

```

        'created_at' => [
            'type' => 'datetime',
            'null' => TRUE
        ],
        'updated_at' => [
            'type' => 'datetime',
            'null' => TRUE
        ]
    ]);

    $this->forge->addKey('id_pegawai', TRUE);
    $this->forge->createTable('pegawai');
}

public function down() {
    $this->forge->dropTable('pegawai');
}
}

```

Kemudian dari command prompt jalankan perintah berikut untuk membuat tabel pegawai:

```

Administrator: Command Prompt
D:\xampp\htdocs\myproject>php spark migrate

CodeIgniter v4.1.1 Command Line Tool - Server Time: 2021-03-08 02:07:50 UTC-06:00

Running all new migrations...
    Running: (App) 2021-03-08-075221_App\Database\Migrations\Pegawai
Done migrations.

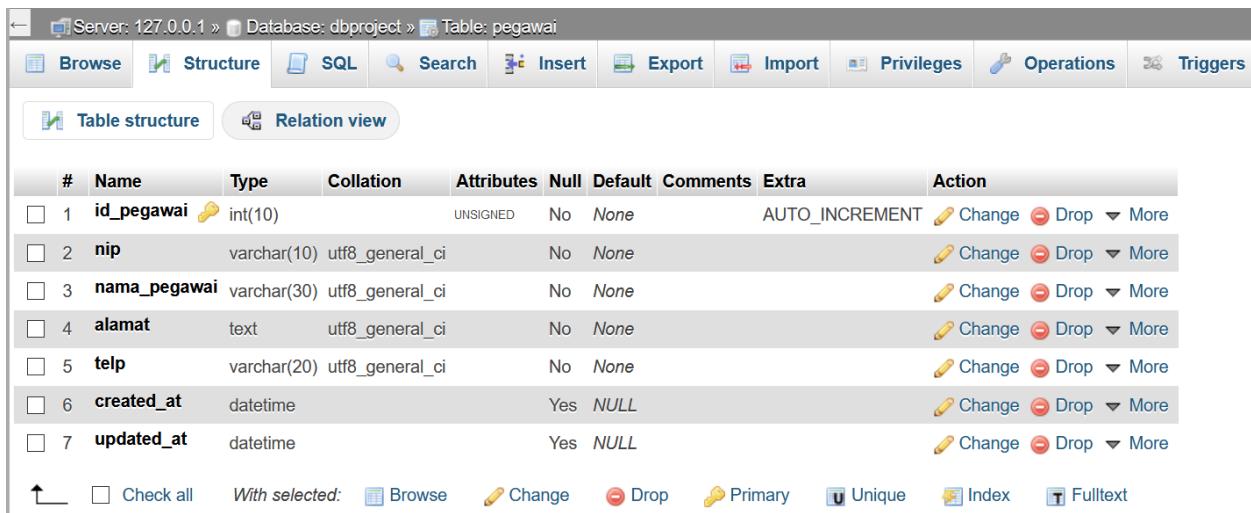
```

Kita bisa melihat hasilnya dengan membuka database di perangkat yang ada, misal kita menggunakan phpmyadmin. Hasilnya terlihat seperti berikut:

Table	Action	Rows	Type	Collation	Size	Overhead
migrations	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	1	InnoDB	utf8_general_ci	16.0 KiB	-
pegawai	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	0	InnoDB	utf8_general_ci	16.0 KiB	-
2 tables	Sum			utf8mb4_general_ci	32.0 KiB	0 B

Gambar 5.1 Tabel pegawai

Jika kita klik tabel pegawai, maka akan terlihat struktur tabel sesuai dengan yang telah didefinisikan pada kode di atas.



The screenshot shows the 'Table structure' view for the 'pegawai' table in the 'dbproject' database. The table has 7 columns:

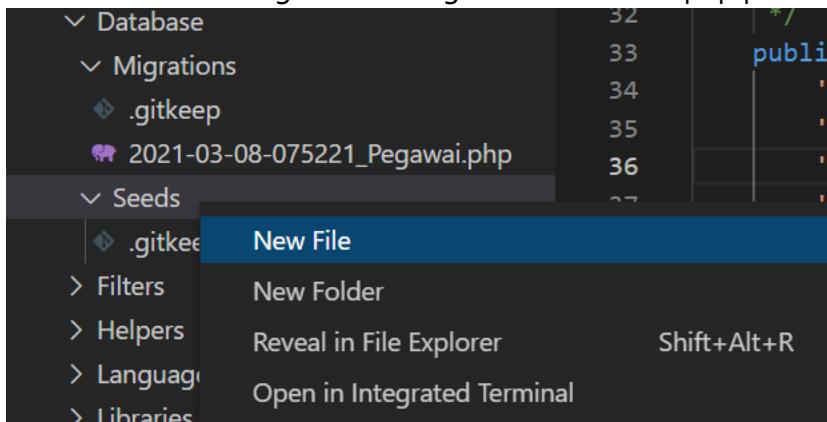
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<b>id_pegawai</b>	int(10)	utf8_general_ci	UNSIGNED	No	None		AUTO_INCREMENT	<span>Change</span> <span>Drop</span> <span>More</span>
2	<b>nip</b>	varchar(10)	utf8_general_ci		No	None			<span>Change</span> <span>Drop</span> <span>More</span>
3	<b>nama_pegawai</b>	varchar(30)	utf8_general_ci		No	None			<span>Change</span> <span>Drop</span> <span>More</span>
4	<b>alamat</b>	text	utf8_general_ci		No	None			<span>Change</span> <span>Drop</span> <span>More</span>
5	<b>telp</b>	varchar(20)	utf8_general_ci		No	None			<span>Change</span> <span>Drop</span> <span>More</span>
6	<b>created_at</b>	datetime			Yes	NULL			<span>Change</span> <span>Drop</span> <span>More</span>
7	<b>updated_at</b>	datetime			Yes	NULL			<span>Change</span> <span>Drop</span> <span>More</span>

At the bottom, there are buttons for 'Check all', 'With selected:', and various actions like 'Browse', 'Change', 'Drop', etc.

Gambar 2 Stuktur tabel pegawai

### 5.1.2. Memasukkan Data ke Dalam Database

Untuk menampilkan data, kita perlu memasukkan data terlebih dahulu ke dalam tabel. Kita bisa memasukkan dengan cara manual menggunakan phpmyadmin atau menggunakan cara yang telah disediakan oleh codeigniter4 yaitu menggunakan seeder. Kita akan pakai cara kedua, jadi buatlah file baru dengan nama PegawaiTableSeeder.php pada folder app\Database\Seeds.



Kemudian ketikkan script berikut:

```
<?php

namespace App\Database\Seeds;

class PegawaiTableSeeder extends \CodeIgniter\Database\Seeder
{
    /**
     * Run the database seeds.
     */
}
```

```

* @return void
*/
public function run()
{
    $data = [
        [
            'nip'          => '2021010101',
            'nama_pegawai' => 'Steve Grant Rogers',
            'alamat'       => 'Manhattan, NY No. 405',
            'telp'          => '08123456789',
            'created_at'   => date('Y-m-d H:i:s'),
            'updated_at'   => date('Y-m-d H:i:s')
        ],
        [
            'nip'          => '2021010102',
            'nama_pegawai' => 'Thor Odinson',
            'alamat'       => 'Asgard, Cave No. 179',
            'telp'          => '08123456788',
            'created_at'   => date('Y-m-d H:i:s'),
            'updated_at'   => date('Y-m-d H:i:s')
        ],
        [
            'nip'          => '2021010103',
            'nama_pegawai' => 'Anthony Edward Stark',
            'alamat'       => 'Long Island, NY No. 103',
            'telp'          => '08123456787',
            'created_at'   => date('Y-m-d H:i:s'),
            'updated_at'   => date('Y-m-d H:i:s')
        ],
        [
            'nip'          => '2021010104',
            'nama_pegawai' => 'Nicholas Josep Fury Jr.',
            'alamat'       => 'Atlanta, Georgia No. 1',
            'telp'          => '08123456786',
            'created_at'   => date('Y-m-d H:i:s'),
            'updated_at'   => date('Y-m-d H:i:s')
        ],
        [
            'nip'          => '2021010105',
            'nama_pegawai' => 'Robert Bruce Banner',
            'alamat'       => 'Dayton, Ohio No. 185',
            'telp'          => '08123456785',
            'created_at'   => date('Y-m-d H:i:s'),
            'updated_at'   => date('Y-m-d H:i:s')
        ],
    ]
}

```

```
[
  [
    'nip'          => '2021010106',
    'nama_pegawai' => 'Natalia Alianovna Romanova',
    'alamat'       => 'Stalingrad No.804',
    'telp'         => '08123456784',
    'created_at'   => date('Y-m-d H:i:s'),
    'updated_at'   => date('Y-m-d H:i:s')
  ],
  [
    [
      'nip'          => '2021010107',
      'nama_pegawai' => 'Clinton Francis Barton',
      'alamat'       => 'Waverly Iowa No. 902',
      'telp'         => '08123456783',
      'created_at'   => date('Y-m-d H:i:s'),
      'updated_at'   => date('Y-m-d H:i:s')
    ],
    [
      [
        'nip'          => '2021010108',
        'nama_pegawai' => 'Wanda Maximoff',
        'alamat'       => 'Westview NJ No. 2800',
        'telp'         => '08123456782',
        'created_at'   => date('Y-m-d H:i:s'),
        'updated_at'   => date('Y-m-d H:i:s')
      ],
      [
        [
          'nip'          => '2021010108',
          'nama_pegawai' => 'Vision',
          'alamat'       => 'Westview NJ No. 2800',
          'telp'         => '08123456781',
          'created_at'   => date('Y-m-d H:i:s'),
          'updated_at'   => date('Y-m-d H:i:s')
        ],
        [
          [
            'nip'          => '2021010109',
            'nama_pegawai' => 'Henry Pym',
            'alamat'       => 'LA No 181',
            'telp'         => '08123456780',
            'created_at'   => date('Y-m-d H:i:s'),
            'updated_at'   => date('Y-m-d H:i:s')
          ],
          [
            [
              'nip'          => '2021010110',
              'nama_pegawai' => 'Scott Lang',
              'alamat'       => 'Rhinebeck, NY No. 604',
              'telp'         => '08123456779',
            ]
          ]
        ]
      ]
    ]
  ]
]
```

```

        'created_at'      => date('Y-m-d H:i:s'),
        'updated_at'      => date('Y-m-d H:i:s')
    ],
    [
        'nip'              => '2021010111',
        'nama_pegawai'    => 'Gamora Zen Whoberi Ben Titan',
        'alamat'          => 'Zen-Whoberis No. 116',
        'telp'             => '08123456778',
        'created_at'      => date('Y-m-d H:i:s'),
        'updated_at'      => date('Y-m-d H:i:s')
    ]
];
$this->db->table('pegawai')->insertBatch($data);
}
}

```

Simpan, Kemudian dari command prompt jalankan perintah berikut untuk menambahkan data pegawai:

```

Administrator: Command Prompt
D:\xampp\htdocs\myproject>php spark db:seed PegawaiTableSeeder

CodeIgniter v4.1.1 Command Line Tool - Server Time: 2021-03-08 03:51:37 UTC-06:00

Seeded: App\Database\Seeds\PegawaiTableSeeder

```

Sekarang kita bisa melihat hasilnya dalam database seperti gambar di bawah ini:

The screenshot shows the phpMyAdmin interface for the 'pegawai' table. The table has columns: id\_pegawai, nip, nama\_pegawai, alamat, telp, created\_at, and updated\_at. The data is as follows:

	+ Options	id_pegawai	nip	nama_pegawai	alamat	telp	created_at	updated_at
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	1	2021010101	Steve Grant Rogers	Manhattan, NY No. 405	08123456789	2021-03-08 03:51:37	2021-03-08 03:51:37
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	2	2021010102	Thor Odinson	Asgard, Cave No. 179	08123456788	2021-03-08 03:51:37	2021-03-08 03:51:37
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	3	2021010103	Anthony Edward Stark	Long Island, NY No. 103	08123456787	2021-03-08 03:51:37	2021-03-08 03:51:37
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	4	2021010104	Nicholas Josep Fury Jr.	Atlanta, Georgia No. 1	08123456786	2021-03-08 03:51:37	2021-03-08 03:51:37
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	5	2021010105	Robert Bruce Banner	Dayton, Ohio No.	08123456785	2021-03-08 03:51:37	2021-03-08 03:51:37

Gambar 5.3 Hasil data yang telah dimasukkan

## 5.2. Membuat List Data

### 5.2.1. Membuat Model

Proses menampilkan data yang telah dibuat umumnya ditampilkan dalam bentuk sebuah tabel. Pertama, kita buat terlebih dahulu Model yang sesuai. Untuk membuat file model bisa dengan cara membuat file baru dengan nama Pegawai.php pada folder **app/Models** atau menggunakan composer dengan cara:

```
C:\Administrator: Command Prompt
D:\xampp\htdocs\myproject>php spark make:model Pegawai

CodeIgniter v4.1.1 Command Line Tool - Server Time: 2021-03-08 06:38:50 UTC-06:00

File created: APPPATH\Models\Pegawai.php
```

kemudian ketikkan script berikut:

```
<?php

namespace App\Models;
use CodeIgniter\Model;

class Pegawai extends Model
{
    protected $table = 'pegawai';
    protected $primaryKey = 'id_pegawai';
    protected $allowedFields = ['nip', 'nama_pegawai', 'alamat', 'telp'];
    protected $useTimestamps = true;
    protected $createdField = 'created_at';
    protected $updatedField = 'updated_at';
}
```

### 5.2.2. Membuat Controller

Buat file controller baru dengan nama PegawaiController menggunakan composer:

```
C:\Administrator: Command Prompt
D:\xampp\htdocs\myproject>php spark make:controller PegawaiController

CodeIgniter v4.1.1 Command Line Tool - Server Time: 2021-03-08 08:46:42 UTC-06:00

File created: APPPATH\Controllers\PegawaiController.php
```

Kita juga bisa membuat file controller baru secara manual dengan memasukkannya ke dalam folder app/Controllers. Kemudian ketikkan script ke dalam controller tersebut:

```
<?php

namespace App\Controllers;

use App\Controllers\BaseController;

class PegawaiController extends BaseController
{
    protected $model;

    public function __construct()
    {
        $this->model = new Pegawai();
        $this->helpers = ['form', 'url'];
    }

    public function index()
    {
        $data = [
            'result'      => $this->model->orderBy('nip', 'asc')->paginate(10),
            'pager'       => $this->model->pager,
            'title'       => 'Pegawai List'
        ];
        return view('pegawai/list', $data);
    }

    public function add()
    {
        //
    }

    public function store()
    {
        //
    }

    public function show()
    {
        //
    }

    public function edit($id)
    {
        //
    }
}
```

```

public function update($id)
{
    //
}

public function destroy($id)
{
    //
}
}

```

### 5.2.3. Membuat View

Pada folder **app/Views** kita akan membuat base template dengan nama `base.php`. Base template ini kita gunakan sebagai template tampilan antarmuka yang akan kita gunakan untuk menyeragamkan halaman-halaman lain ada di dalam aplikasi sehingga kita tidak perlu lagi membuat kode yang sama setiap kali membuat halaman web. Setelah membuat file `base.php`, ketikkan script berikut pada file tersebut:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <meta name="= csrf_token() ?" content="= csrf_hash() ?">
    <title><?= $title; ?></title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGs03+Hhxv8T/Q5PaXtkKtu6ug5T0eNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
</head>
<body>
    <!-- Content -->
    <?= $this->renderSection('content') ?>
    <!-- /.Content -->
    <footer class="text-center mt-5">
        <p>
            <em><small>Page rendered in {elapsed_time} seconds</small></em>
            <br />
        </p>
    </footer>
</body>

```

```

        CodeIgniter <?= CodeIgniter\CodeIgniter::CI_VERSION ?>
    </p>
</footer>
<script src="https://code.jquery.com/jquery-
3.4.1.slim.min.js" integrity="sha384-
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n" crossorigin="an
onymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.mi
n.js" integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="an
onymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.
min.js" integrity="sha384-
wfSDF2E50Y2D1uUdj003uMBJnjuUD4IH7YwaYd1iqfktj0Uod8GCExl30g8ifwB6" crossorigin="an
onymous"></script>

<?= $this->renderSection('extra-js') ?>

</body>
</html>

```

Selanjutnya kita buat folder bernama pegawai di dalam folder app/Views. Setelah itu buatlah file bernama list.php di dalam folder pegawai. Halaman view ini digunakan untuk menampilkan data. Ketikkan script berikut pada list.php:

```

<?= $this->extend('base') ?>

<?= $this->section('content') ?>
<div class="container mt-5">
    <div class="row">
        <div class="col-md-12">
            <div class="card">
                <div class="card-header">
                    <i class="fa fa-list"></i>&ampnbspPegawai
                    <div class="float-right">
                        <a href=<?php echo base_url('pegawai/add'); ?>" class="b
tn btn-success btn-sm"><i class="fa fa-plus-circle"></i>&ampnbspAdd Record</a>
                        <a style="margin: 2px;" href=<?php echo base_url('pegawai
'); ?>" class="btn btn-primary btn-sm"><i class="fa fa-
refresh"></i>&ampnbspRefresh</a>
                    </div>
                </div>
                <div class="card-body">
                    <?php if (session()->getFlashdata('success')) { ?>

```

```

<div class="alert alert-success">
    <?php echo session()->getFlashdata('success'); ?>
</div>
<?php } ?>
<?php if (session()->getFlashdata('error')) { ?>
    <div class="alert alert-danger">
        <?php echo session()->getFlashdata('error'); ?>
    </div>
<?php } ?>
<table class="table table-striped table-
bordered" style="font-style: Calibri;font-size:11px">
    <thead>
        <tr>
            <th scope="col">Action</th>
            <th scope="col">NIP</th>
            <th scope="col">Nama Pegawai</th>
            <th scope="col">Alamat</th>
            <th scope="col">Telepon</th>
            <th scope="col">Created Date</th>
        </tr>
    </thead>
    <tbody>
        <?php if (!empty($result) && is_array($result)) { ?>
            <?php foreach ($result as $row) { ?>
                <tr>
                    <td>
                        <a href="#" title="View"><span style=
"font-size: 1em; color: Mediumslateblue;"><i class="fa fa-
eye"></i></span></a>&ampnbsp
                            <a href=<?php echo base_url('pegawai
/edit/' . $row['id_pegawai']); ?>" title="Edit"><span style="font-
size: 1em; color: Dodgerblue;"><i class="fa fa-edit"></i></span></a>&ampnbsp
                            <a href=<?php echo base_url('pegawai
/destroy/' . $row['id_pegawai']); ?>" title="Delete"><span style="font-
size: 1em; color: Tomato;"><i class="fa fa-trash"></span></i></a>
                    </td>
                    <td><?php echo $row['nip']; ?></td>
                    <td><?php echo $row['nama_pegawai']; ?></
td>
                    <td><?php echo $row['alamat']; ?></td>
                    <td><?php echo $row['telp']; ?></td>
                    <td><?php echo $row['created_at'] ?></td>
                </tr>
            <?php } ?>
        <?php } else { ?>

```

```

        <tr>
            <td colspan="5" class="text-
center">No record found !</td>
        </tr>
    <?php } ?>
</tbody>
</table>
<?= $pager->links(); ?>
</div>

</div>
</div>
</div>
</div>
<?= $this->endSection() ?>

<?= $this->section('extra-js') ?>
<script>
    $(document).ready(function() {
        $('.pagination li').addClass('page-item');
        $('.pagination li a').addClass('page-link');
    })
</script>
<?= $this->endSection() ?>

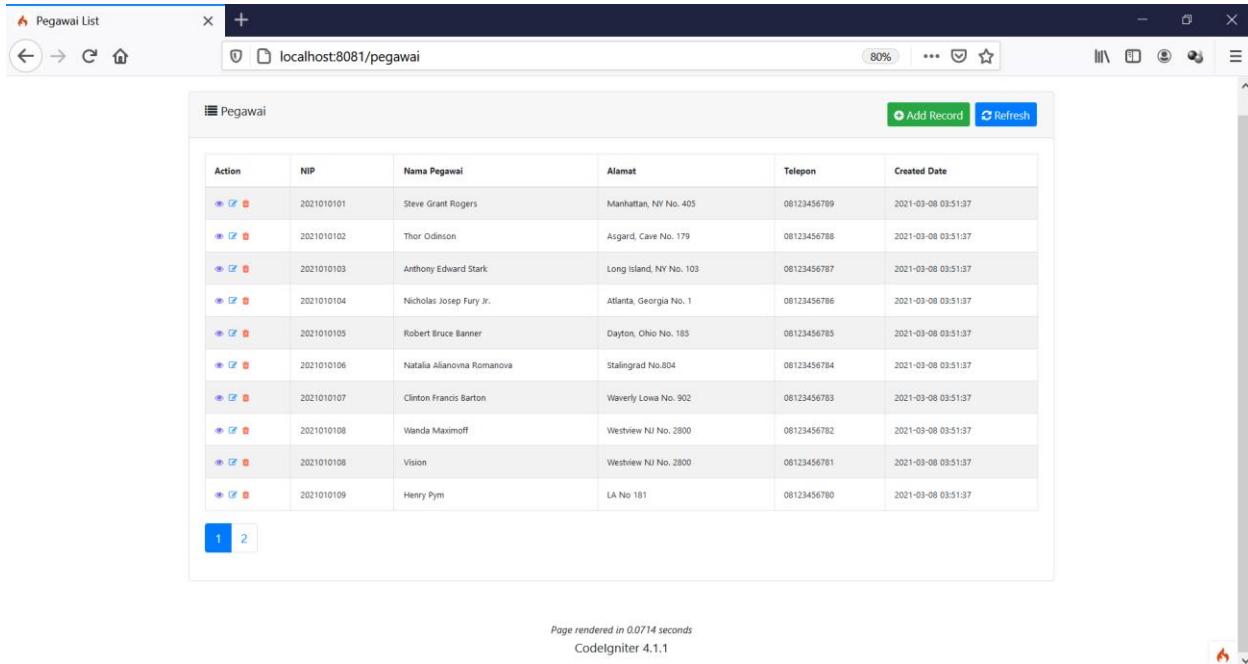
```

#### 5.2.4. Membuat Routes

Buka file Routes.php folder app/Config kemudian tambahkan routing berikut:

```
$routes->get('pegawai', 'PegawaiController::index');
```

Jalankan local development server menggunakan perintah php spark serve dan masukkan halaman <http://localhost:8081/pegawai> di dalam browser sehingga menampilkan tampilan seperti ini:



Gambar 5.4 Tampilan data pegawai pada halaman web

## 5.3. Membuat Add Data

### 5.3.1. Edit Controller

Bukalah Controller PegawaiController di folder App/Controllers. Kemudian tambahkan kode di bagian fungsi add dan store seperti berikut :

```
public function add() {
    session();
    $data = [
        'title' => 'Add Record Pegawai',
        'validation' => \Config\Services::validation()
    ];
    return view('pegawai/add', $data);
}

public function store() {
    if (!$this->validate([
        'nip'          => 'required|min_length[5]|max_length[10]|is_unique[pegawai.nip]',
        'nama_pegawai' => 'required',
        'alamat'       => 'required',
        'telp'         => 'required'
    ]))
```

```

        ]);
        return redirect()->to('/pegawai/add')->withInput();
    }
    $nip = $this->request->getPost('nip');
    $nama_pegawai = $this->request->getPost('nama_pegawai');
    $alamat = $this->request->getPost('alamat');
    $telp = $this->request->getPost('telp');
    $pegawai = [
        'nip'          => $nip,
        'nama_pegawai' => $nama_pegawai,
        'alamat'       => $alamat,
        'telp'         => $telp,
    ];
    $save = $this->model->save($pegawai);
    if ($save) {
        session()->
        >setflashdata('success', 'Record has been added successfully.');
        return redirect()->to(base_url('pegawai'));
    } else {
        session()->
        >setflashdata('error', 'Some problems occurred, please try again.');
        return redirect()->back();
    }
}

```

### 5.3.2. Menambah View add.php

Berikutnya adalah kita membuat form untuk memasukkan data ke dalam database. Form ini kita beri nama add.php dan dimasukkan dalam folder Views/pegawai. Masukkan kode berikut ini ke dalam file add.php:

```

<?= $this->extend('base') ?>
<?= $this->section('content') ?>
<div class="container mt-5">
    <div class="row">
        <div class="col-md-6">
            <div class="card">
                <div class="card-header">
                    <?= $title ?>
                </div>
                <div class="card-body">
                    <?= form_open('pegawai/store'); ?>
                    <?= csrf_field(); ?>
                    <div class="form-group">
                        <label for="nip">NIP</label>

```

```

        <input type="text" name="nip" class="form-
control <?= ($validation->hasError('nip')) ? 'is-invalid' : ''; ?>" value="<?= old('nip'); ?>">
        <div class="invalid-feedback">
            <?= $validation->getError('nip'); ?>
        </div>
    </div>
    <div class="form-group">
        <label for="nama_pegawai">Nama Pegawai</label>
        <input type="text" name="nama_pegawai" class="form-
control <?= ($validation->hasError('nama_pegawai')) ? 'is-invalid' : ''; ?>" value="<?= old('nama_pegawai'); ?>">
        <div class="invalid-feedback">
            <?= $validation->getError('nama_pegawai'); ?>
        </div>
    </div>
    <div class="form-group">
        <label for="alamat">Alamat</label>
        <input type="text" name="alamat" class="form-
control <?= ($validation->hasError('alamat')) ? 'is-invalid' : ''; ?>" value="<?= old('alamat'); ?>">
        <div class="invalid-feedback">
            <?= $validation->getError('alamat'); ?>
        </div>
    </div>
    <div class="form-group">
        <label for="telp">Telepon</label>
        <input type="text" name="telp" class="form-
control <?= ($validation->hasError('telp')) ? 'is-invalid' : ''; ?>" value="<?= old('telp'); ?>">
        <div class="invalid-feedback">
            <?= $validation->getError('telp'); ?>
        </div>
    </div>
    <div class="form-group">
        <a href="<?= base_url('pegawai') ?>" class="btn btn-
primary"><i class="fa fa-arrow-circle-left"></i>&nbsp;Back</a>
        <button class="btn btn-success"><i class="fa fa-
save"></i>&nbsp;Save</button>
        </div>
        <?= form_close(); ?>
    </div>
</div>
</div>

```

```
</div>
<?= $this->endSection() ?>
```

### 5.3.3. Membuat Routes Untuk Add dan Store

Buka file Routes.php folder app/Config kemudian tambahkan routing berikut:

```
$routes->get('pegawai/add', 'PegawaiController::add');
$routes->post('pegawai/store', 'PegawaiController::store');
```

Routes pertama untuk memanggil fungsi add di controller PegawaiController. Jika kita buka halaman web pada <http://localhost:8081/pegawai> dan klik tombol Add Record, maka halaman akan dialihkan ke <http://localhost:8081/pegawai/add> di dalam browser sehingga menampilkan tampilan seperti ini:

The screenshot shows a simple web form titled "Add Record Pegawai". It contains four input fields: "NIP", "Nama Pegawai", "Alamat", and "Telepon". Below the fields are two buttons: a blue "Back" button and a green "Save" button.

Gambar 5.5 Tampilan Add Record Data Pegawai

Keterangan:

- Tombol Back digunakan untuk kembali ke halaman sebelumnya.
- Tombol Save untuk menyimpan data

Pada saat pengguna klik tombol save, maka rute akan dialihkan ke fungsi store di controller PegawaiController lalu data akan tersimpan ke dalam database. Sekarang, kita ambil contoh untuk memasukkan data seperti di bawah:

Add Record Pegawai

NIP  
2021010112

Nama Pegawai  
Sharon Carter

Alamat  
Unknown Madripoor

Telepon  
08123456777

Back Save

Gambar 5.6 Tampilan contoh data pegawia

Setelah itu, tekan tombol Save, maka data akan tersimpan dan halaman akan dialihkan ke halaman menampilkan data dan muncul pesan info seperti di bawah:

Record has been added successfully.

Jika kita melihat di halaman terakhir, terlihat data telah berhasil masuk.

Pegawai

Add Record Refresh

Action	NIP	Nama Pegawai	Alamat	Telepon	Created Date
<span style="color: blue;">Eye</span> <span style="color: blue;">Edit</span> <span style="color: red;">Delete</span>	2021010109	Henry Pym	LA No 181	08123456780	2021-03-08 03:51:37
<span style="color: blue;">Eye</span> <span style="color: blue;">Edit</span> <span style="color: red;">Delete</span>	2021010110	Scott Lang	Rhinebeck, NY No. 604	08123456779	2021-03-08 03:51:37
<span style="color: blue;">Eye</span> <span style="color: blue;">Edit</span> <span style="color: red;">Delete</span>	2021010111	Gamora Zen Whoberi Ben Titan	Zen-Whoberis No. 116	08123456778	2021-03-08 03:51:37
<span style="color: blue;">Eye</span> <span style="color: blue;">Edit</span> <span style="color: red;">Delete</span>	2021010112	Sharon Carter	Unknown Madripoor	08123456777	2021-04-28 05:10:48

1 2

Gambar 6.7 Tampilan data yang telah berhasil dimasukkan

## 5.4. Membuat Edit Data

### 5.4.1. Edit Controller

Bukalah Controller PegawaiController di folder App/Controllers. Kemudian tambahkan kode di bagian fungsi edit dan update seperti berikut :

```

public function edit($id)
{
    session();
    $pegawai = $this->model->find($id);
    if (empty($pegawai)) {
        session()->setflashdata('error', 'Record not found');
        return redirect()->back();
    }
    $data = [
        'title'      => 'Edit Pegawai',
        'result'     => $pegawai,
        'validation' => \Config\Services::validation()
    ];
    return view('pegawai/edit', $data);
}

public function update($id)
{
    if (!$this->validate([
        'nip'          => 'required|min_length[5]|max_length[15]|is_unique[pegawai.nip,id_pegawai,{id_pegawai}]',
        'nama_pegawai' => 'required',
        'alamat'       => 'required',
        'telp'         => 'required'
    ])){
        return redirect()->back()->withInput();
    }
    $nip = $this->request->getPost('nip');
    $nama_pegawai = $this->request->getPost('nama_pegawai');
    $alamat = $this->request->getPost('alamat');
    $telp = $this->request->getPost('telp');
    $pegawai = [
        'nip'          => $nip,
        'nama_pegawai' => $nama_pegawai,
        'alamat'       => $alamat,
        'telp'         => $telp
    ];
    $update = $this->model->update($id, $pegawai);
    if ($update) {
        session()->
>setflashdata('success', 'Record has been updated successfully');
        return redirect()->to(base_url('pegawai'));
    } else {
}

```

```

        session()->
>setflashdata('error', 'Some problems occurred, please try again.');
        return redirect()->back();
    }
}

```

#### 5.4.2. Menambah View edit.php

Berikutnya adalah kita membuat form untuk mengubah data ke dalam database. Form ini kita beri nama edit.php dan dimasukkan dalam folder Views/pegawai. Masukkan kode berikut ini ke dalam file edit.php:

```

<?= $this->extend('base') ?>
<?= $this->section('content') ?>
<div class="container mt-5">
    <div class="row">
        <div class="col-md-6">
            <div class="card">
                <div class="card-header">
                    <?= $title ?>
                </div>
                <div class="card-body">
                    <?= form_open('pegawai/update/'. $result['id_pegawai']); ?>
                    <input type="hidden" name="id_pegawai" value="<?= $result['id_pegawai'];?>" class="form-control" required>
                    <div class="form-group">
                        <label for="nip">NIP</label>
                        <input type="text" name="nip" value="<?= $result['nip'];?>" class="form-control <?= ($validation->hasError('nip')) ? 'is-invalid' : ''; ?>">
                        <div class="invalid-feedback">
                            <?= $validation->getError('nip'); ?>
                        </div>
                    </div>
                    <div class="form-group">
                        <label for="nama_pegawai">Nama Pegawai</label>
                        <input type="text" name="nama_pegawai" value="<?= $result['nama_pegawai'];?>" class="form-control <?= ($validation->hasError('nama_pegawai')) ? 'is-invalid' : ''; ?>">
                        <div class="invalid-feedback">
                            <?= $validation->getError('nama_pegawai'); ?>
                        </div>
                    </div>
                    <div class="form-group">

```

```

        <label for="alamat">Alamat</label>
        <input type="text" name="alamat" value=<?= $result['alamat']; ?>" class="form-control <?= ($validation->hasError('alamat')) ? 'is-invalid' : ''; ?>">
            <div class="invalid-feedback">
                <?= $validation->getError('alamat'); ?>
            </div>
        </div>
        <div class="form-group">
            <label for="telp">Telepon</label>
            <input type="text" name="telp" value=<?= $result['telp'] ; ?>" class="form-control <?= ($validation->hasError('telp')) ? 'is-invalid' : ''; ?>">
                value=<?= old('telp'); ?>">
            <div class="invalid-feedback">
                <?= $validation->getError('telp'); ?>
            </div>
        </div>
        <div class="form-group">
            <a href=<?= base_url('pegawai') ?>" class="btn btn-primary"><i class="fa fa-arrow-circle-left"></i>&ampnbspBack</a>
            <button class="btn btn-success"><i class="fa fa-save"></i>&ampnbspUpdate</button>
        </div>
        <?= form_close(); ?>
    </div>
</div>
<?= $this->endSection() ?>

```

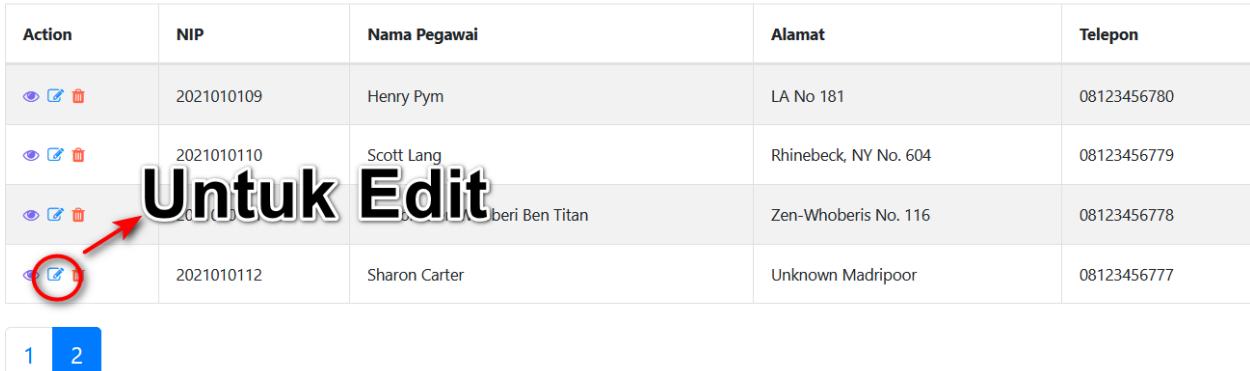
#### 5.4.3. Membuat Routes Untuk Edit dan Update

Buka file Routes.php folder app/Config kemudian tambahkan routing berikut:

```
$routes->get('pegawai/edit/(:num)', 'PegawaiController::edit/$1');
$routes->post('pegawai/update', 'PegawaiController::update');
```

Routes pertama untuk memanggil fungsi edit di controller PegawaiController. Jika kita buka halaman web pada <http://localhost:8081/pegawai> dan klik tombol edit, maka halaman akan dialihkan ke <http://localhost:8081/pegawai/edit/> di dalam browser. Perhatikan langkah-langkah berikut:

1. Buka halaman <http://localhost:8081/pegawai> dan klik tombol edit seperti di gambar bawah:

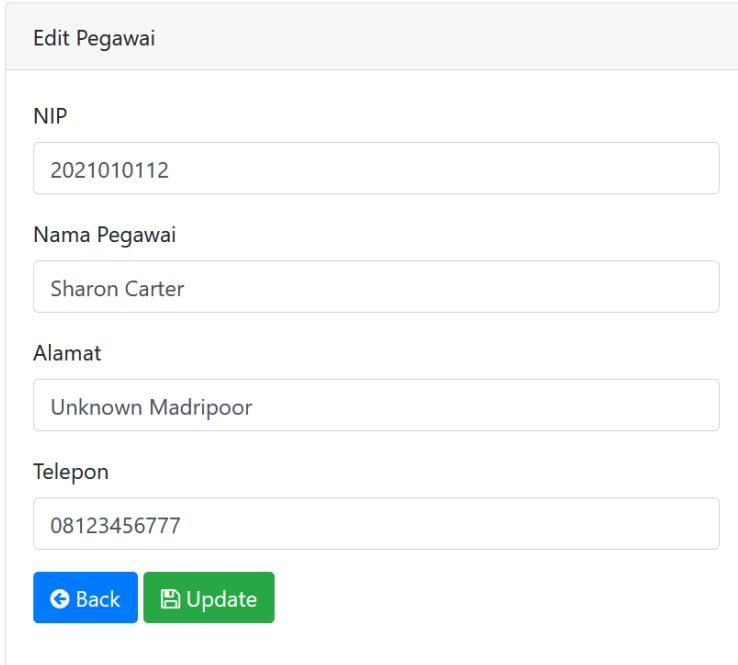


A screenshot of a table displaying employee information. The columns are labeled: Action, NIP, Nama Pegawai, Alamat, and Telepon. There are five rows of data. In the first row, the 'Action' column contains icons for view, edit, and delete. An arrow points from the text 'Untuk Edit' to the edit icon in the first row. A red circle highlights the edit icon in the fourth row. Below the table, there are two blue buttons labeled '1' and '2'.

Action	NIP	Nama Pegawai	Alamat	Telepon
	2021010109	Henry Pym	LA No 181	08123456780
	2021010110	Scott Lang	Rhinebeck, NY No. 604	08123456779
	2021010111	Beri Ben Titan	Zen-Whoberis No. 116	08123456778
	2021010112	Sharon Carter	Unknown Madripoor	08123456777

Gambar 5.8 Tampilan icon Edit

2. Maka akan muncul halaman edit:



A screenshot of an 'Edit Pegawai' form. The form fields are: NIP (2021010112), Nama Pegawai (Sharon Carter), Alamat (Unknown Madripoor), and Telepon (08123456777). At the bottom are two buttons: 'Back' and a green 'Update' button.

Gambar 5.9 Tampilan Form Edit Pegawai

3. Ubah nama pegawai dan alamat menjadi seperti di bawah:

Edit Pegawai

NIP

Nama Pegawai

Alamat

Telepon

[Back](#) [Update](#)

Gambar 5.10 Contoh mengubah data pegawai

4. Tekan tombol Update untuk menyimpan hasil perubahan dan halaman akan dialihkan ke halaman menampilkan data dan muncul pesan info seperti di bawah:

Record has been updated successfully

5. Jika kita melihat di halaman terakhir, terlihat data telah berhasil diubah

Action	NIP	Nama Pegawai	Alamat	Telepon	Created Date
	2021010110	Scott Lang	Rhinebeck, NY No. 604	08123456779	2021-03-08 03:51:37
	2021010111	Gamora Zen Whoberi Ben Titan	Zen-Whoberis No. 116	08123456778	2021-03-08 03:51:37
	2021010112	Power Broker	Los Angeles, CA	08123456778	2021-04-28 05:10:48
1 2					

Gambar 5.11 Tabel hasil perubahan

## 5.5. Membuat Hapus Data

### 5.5.1. Edit Controller

Bukalah Controller PegawaiController di folder App/Controllers. Kemudian tambahkan kode di bagian fungsi destroy seperti berikut :

```
public function destroy($id)
{
    if (empty($id)) {
        return redirect()->to(base_url('pegawai'));
    }
    $delete = $this->model->delete($id);
    if ($delete) {
        session()->
>setflashdata('success', 'Record has been removed successfully.');
        return redirect()->to(base_url('pegawai'));
    } else {
        session()->
>setflashdata('error', 'Some problems occurred, please try again.');
        return redirect()->to(base_url('pegawai'));
    }
}
```

### 5.5.2. Membuat Routes Untuk Delete

Buka file Routes.php folder app/Config kemudian tambahkan routing berikut:

```
$routes->get('pegawai/destroy/(:num)', 'PegawaiController::destroy/$1');
```

Routes untuk memanggil fungsi destroy di controller PegawaiController. Jika kita buka halaman web pada <http://localhost:8081/pegawai> dan klik tombol delete, maka data akan langsung terhapus. Perhatikan langkah-langkah berikut:

1. Buka halaman <http://localhost:8081/pegawai> dan klik tombol delete seperti di gambar bawah:

Action	NIP	Nama Pegawai	Alamat	Telepon	Created Date
	20101010	Scott Lar	Rhinebeck, NY No. 604	08123456779	2021-03-08 03:51:37
	20101011	Gamora Zen Whoberi Ben Titan	Zen-Whoberis No. 116	08123456778	2021-03-08 03:51:37
	201010112	Power Broker	Los Angeles, CA	08123456778	2021-04-28 05:10:48

Gambar 5.12 Tampilan icon delete

2. Dalam halaman akan muncul pesan info seperti di bawah:

Record has been removed successfully.

3. Jika kita melihat di halaman terakhir, sudah tidak terlihat lagi data yang kita hapus

Action	NIP	Nama Pegawai	Alamat	Telepon	Created Date
	2021010110	Scott Lang	Rhinebeck, NY No. 604	08123456779	2021-03-08 03:51:37
	2021010111	Gamora Zen Whoberi Ben Titan	Zen-Whoberis No. 116	08123456778	2021-03-08 03:51:37

Gambar 5.13 Tampilan perubahan data pada tabel yang telah terhapus

## 5.6. Membuat Info Detail Data

### 5.6.1. Edit Controller

Bukalah Controller PegawaiController di folder App/Controllers. Kemudian tambahkan kode di bagian fungsi show seperti berikut :

```
public function show($id)
{
    $pegawai = $this->model->find($id);
    if (empty($pegawai)) {
        session()->setFlashdata('error', 'Record not found');
        return redirect()->back();
    }
    $data = [
        'title'      => 'View Pegawai',
        'result'     => $pegawai
    ];
    return view('pegawai/view', $data);
}
```

### 5.6.2. Menambah View view.php

Berikutnya adalah kita membuat form untuk mengubah data ke dalam database. Form ini kita beri nama edit.php dan dimasukkan dalam folder Views/pegawai. Masukkan kode berikut ini ke dalam file view.php:

```

<?= $this->extend('base') ?>
<?= $this->section('content') ?>


<?= $title ?>



| NIP          | <?= \$result['nip'];?>          |
|--------------|---------------------------------|
| Nama Pegawai | <?= \$result['nama_pegawai'];?> |
| Alamat       | <?= \$result['alamat'];?>       |
| Telp         | <?= \$result['telp'];?>         |
| Created Date | <?= \$result['created_at'];?>   |

</i>&ampnbspBack</a>


<?= $this->endSection() ?>


```

### 5.6.3. Menambah link pada list.php

Berikutnya adalah kita tambahkan link untuk melihat detail dari baris data yang dipilih. Link ini kita masukkan pada bagian kode berikut:

```
<a href="#" title="View"><span style="font-size: 1em; color: Mediumslateblue;"><i class="fa fa-eye"></i></span></a>&ampnbsp
```

Diubah menjadi

```
<a href="=base_url('pegawai/show/' . $row['id_pegawai'])?" title="View"><span style="font-size: 1em; color: Mediumslateblue;"><i class="fa fa-eye"></i></span></a>&ampnbsp
```

### 5.6.4. Membuat Routes Untuk Detail Data

Buka file Routes.php folder app/Config kemudian tambahkan routing berikut:

```
$routes->get('pegawai/show/(:num)', 'PegawaiController::show/$1');
```

Routes untuk memanggil fungsi show di controller PegawaiController. Jika kita buka halaman web pada <http://localhost:8081/pegawai> dan klik tombol view (simbol mata), maka halaman akan dialihkan ke view.php. Perhatikan langkah-langkah berikut:

1. Buka halaman <http://localhost:8081/pegawai> dan klik tombol delete seperti di gambar bawah:

Action	NIP	Nama Pegawai	Alamat	Telepon	Created Date
	2021010101	Steve Grant Rogers	Manhattan, NY No. 405	08123456789	2021-03-08 03:51:37
	2021010102	Thor Odinson	Asgard, Cave No. 179	08123456788	2021-03-08 03:51:37
	2021010103	Anthony Edward Stark	Long Island, NY No. 103	08123456787	2021-03-08 03:51:37
	2021010104	Nicholas Josep Fury Jr.	Atlanta, Georgia No. 1	08123456786	2021-03-08 03:51:37

Gambar 5.14 Tampilan icon view

2. Maka akan muncul halaman detail data (view):

View Pegawai	
<b>NIP</b>	2021010101
<b>Nama Pegawai</b>	Steve Grant Rogers
<b>Alamat</b>	Manhattan, NY No. 405
<b>Telp</b>	08123456789
<b>Created Date</b>	2021-03-08 03:51:37

[Back](#)

Gambar 5.15 Tampilan info detail pegawai

## LATIHAN

1. Buatlah sebuah projek web CRUD yang berisi data sebagai berikut:
  - a. Nama
  - b. Tempat dan Tanggal Lahir
  - c. Alamat
  - d. No Telepon
  - e. Jenis Kelamin
  - f. Pendidikan

# BAB 6

## MEMBUAT OTENTIKASI

### Tujuan instruksional:

- Mampu membuat halaman untuk otentikasi pengguna

## 6.1. Prinsip Manajemen Keamanan

Kegiatan mengamankan informasi muncul bersamaan dengan ide menyimpan informasi (data). Prinsip manajemen keamanan mendefinisikan parameter dasar yang diperlukan untuk mengamankan suatu lingkungan kerja. Parameter tersebut selanjutnya digunakan sebagai acuan untuk melakukan desain, implementasi, dan administrasi suatu sistem. Pertimbangan terhadap parameter-parameter tersebut wajib dilakukan oleh profesional keamanan sistem karena pertumbuhan perangkat komunikasi dan komputasi semakin meningkat.

Pertumbuhan perangkat komunikasi dan komputasi tentu saja dibarengi dengan peningkatan layanan dan pengguna. Peningkatan layanan tersebut tentu saja berakibat pada meningkatnya celah keamanan dan makin banyaknya serangan yang mungkin terjadi. Tujuan utama dari manajemen keamanan terbagi dalam 3 prinsip yaitu confidentiality, integrity, dan availability. Kendali keamanan harus dialamatkan pada ketiga prinsip tersebut yang selanjutnya akan dievaluasi berdasarkan ancaman yang terjadi pada ke salah satu atau lebih dari prinsip tersebut. Ketiga prinsip ini dapat dipertimbangkan sebagai tujuan akhir dan kebutuhan suatu organisasi.

### 6.1.1. Confidentiality

Confidentiality (kerahasiaan) adalah perlindungan informasi dalam sistem agar pihak yang tidak berhak tidak dapat mengaksesnya. Banyak yang mempercayai prinsip ini merupakan prinsip terpenting dalam dunia militer dan pemerintahan agar rahasia militer maupun negara tetap terjaga. Namun, suatu perusahaan atau organisasi masyarakat perlu pula untuk memanfaatkan prinsip ini sejauh dapat memberikan dampak signifikan terhadap data perusahaan sehingga rahasia perusahaan tetap terjaga dari kompetitor atau pihak lain yang menginginkan informasi yang bersifat sensitif.

Permasalahan privacy meningkat dengan cepat dalam beberapa tahun terakhir yang pada akhirnya menempatkan confidentiality sebagai prinsip dalam mengamankan informasi khususnya pada sistem yang terotomatisasi dan terintegrasi.

Confidentiality harus didefinisikan dengan baik dan prosedur untuk menjaga hal ini harus diimplementasikan secara hati-hati. Hal yang paling krusial dalam mengimplementasikan confidentiality adalah identifikasi dan otentikasi pengguna. Metode identifikasi pada tiap pengguna mampu secara efektif memastikan kebijakan yang diberikan sesuai data yang diperlukan.

### 6.1.2. Integrity

Integrity merupakan perlindungan terhadap data dari perubahan yang tidak diijinkan secara sengaja maupun tidak. Tantangan dari program yang mendukung keamanan adalah data harus dijaga pada keadaan yang diharapkan oleh pengguna, meskipun ketidaksengajaan terhadap perubahan data sangat dimungkinkan terjadi.

Elemen tambahan dari integritas adalah kebutuhan untuk melindungi proses atau program dari perubahan yang tidak diijinkan, khususnya mencegah terjadinya kejahatan maupun kesalahan yang mempengaruhi integritas data. Hal ini sangat sulit dilakukan, karena kesalahan sangat mungkin terjadi ketika suatu data dimodifikasi oleh pengguna yang memiliki ijin, misalnya terjadinya kerusakan file. Prinsip integritas memastikan tidak adanya kesalahan terhadap data yang dijaga.

Untuk memastikan integritas suatu data dapat dijaga, proses otentikasi dan identifikasi pengguna merupakan hal utama yang perlu diperhatikan. Integritas tergantung kepada access control yang akan mengidentifikasi pengguna yang berusaha mengakses.

Tujuan dari integritas adalah:

1. Mencegah pengguna yang tidak berhak memodifikasi data atau program
2. Mencegah pengguna yang memiliki hak memodifikasi yang tak diijinkan atau tidak sesuai ketentuan
3. Menjaga konsistensi data dan program secara internal maupun eksternal

### 6.1.3. Availability

Ketersediaan adalah jaminan bahwa sistem komputer dapat diakses oleh pengguna yang diijinkan ketika diperlukan. Kehilangan kemampuan memproses data yang diakibatkan oleh bencana dapat diatasi dengan merencanakan suatu tempat alternatif ketika bencana terjadi sehingga layanan tetap dapat bekerja meskipun sistem komputer utama mengalami gangguan.

Secara teknis langkah lain yang dapat diambil untuk mengamankan ketersediaan layanan adalah menjaga agar pengguna yang tidak berhak tidak menggunakan sistem komputer, mempersiapkan sistem cadangan.

## 6.2 Identification

Identifikasi adalah proses dimana suatu subyek (pengguna maupun sistem) dikenali dan dimulainya pertanggungjawaban. Subyek harus memberikan identitas pada sistem untuk memulai proses otentikasi, otorisasi, dan pertanggungjawaban. Pemberian identitas dapat berupa penulisan username, penggesekan smartcard, penggunaan token, ungkapan dengan suara, pemotretan wajah, sidik jadi atau lainnya. Pemberian suatu nomor identifikasi (id number) juga merupakan suatu proses identifikasi. Tanpa identitas suatu sistem tidak memiliki metode untuk menghubungkan suatu otentikasi dengan subyek.

Apabila suatu subyek telah teridentifikasi maka subyek tersebut dapat dipertanggungjawabkan apabila melakukan suatu proses, jadi sebenarnya sistem mengidentifikasi apa yang dikerjakan atau ditugaskan oleh suatu subyek berdasarkan data identifikasi yang telah diperoleh biasanya melalui suatu id number.

## 6.3. Otentikasi

Otentikasi merupakan suatu proses untuk memverifikasi atau menguji suatu klaim terhadap identitas dianggap valid. Otentikasi membutuhkan tambahan informasi dari subyek yang harus sesuai dengan identifikasi yang diindikasikan. Metode otentikasi yang paling umum digunakan adalah password. Sebenarnya metode otentikasi ini membandingkan suatu faktor dengan data identitas yang valid yang telah disimpan sebelumnya dalam suatu database (misal: database pengguna) apabila data yang dimasukkan oleh subyek ketika terjadi proses otentikasi dianggap valid (sama dengan data yang terdapat dalam database) maka subyek akan diijinkan dan bila berbeda maka akan ditolak. Otentikasi digunakan untuk memverifikasi identitas terhadap informasi yang bersifat rahasia. Kemampuan suatu sistem dalam melakukan otentikasi dapat dianggap mewakili tingkat keamanan suatu sistem.

Identifikasi dan otentikasi sebenarnya merupakan satu kesatuan yang terjadi dalam satu kesatuan proses yang tidak dapat dipisahkan. Umumnya otentikasi menggunakan prinsip "something you know" atau "something you have" sehingga sistem dapat melakukan validasi terhadap identitas subyek karena hanya subyek tersebutlah yang memiliki atau mengetahui hal tersebut.

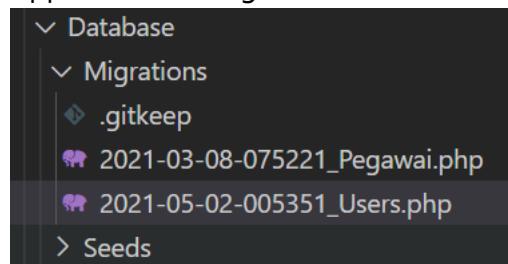
## 6.4. Membuat Form Register

### 6.4.1. Membuat Tabel Users

Pertama kita membuat tabel yang akan digunakan sebagai data untuk identifikasi ketika verifikasi login. Kita buat menggunakan fitur migrasi.

```
D:\xampp\htdocs\myproject>php spark migrate:create users
CodeIgniter v4.1.1 Command Line Tool - Server Time: 2021-05-01 19:53:51 UTC-05:00
File created: APPPATH\Database\Migrations\2021-05-02-005351_Users.php
```

Selanjutnya akan terbentuk file baru dengan nama 2021-XXXX\_Users.php pada folder App/Database/Migrations



Buka file tersebut kemudian edit menjadi seperti berikut:

```
<?php

namespace App\Database\Migrations;

use CodeIgniter\Database\Migration;

class Users extends Migration
{
    public function up() {
        $this->forge->addField([
            'user_id' => [
                'type' => 'INT',
                'unsigned' => TRUE,
                'auto_increment' => TRUE
            ],
            'user_name' => [
                'type' => 'VARCHAR',
                'constraint' => 100,
                'null' => FALSE
            ]
        ]);
    }

    public function down() {
        $this->forge->dropField('user_id');
    }
}
```

```

        ],
        'user_email' => [
            'type' => 'VARCHAR',
            'constraint' => 100,
            'null' => FALSE
        ],
        'user_password' => [
            'type' => 'VARCHAR',
            'constraint' => 100,
            'null' => FALSE
        ],
        'created_at' => [
            'type' => 'datetime',
            'null' => TRUE
        ],
        'updated_at' => [
            'type' => 'datetime',
            'null' => TRUE
        ]
    ]);
}

$this->forge->addKey('user_id', TRUE);
$this->forge->createTable('users');
}

public function down() {
    $this->forge->dropTable('users');
}
}

```

Kemudian dari command prompt jalankan perintah berikut untuk membuat tabel users:

```

D:\xampp\htdocs\myproject>php spark migrate

CodeIgniter v4.1.1 Command Line Tool - Server Time: 2021-05-01 20:12:46 UTC-05:00

Running all new migrations...
    Running: (App) 2021-05-02-005351_App\Database\Migrations\Users
Done migrations.

```

Kita bisa melihat hasilnya dengan membuka database di perangkat yang ada, misal kita menggunakan phpmyadmin. Hasilnya terlihat seperti berikut:

	Table	Action
<input type="checkbox"/>	migrations	
<input type="checkbox"/>	pegawai	
<input type="checkbox"/>	users	
3 tables		Sum

Gambar 6.1. Tabel users

Jika kita klik tabel users, maka akan terlihat struktur tabel sesuai dengan yang telah didefinisikan pada kode di atas.

Table structure		Relation view							
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	
1	user_id	int(10)		UNSIGNED	No	None		AUTO_INCREMENT	
2	user_name	varchar(100)	utf8_general_ci		No	None			
3	user_email	varchar(100)	utf8_general_ci		No	None			
4	user_password	varchar(100)	utf8_general_ci		No	None			
5	created_at	datetime			Yes	NULL			
6	updated_at	datetime			Yes	NULL			

Gambar 6.2. Struktur tabel users

#### 6.4.2. Membuat Model Users

Proses menampilkan data yang telah dibuat umumnya ditampilkan dalam bentuk sebuah tabel. Pertama, kita buat terlebih dahulu Model yang sesuai. Untuk membuat file model bisa dengan cara membuat file baru dengan nama `Users.php` pada folder **app/Models** atau menggunakan composer dengan cara:

```
D:\xampp\htdocs\myproject>php spark make:model Users
CodeIgniter v4.1.1 Command Line Tool - Server Time: 2021-05-01 20:17:26 UTC-05:00
File created: APPPATH\Models\Users.php
```

kemudian ketikkan script berikut:

```
<?php
namespace App\Models;
```

```

use CodeIgniter\Model;

class User extends Model{
    protected $table = 'users';
    protected $primaryKey = 'user_id';
    protected $allowedFields = ['user_name', 'user_email', 'user_password'];
    protected $useTimestamps = true;
    protected $createdField = 'created_at';
    protected $updatedField = 'updated_at';
}

```

#### 6.4.3. Membuat Controller Register

Buat file controller baru dengan nama RegisterController menggunakan composer:

```

D:\xampp\htdocs\myproject>php spark make:controller RegisterController

CodeIgniter v4.1.1 Command Line Tool - Server Time: 2021-05-01 20:34:57 UTC-05:00

File created: APPPATH\Controllers\RegisterController.php

```

Kita juga bisa membuat file controller baru secara manual dengan memasukkannya ke dalam folder app/Controllers. Pada controller, kita akan membuat tiga fungsi, yaitu: fungsi index() dan save(). Fungsi index() untuk menampilkan sebuah view benama "register", sedangkan fungsi save() untuk menyimpan data ke tabel users yang ada di database sekaligus mengenkripsi password dengan fungsi password\_hash(). Kode lengkap dalam controller seperti berikut ini:

```

<?php

namespace App\Controllers;

use App\Controllers\BaseController;
use App\Models\Users;

class RegisterController extends BaseController
{
    protected $model;

    public function __construct() {
        $this->model = new Users();
        $this->helpers = ['form', 'url'];
    }

    public function index()

```

```

    {
        session();
        $data = [
            'title' => 'Register Form',
            'validation' => \Config\Services::validation()
        ];
        return view('users/register', $data);
    }

    public function save()
    {
        if (!$this->validate([
            'name'          => 'required',
            'email'         => 'required|valid_email|is_unique[users.user_email]'

            ,
            'password'      => 'required|min_length[6]|max_length[200]',
            'confpassword'  => 'matches[password]'
        ])){
            return redirect()->to(base_url('register'))->withInput();
        }
        $name = $this->request->getPost('name');
        $email = $this->request->getPost('email');
        $password = password_hash($this->request-
>getPost('password'), PASSWORD_DEFAULT);
        $users = [
            'user_name'     => $name,
            'user_email'    => $email,
            'user_password' => $password
        ];

        $save = $this->model->save($users);
        if ($save) {
            session()->
>setflashdata('success', 'Record has been added successfully.');
            return redirect()->to(base_url('register'));
        } else {
            session()->
>setflashdata('error', 'Some problems occurred, please try again.');
            return redirect()->back();
        }
    }
}

```

#### 6.4.4. Membuat View register.php

Berikutnya adalah kita membuat form untuk memasukkan data user ke dalam database. Form ini kita beri nama register.php dan dimasukkan dalam folder Views/users. Masukkan kode berikut ini ke dalam file register.php:

```
<?= $this->extend('base') ?>
<?= $this->section('content') ?>
<div class="container mt-5">
    <div class="row">
        <div class="col-md-6">
            <div class="card">
                <div class="card-header">
                    <?= $title ?>
                </div>
                <div class="card-body">
                    <?php if (session()->getFlashdata('success')) { ?>
                        <div class="alert alert-success">
                            <?php echo session()->getFlashdata('success'); ?>
                        </div>
                    <?php } ?>
                    <?php if (session()->getFlashdata('error')) { ?>
                        <div class="alert alert-danger">
                            <?php echo session()->getFlashdata('error'); ?>
                        </div>
                    <?php } ?>
                    <?= form_open('register/save'); ?>
                    <?= csrf_field(); ?>
                    <div class="form-group">
                        <label for="name">Name</label>
                        <input type="text" name="name" class="form-control" value="<?= old('name'); ?>">
                        <div class="invalid-feedback">
                            <?= $validation->getError('name'); ?>
                        </div>
                    </div>
                    <div class="form-group">
                        <label for="email">Email</label>
                        <input type="email" name="email" class="form-control" value="<?= old('email'); ?>">
                        <div class="invalid-feedback">
                            <?= $validation->getError('email'); ?>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

```

        </div>
    <div class="form-group">
        <label for="password">Password</label>
        <input type="password" name="password" class="form-control" <?= ($validation->hasError('password')) ? 'is-invalid' : ''; ?>" value="<?= old('password'); ?>">
        <div class="invalid-feedback">
            <?= $validation->getError('password'); ?>
        </div>
    </div>
    <div class="form-group">
        <label for="confpassword">Confirm Password</label>
        <input type="password" name="confpassword" class="form-control" <?= ($validation->hasError('confpassword')) ? 'is-invalid' : ''; ?>" value="<?= old('confpassword'); ?>">
        <div class="invalid-feedback">
            <?= $validation->getError('confpassword'); ?>
        </div>
    </div>
    <div class="form-group">
        <a href="<?= base_url('login') ?>" class="btn btn-primary"><i class="fa fa-arrow-circle-left"></i>&ampnbspBack</a>
        <button class="btn btn-success"><i class="fa fa-save"></i>&ampnbspRegister</button>
    </div>
    <?= form_close(); ?>
</div>
</div>
<?= $this->endSection() ?>

```

#### 6.4.5. Membuat Routes Untuk Index dan Save

Buka file Routes.php folder app/Config kemudian tambahkan routing berikut:

```
$routes->get('register', 'RegisterController::index');
$routes->post('register/save', 'RegisterController::save');
```

Routes pertama untuk memanggil fungsi index di controller RegisterController. Jika kita buka halaman web pada <http://localhost:8081/register>, maka akan tampil halaman berikut:

The screenshot shows a registration form titled "Register Form". It contains four input fields: "Name", "Email", "Password", and "Confirm Password", each with a corresponding text input box. Below the input fields are two buttons: a blue "Back" button with a left arrow icon and a green "Register" button with a document icon.

Gambar 6.3. Form Registrasi

Keterangan:

- Tombol Back digunakan untuk kembali ke login (catatan: halaman login akan dibuat di subbab berikutnya).
- Tombol Register untuk menyimpan data

Sekarang, kita coba masukkan data:

The screenshot shows the same registration form as in Gambar 6.3, but with data entered into the fields. The "Name" field contains "Sam Wilson", the "Email" field contains "captainamerica@avengers.com", and both the "Password" and "Confirm Password" fields contain "\*\*\*\*\*". The "Back" and "Register" buttons are visible at the bottom.

Gambar 6.4. Input Data Form Registrasi

Setelah itu, tekan tombol Register, maka data akan tersimpan dan muncul pesan info seperti di bawah:

Record has been added successfully.

Jika kita melihat di dalam database, terlihat data telah berhasil masuk.

<b>user_id</b>	<b>user_name</b>	<b>user_email</b>	<b>user_password</b>
1	Robert Downey	ironman@avengers.com	\$2y\$10\$DLT.T/hHINyAJXUWVP6YC.IZKp4/hd2FZ2rJmQ9rZBj...
2	Sam Wilson	captainamerica@avengers.com	\$2y\$10\$9vNt7TVjiBb1h0VPJyrCveUGUPfI14RdkMIS32yBQ6t...

Gambar 6.5. Data users dalam tabel di database

## 6.5. Membuat Form Otentikasi

### 6.5.1. Membuat Controller Login dan Dashboard

Buat file controller baru dengan nama LoginController menggunakan composer:

```
D:\xampp\htdocs\myproject>php spark make:controller LoginController
CodeIgniter v4.1.1 Command Line Tool - Server Time: 2021-05-01 22:32:39 UTC-05:00
File created: APPPATH\Controllers\LoginController.php
```

Kita juga bisa membuat file controller baru secara manual dengan memasukkannya ke dalam folder app/Controllers. Pada controller, kita akan membuat tiga fungsi, yaitu: fungsi index(), auth(), dan logout(). Fungsi index() untuk menampilkan sebuah view benama "login". Fungsi auth() untuk melakukan otentikasi dan membuat variable session jika otentikasi valid. Fungsi logout() untuk logout beserta menghapus variable session. Kode lengkap dalam controller seperti berikut ini:

```
<?php

namespace App\Controllers;

use App\Controllers\BaseController;
use App\Models\Users;

class LoginController extends BaseController
{
    protected $model;
```

```

public function __construct() {
    $this->model = new Users();
    $this->helpers = ['form', 'url'];
}
public function index()
{
    $data = [
        'title'      => 'Login Form'
    ];
    return view('users/login', $data);
}

public function auth()
{
    $session = session();
    $model = new Users();
    $email = $this->request->getPost('email');
    $password = $this->request->getPost('password');
    $data = $model->where('user_email', $email)->first();
    if($data){
        $pass = $data['user_password'];
        $verify_pass = password_verify($password, $pass);
        if($verify_pass){
            $ses_data = [
                'user_id'      => $data['user_id'],
                'user_name'    => $data['user_name'],
                'user_email'   => $data['user_email'],
                'logged_in'    => TRUE
            ];
            $session->set($ses_data);
            return redirect()->to('/dashboard');
        }else{
            $session->setflashdata('msg', 'Wrong Password');
            return redirect()->to('/login');
        }
    }else{
        $session->setflashdata('msg', 'Email not Found');
        return redirect()->to('/login');
    }
}

public function logout()
{
    $session = session();
    $session->destroy();
}

```

```

        return redirect()->to('/login');
    }
}

```

Kemudian, kita buat controller bernama DashboardController menggunakan composer

```
D:\xampp\htdocs\myproject>php spark make:controller DashboardController

CodeIgniter v4.1.1 Command Line Tool - Server Time: 2021-05-02 13:11:44 UTC-05:00

File created: APPPATH\Controllers\DashboardController.php
```

Dan masukkan kode berikut didalamnya:

```

<?php

namespace App\Controllers;

use App\Controllers\BaseController;

class DashboardController extends BaseController
{
    public function index()
    {
        $data = [
            'title'      => 'Dashboard'
        ];
        return view('users/dashboard', $data);
    }
}

```

### 6.5.2. Membuat View Login dan Dashboard

Berikutnya adalah kita membuat form login yang digunakan oleh pengguna untuk masuk ke dalam sistem. Form ini kita beri nama login.php dan dimasukkan dalam folder Views/users. Masukkan kode berikut ini ke dalam file login.php:

```

<?= $this->extend('base') ?>
<?= $this->section('content') ?>
<div class="container mt-5">
    <div class="row">
        <div class="col-md-6">
            <div class="card">
                <div class="card-header">
                    <?= $title ?>
                </div>
                <div class="card-body">
```

```

<div class="col-6">
    <h1>Sign In</h1>
    <?php if(session()->getFlashdata('msg')):>?
        <div class="alert alert-danger"><?= session()-
>getFlashdata('msg') ?></div>
        <?php endif;?>
        <?= form_open('login/auth'); ?>
        <?= csrf_field(); ?>
        <div class="mb-3">
            <label for="InputForEmail" class="form-
label">Email address</label>
            <input type="email" name="email" class="form-
control" id="InputForEmail" value=<?= set_value('email') ?>">
        </div>
        <div class="mb-3">
            <label for="InputForPassword" class="form-
label">Password</label>
            <input type="password" name="password" class="for-
m-control" id="InputForPassword">
        </div>
        <button type="submit" class="btn btn-
primary">Login</button>
        <?= form_close(); ?>
    </div>
</div>
<?= $this->endSection() ?>
```

Jika pengguna memasukkan data yang salah, maka error akan ditampilkan di halaman login, sebaliknya jika data sesuai maka halaman akan dialihkan ke halaman dashboard. Sekarang, kita buat halaman dashboard.php:

```

<?= $this->extend('base') ?>
<?= $this->section('content') ?>
<?php
$session = session();
?>
<div class="container mt-5">
    <div class="row">
        <div class="col-md-6">
            <div class="card">
                <div class="card-header">
```

```

        <?= $title ?>
    </div>
    <div class="card-body">
        Welcome <?=$session-
>get('user_name');?> | <a href="= base_url('logout') ?&gt;"&gt; Logout &lt;/a&gt;
    &lt;/div&gt;
&lt;/div&gt;
&lt;/div&gt;
&lt;?= $this-&gt;endSection() ?&gt;</pre

```

### 6.5.3. Membuat file Filters Auth.php

CodeIgniter 4 menyediakan fitur Filter yang berfungsi untuk menangani Before Request ataupun After Request. Fitur ini sangat bermanfaat untuk melakukan validasi untuk setiap request yang ditulis dengan kode yang sama. Pada kasus ini, kita akan memproteksi fungsi index() pada controller DashboardController.php dari pengguna yang belum login menggunakan Filter.

Sekarang, buatlah sebuah file Filter bernama "Auth.php" pada folder "app/Filters", kemudian ketikan kode berikut:

```

<?php namespace App\Filters;

use CodeIgniter\HTTP\RequestInterface;
use CodeIgniter\HTTP\ResponseInterface;
use CodeIgniter\Filters\FilterInterface;

class Auth implements FilterInterface
{
    public function before(RequestInterface $request, $arguments = null)
    {
        // jika user belum login
        if(! session()->get('logged_in')){
            // maka redirect ke halaman login
            return redirect()->to('/login');
        }
    }

    //-----

    public function after(RequestInterface $request, ResponseInterface $response,
$arguments = null)
    {
        // Do something here
    }
}
```

```

    }
}
```

Pada Filter "Auth.php" diatas, terdapat 2 function yaitu: fungsi before() dan fungsi after(). Pada kasus ini, kita hanya bermain di function before(). Fungsi before digunakan untuk melakukan validasi request sebelum suatu method controller dijalankan. Pada filter "Auth.php" di atas, kita mengalihkan pengguna ke halaman login jika mengakses suatu halaman sebelum login terlebih dahulu. Sedangkan fungsi after() dieksekusi setelah suatu method controller dijalankan.

Selanjutnya buka file "Filters.php" yang terdapat pada Folder "app\Config", kemudian ubah kode berikut:

```
public $aliases = [
    'csrf'      => CSRF::class,
    'toolbar'   => DebugToolbar::class,
    'honeypot'  => Honeypot::class,
];
menjadi
```

```
public $aliases = [
    'csrf'      => CSRF::class,
    'toolbar'   => DebugToolbar::class,
    'honeypot'  => Honeypot::class,
    'auth'       => \App\Filters\Auth::class,
];
```

Pada kode diatas, kita menambahkan satu baris kode tambahan yaitu "auth" yang berfungsi untuk memanggil kelas Auth dalam folder App\Filters.

#### 6.5.4. Membuat Routes Login dan Dashboard

Buka file Routes.php folder app\Config kemudian tambahkan routing berikut:

```
$routes->get('login', 'LoginController::index');
$routes->post('login/auth', 'LoginController::auth');
$routes->get('logout', 'LoginController::logout');
$routes->get('dashboard', 'DashboardController::index',['filter' => 'auth']);
```

Routes pertama untuk memanggil fungsi login di controller LoginController. Sekarang kita buka halaman login pada <http://localhost:8081/login>

The screenshot shows a 'Login Form' window. At the top, it says 'Sign In'. Below that is a label 'Email address' followed by an empty input field. Then there is a label 'Password' followed by another empty input field. At the bottom is a blue 'Login' button.

Gambar 6.6. Tampilan form login

Routes login/auth memanggil fungsi auth yang digunakan untuk melakukan otentikasi pengguna yang ingin masuk ke dalam dashboard. Misal, kita coba memasukkan data yang kosong atau email yang salah. Hasilnya seperti berikut:

The screenshot shows a 'Login Form' window. At the top, it says 'Sign In'. Below that is a red box containing the text 'Email not Found'. Then there is a label 'Email address' followed by an empty input field. Then there is a label 'Password' followed by another empty input field. At the bottom is a blue 'Login' button.

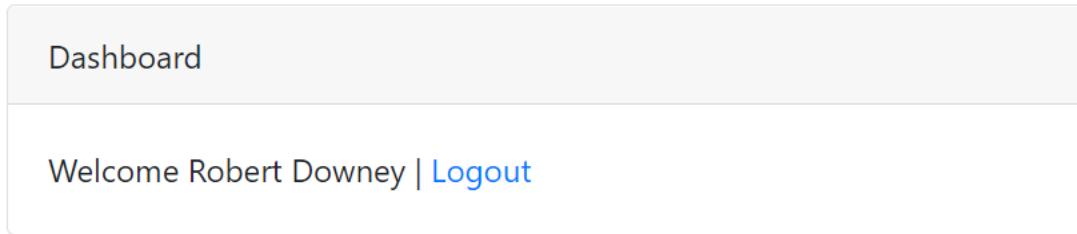
Gambar 6.7. Tampilan validasi form login

Kemudian kita coba memasukan email yang benar dengan password yang salah. Hasilnya seperti ini:

The screenshot shows a 'Login Form' with a title 'Sign In'. Below the title, there is a red rectangular box containing the text 'Wrong Password'. Below this box are two input fields: one for 'Email address' and one for 'Password', both of which are currently empty. At the bottom of the form is a blue rectangular button labeled 'Login'.

Gambar 6.8. Tampilan validasi form login kedua

Jika kita memasukkan data yang valid, maka halaman akan dialihkan ke halaman dashboard:



Gambar 6.9. Tampilan halaman dashboard

Halaman dashboard ini kita proteksi menggunakan filter auth. Jika memiliki halaman lain yang ingin diproteksi, maka tinggal tambahkan di route dan tambahkan juga filter "auth"-nya, maka halaman tersebut tidak dapat diakses sebelum login tanpa harus membuat file Filter lagi. Klik link Logout untuk menghapus sesi dan kembali ke halaman Login.

### Latihan

Buatlah halaman register dan halaman login. Jika user berhasil login, maka halaman dialihkan ke halaman dashboard yang berisi info user. Dalam halaman dashboard terdapat menu untuk logout kembali ke halaman login

# BAB 7

# MEMBUAT REST API

## Tujuan instruksional:

- Mampu membuat RESTful API untuk setiap EndPoint CRUD

Representational State Transfer (REST) adalah gaya arsitektur untuk aplikasi terdistribusi. Biasanya digunakan untuk membuat aplikasi interaktif yang menggunakan Web Services. Web Service yang mengikuti pedoman ini disebut RESTful. Web Service semacam itu harus menyediakan sumber daya Web-nya dalam representasi tekstual dan memungkinkan untuk dibaca dan dimodifikasi dengan protokol stateless dan serangkaian operasi yang telah ditentukan. Pendekatan ini memungkinkan terjadinya interoperabilitas antar sistem komputer di Internet yang menyediakan layanan ini. Metode ini sering diterapkan dalam pengembangan aplikasi.

RESTful API merupakan implementasi dari API (Application Programming Interface). Tujuannya adalah untuk menjadikan sistem dengan performa yang baik, cepat, dan mudah untuk dikembangkan terutama dalam pertukaran dan komunikasi data.

## 7.1. Pengenalan API

API adalah kumpulan kode program untuk komunikasi antar perangkat lunak satu dengan lainnya dengan menggunakan data. Ada 2 konsep penting yang perlu diperhatikan:

### 1. Client

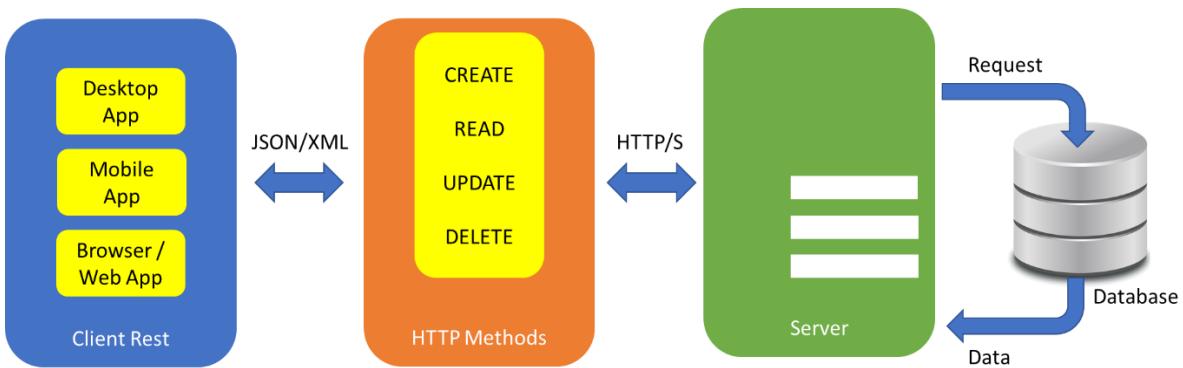
Client adalah pengguna dari API. Client bisa developer, misal kita menggunakan facebook API sebagai developer untuk membaca dan menulis baris data dari facebook, membuat post baru dan melakukan aktifitas lainnya dalam program yang kita kembangkan. Client juga bisa berupa web browser, jika kita masuk ke halaman website facebook, browser kita bertindak sebagai client dan memanggil API dari facebook dan menggunakan datanya untuk ditampilkan ke layar.

### 2. Resource

Setiap objek API dapat menyediakan sumber daya data. Misalnya resource atau sumber daya dari API Twitter yang berupa pengguna dan gambar. Setiap resource memiliki identifikasi yang unik. Identifikasi ini bisa berupa nama atau angka.

## 7.2. Pengenalan REST

RESTful API merupakan representasi state dari resource yang diminta kemudian ditransfer dari server ke pengguna. Protokol yang digunakan adalah protokol HTTP yang stateless dan tidak menyimpan data yang diminta. Dengan REST API, kita submit permintaan untuk operasi CREATE, READ, UPDATE, dan DELETE menggunakan protokol ini dan menerima responnya. Hasil respon ini dapat berupa XML atau JSON.



Gambar 7.1. Arsitektur REST API

Ketika salah satu API dalam server dipanggil bergantung pada 2 hal yaitu:

1. Pengenal sumber daya. Ini adalah URL sumber daya yang juga dikenal sebagai endpoint
2. Operasi yang ingin server lakukan pada sumber daya itu berdasarkan dari metode HTTP yang digunakan. Dengan mengikuti metode HTTP tersebut, server akan melakukan operasi yang diinginkan, Metode itu antara lain:
  - a. GET untuk mendapatkan data dari server atau lebih dikenal dengan istilah READ,
  - b. POST untuk meng-CREATE data baru,
  - c. PUT untuk UPDATE data, dan
  - d. DELETE untuk menghapus data.

### 7.3. Design EndPoint RESTful API

Langkah awal sebelum membuat RESTful API adalah kita perlu mendefinisikan EndPoint dari RESTful API yang akan dibuat. EndPoint merupakan routes dari API yang akan dibuat. Pada contoh pembuatan RESTful API ini kita akan menggunakan tabel baru bernama subject. Kemudian kita buat kode RESTful API sederhana untuk mengambil data dari server (GET), membuat data baru ke server (POST), mengupdate data ke server (PUT), dan menghapus data ke server (DELETE) dari tabel di database. Berikut rancangan dari RESTful API yang akan kita buat:

Tabel 7.1 Tabel EndPoint

Method	EndPoint	Deskripsi
GET	/subject	Menampilkan semua data subject
GET	/subject/{id}	Menampilkan data subject tertentu
POST	/subject	Memasukkan data subject baru
PUT	/subject/{id}	Mengubah data subject
DELETE	/subject/{id}	Menghapus data subject

## 7.4. Membuat Tabel Subject

Bukalah command prompt atau terminal. Masuk ke direktori projek kemudian buat tabel subject menggunakan perintah berikut:

```
D:\xampp\htdocs\myproject>php spark migrate:create subject
CodeIgniter v4.1.1 Command Line Tool - Server Time: 2021-05-03 03:37:06 UTC-05:00
File created: APPPATH\Database\Migrations\2021-05-03-083707_Subject.php
```

Selanjutnya akan terbentuk file baru dengan nama 2021-XXXX\_Subject.php pada folder App/Database/Migrations. Buka file tersebut kemudian edit menjadi seperti berikut:

```
<?php

namespace App\Database\Migrations;

use CodeIgniter\Database\Migration;

class Subject extends Migration
{
    public function up() {
        $this->forge->addField([
            'subject_id' => [
                'type' => 'INT',
                'unsigned' => TRUE,
                'auto_increment' => TRUE
            ],
            'subject_name' => [
                'type' => 'VARCHAR',
                'constraint' => 100,
                'null' => FALSE
            ],
            'skls' => [
                'type' => 'INT',
                'null' => FALSE
            ],
            'created_at' => [
                'type' => 'datetime',
                'null' => TRUE
            ],
            'updated_at' => [
                'type' => 'datetime',
                'null' => TRUE
            ]
        ]);
    }

    public function down() {
        $this->forge->dropTable('subject');
    }
}
```

```

    ]);

    $this->forge->addKey('subject_id', TRUE);
    $this->forge->createTable('subject');
}

public function down() {
    $this->forge->dropTable('subject');
}
}

```

Kemudian dari command prompt jalankan perintah berikut untuk membuat tabel subject:

```
D:\xampp\htdocs\myproject>php spark migrate
```

```
CodeIgniter v4.1.1 Command Line Tool - Server Time: 2021-05-03 03:38:36 UTC-05:00

Running all new migrations...
    Running: (App) 2021-05-03-083707_App\Database\Migrations\Subject
Done migrations.
```

Kita bisa melihat hasilnya dengan membuka database di perangkat yang ada, misal kita menggunakan phpmyadmin. Jika kita klik tabel pegawai, maka akan terlihat struktur tabel sesuai dengan yang telah didefinisikan.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	subject_id	int(10)		UNSIGNED	No	None		AUTO_INCREMENT	Change  Drop  More
2	subject_name	varchar(100)	utf8_general_ci		No	None			Change  Drop  More
3	sks	int(11)			No	None			Change  Drop  More
4	created_at	datetime			Yes	NULL			Change  Drop  More
5	updated_at	datetime			Yes	NULL			Change  Drop  More

Gambar 7.2. Tabel Subject di Database

## 7.5. Memasukkan Model Subject

Berikutnya kita membuat model untuk tabel subject yang telah dibuat menggunakan composer.

```
D:\xampp\htdocs\myproject>php spark make:model Subject

CodeIgniter v4.1.1 Command Line Tool - Server Time: 2021-05-03 03:39:10 UTC-05:00

File created: APPPATH\Models\Subject.php
```

Lalu, isikan kode dalam model Subject seperti di bawah ini:

```
<?php

namespace App\Models;

use CodeIgniter\Model;

class Subject extends Model
{
    protected $table = 'subject';
    protected $primaryKey = 'subject_id';
    protected $allowedFields = ['subject_name', 'sks'];
    protected $useTimestamps = true;
    protected $createdField = 'created_at';
    protected $updatedField = 'updated_at';
}
```

## 7.6. Memasukkan Data ke Dalam Tabel Subject

Untuk contoh menampilkan data, kita perlu memasukkan data terlebih dahulu ke dalam tabel. Kita bisa memasukkan dengan cara manual menggunakan phpmyadmin atau menggunakan cara yang telah disediakan oleh codeigniter4 yaitu menggunakan seeder. Kita akan pakai cara kedua, jadi buatlah file baru dengan nama SubjectTableSeeder.php pada folder app\Database\Seeds. Kemudian ketikkan script berikut:

```
<?php

namespace App\Database\Seeds;

class SubjectTableSeeder extends \CodeIgniter\Database\Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
```

```

$data = [
    [
        'subject_name' => 'Object Oriented Programming',
        'sks' => '2',
        'created_at' => date('Y-m-d H:i:s'),
        'updated_at' => date('Y-m-d H:i:s')
    ],
    [
        'subject_name' => 'Web Programming',
        'sks' => '3',
        'created_at' => date('Y-m-d H:i:s'),
        'updated_at' => date('Y-m-d H:i:s')
    ],
    [
        'subject_name' => 'Information Security System',
        'sks' => '2',
        'created_at' => date('Y-m-d H:i:s'),
        'updated_at' => date('Y-m-d H:i:s')
    ],
    [
        'subject_name' => 'Algorithm and Data Structure',
        'sks' => '3',
        'created_at' => date('Y-m-d H:i:s'),
        'updated_at' => date('Y-m-d H:i:s')
    ],
    [
        'subject_name' => 'Human Computer Interaction',
        'sks' => '3',
        'created_at' => date('Y-m-d H:i:s'),
        'updated_at' => date('Y-m-d H:i:s')
    ]
];
$this->db->table('subject')->insertBatch($data);
}
}

```

Simpan, Kemudian dari command prompt jalankan perintah berikut untuk menambahkan data subject:

```

D:\xampp\htdocs\myproject>php spark db:seed SubjectTableSeeder

CodeIgniter v4.1.1 Command Line Tool - Server Time: 2021-05-03 03:40:56 UTC-05:00

Seeded: App\Database\Seeds\SubjectTableSeeder

```

Sekarang kita bisa melihat hasilnya dalam database seperti gambar di bawah ini:

subject_id	subject_name	skls
1	Object Oriented Programming	2
2	Web Programming	3
3	Information Security System	2
4	Algorithm and Data Structure	3
5	Human Computer Interaction	3

Gambar 7.3. Data dalam tabel subject

## 7.7. Membuat Controller SubjectController

Buat sebuah controller baru bernama "SubjectController.php" pada folder "app/Controllers", kemudian ketikan kode berikut:

```
<?php

namespace App\Controllers;

use App\Controllers\BaseController;
use CodeIgniter\RESTful\ResourceController;
use CodeIgniter\API\ResponseTrait;
use App\Models\Subject;

class SubjectController extends ResourceController
{
    use ResponseTrait;

    //menampilkan semua data subject
    public function index()
    {
        $model = new Subject();
        $data = $model->findAll();
        return $this->respond($data, 200);
    }

    // mendapatkan data tertentu
    public function show($id = null)
    {
        $model = new Subject();
        $data = $model->getWhere(['subject_id' => $id])->getResult();
        if($data){
            return $this->respond($data);
        }
    }
}
```

```

    }else{
        return $this->failNotFound('No Data Found with id '.$id);
    }
}

// create a subject
public function create()
{
    $model = new Subject();
    $data = [
        'subject_name' => $this->request->getPost('subject_name'),
        'sks' => $this->request->getPost('sks')
    ];

    $model->insert($data);
    $response = [
        'status' => 201,
        'error' => null,
        'messages' => [
            'success' => 'Data Saved'
        ]
    ];
}

return $this->respondCreated($response);
}

// update subject
public function update($id = null)
{
    $model = new Subject();
    $json = $this->request->getJSON();
    if($json){
        $data = [
            'subject_name' => $json->subject_name,
            'sks' => $json->SKS
        ];
    }else{
        $input = $this->request->getRawInput();
        $data = [
            'subject_name' => $input['subject_name'],
            'sks' => $input['sks']
        ];
    }
}

```

```

// update ke Database
$model->update($id, $data);
$response = [
    'status' => 200,
    'error' => null,
    'messages' => [
        'success' => 'Data Updated'
    ]
];
return $this->respond($response);
}

// delete subject
public function delete($id = null)
{
    $model = new Subject();
    $data = $model->find($id);
    if($data){
        $model->delete($id);
        $response = [
            'status' => 200,
            'error' => null,
            'messages' => [
                'success' => 'Data Deleted'
            ]
        ];
        return $this->respondDeleted($response);
    }else{
        return $this->failNotFound('No Data Found with id '.$id);
    }
}

```

CodeIgniter 4 telah memberikan kemudahan bagi web developer dalam membuat RESTful API. Dapat dilihat pada controller SubjectController.php diatas, dengan hanya mengextends ResourceController kita telah dapat membuat RESTful API. ResourceController menyediakan titik awal untuk RESTful API dengan metode yang sesuai dengan route resource. Kita juga bisa dengan mudah membuat response dengan menggunakan API ResponseTrait. CodeIgniter menyediakan API Response Trait yang bisa digunakan dengan controller apapun untuk membuat response menjadi sederhana tanpa perlu mengingat kode status HTTP yang harus dikembalikan.

## 7.7. Konfigurasi Routes.php

Langkah berikutnya adalah melakukan konfigurasi pada file Routes.php yang terdapat pada folder "app/Config". Buka file "Routes.php" pada folder "app/Config", kemudian tambahkan kode berikut:

```
$routes->get('subject', 'SubjectController::index');
$routes->post('subject', 'SubjectController::create');
$routes->get('subject/(:num)', 'SubjectController::show/$1');
$routes->put('subject/(:num)', 'SubjectController::update/$1');
$routes->delete('subject/(:num)', 'SubjectController::delete/$1');
```

Keterangan:

- Route pertama untuk menampilkan semua data subject
- Route kedua untuk memasukkan data subject baru
- Route ketiga untuk menampilkan data subject tertentu
- Route keempat untuk mengubah data subject tertentu
- Route kelima untuk menghapus data subject tertentu

## 7.8. Mengaktifkan Cross-Origin Resource Sharing (CORS)

Cross-Origin Resource Sharing (CORS) adalah mekanisme berbasis header HTTP yang memungkinkan server untuk menunjukkan asal lain (domain, skema, atau port) selain miliknya yang mana browser harus mengizinkan pemenuhan sumber daya. CORS juga mengandalkan mekanisme yang digunakan browser untuk membuat permintaan "preflight" ke server yang menghosting sumber daya lintas asal, untuk memeriksa apakah server akan mengizinkan permintaan yang sebenarnya. Dalam preflight tersebut, browser mengirimkan header yang menunjukkan metode HTTP dan header yang akan digunakan dalam permintaan sebenarnya. Agar resources dapat diakses di luar domain, kita perlu mengaktifkan CORS. Untuk menaktifkan CORS, buat file bernama "Cors.php" pada folder "app/Filters". Kemudian ketikan kode berikut:

```
<?php namespace App\Filters;

use CodeIgniter\HTTP\RequestInterface;
use CodeIgniter\HTTP\ResponseInterface;
use CodeIgniter\Filters\FilterInterface;

Class Cors implements FilterInterface
{
    public function before(RequestInterface $request, $arguments = null)
    {
        header('Access-Control-Allow-Origin: *');
```

```

        header("Access-Control-Allow-Headers: X-API-KEY, Origin, X-Requested-With, Content-Type, Accept, Access-Control-Request-Method, Authorization");
        header("Access-Control-Allow-Methods: GET, POST, OPTIONS, PUT, DELETE");
        $method = $_SERVER['REQUEST_METHOD'];
        if ($method == "OPTIONS") {
            die();
        }
    }

    public function after(RequestInterface $request, ResponseInterface $response,
    $arguments = null)
    {
        // Do something here
    }
}

```

Selanjutnya buka file "Filters.php" yang terdapat pada folder "app/Config". Kemudian temukan kode berikut:

```

public $aliases = [
    'csrf'      => CSRF::class,
    'toolbar'   => DebugToolbar::class,
    'honeypot'  => Honeypot::class,
    'auth'       => \App\Filters\Auth::class,
];

```

Ubah menjadi

```

public $aliases = [
    'csrf'      => CSRF::class,
    'toolbar'   => DebugToolbar::class,
    'honeypot'  => Honeypot::class,
    'auth'       => \App\Filters\Auth::class,
    'cors'       => \App\Filters\Cors::class,
];

```

Selanjutnya definisikan "cors" pada public globals seperti berikut:

```

public $globals = [
    'before' => [
        // 'honeypot',
        // 'csrf',
        'cors',
    ],
]

```

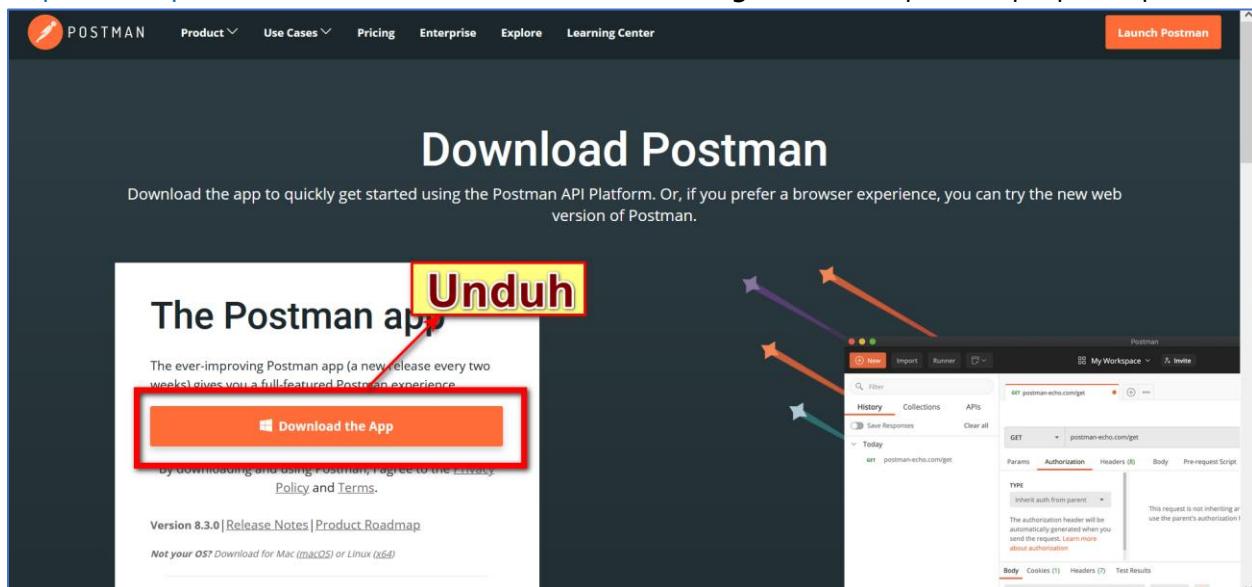
```
'after' => [
  'toolbar',
  // 'honeypot',
],
];
```

## 7.9. Pengujian RESTful API Menggunakan Postman

Postman adalah platform kolaborasi untuk pengembangan API. Fitur Postman menyederhanakan setiap langkah pembuatan API dan menyederhanakan kolaborasi sehingga kita dapat membuat API yang lebih baik dan lebih cepat. Postman berfungsi sebagai REST CLIENT untuk uji coba REST API. Postman biasa digunakan oleh developer pembuat API sebagai alat untuk menguji API yang telah dibuat.

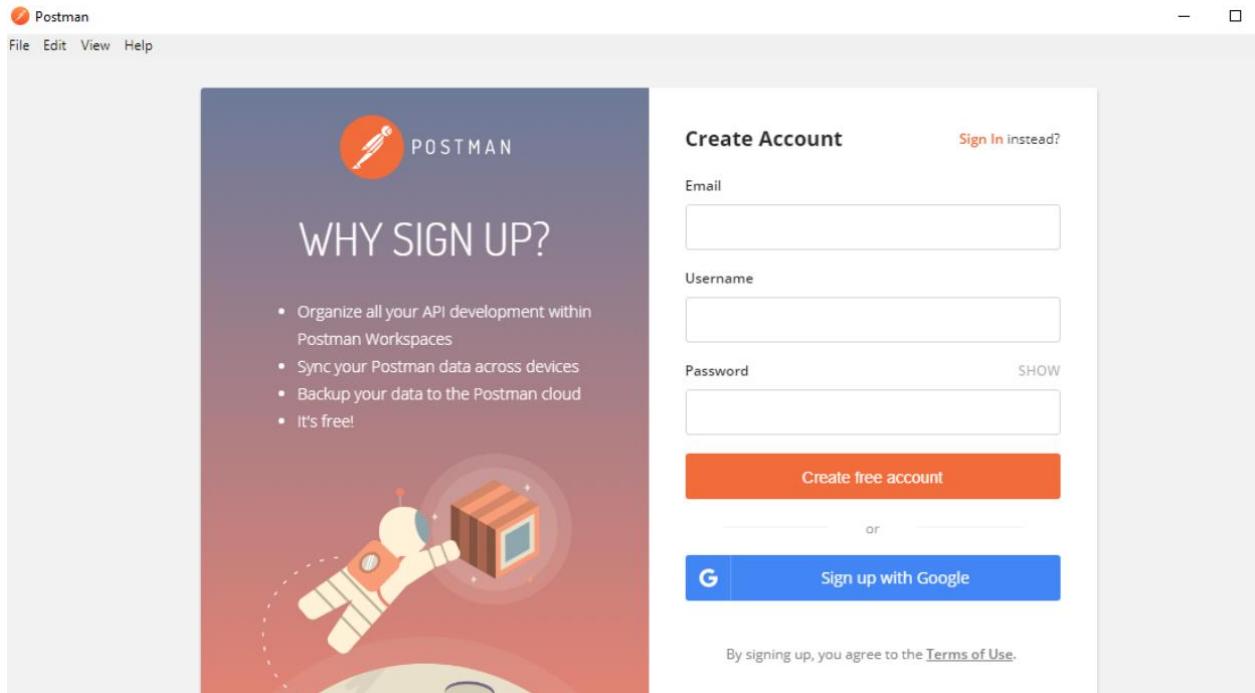
### 7.9.1. Instalasi Postman

Hal yang harus dilakukan pertama yaitu Unduh Postman di halaman <https://www.postman.com/downloads/>. Pilih sesuai dengan Sistem Operasi laptop/komputer.



Gambar 7.4. Halaman download Postman

Setelah selesai mengunduh, jalankan setup instalasi aplikasi Postman. Setup akan menjalankan proses instalasi aplikasi Postman secara otomatis. Setelah proses instalasi selesai, kita akan diarahkan untuk membuat akun Postman dan sign in menggunakan akun tersebut. Kita juga dapat melakukan sign in menggunakan akun Google.



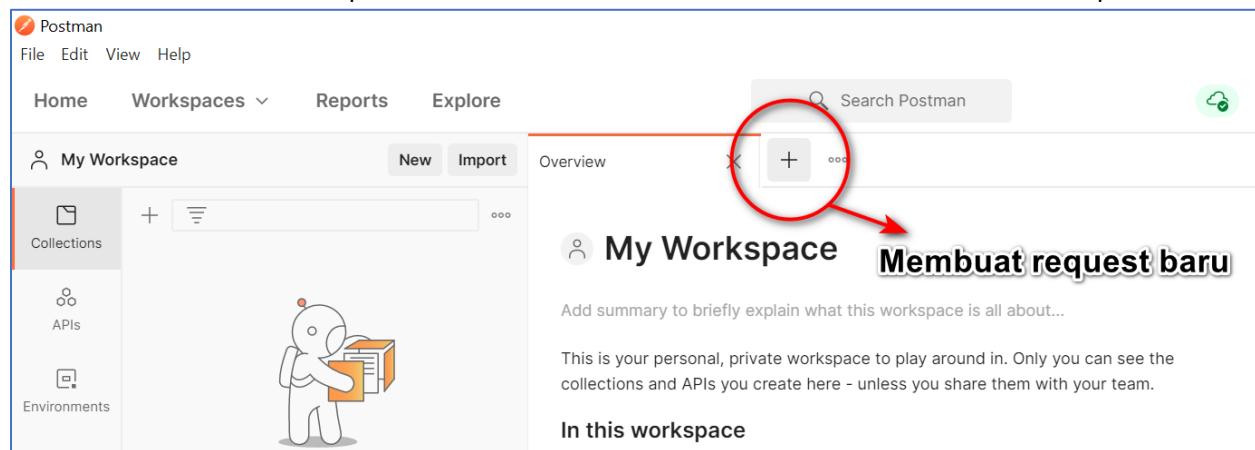
Gambar 7.5. Halaman Sign Up

### 7.9.2. Pengujian

Jalankan project dengan mengetikkan perintah berikut pada Terminal / Command Prompt:

```
Administrator: Command Prompt - php spark serve
D:\xampp\htdocs\myproject>php spark serve
```

Setelah itu buka kembali aplikasi Postman dan tekan tombol "+" untuk membuat request baru:



Gambar 7.6. Membuat Request Baru

Kemudian kita coba setiap EndPoint dalam REST yang telah dibuat:

### 1. Menampilkan semua data

Masukkan URL <http://localhost:8081/subject> pada kolom URL di Postman. Pastikan method yang digunakan adalah GET dan tekan tombol Send.

Hasil responsenya sebagai berikut:

```

1
2 {
3     "subject_id": "1",
4     "subject_name": "Object Oriented Programming",
5     "skls": "2",
6     "created_at": "2021-05-03 03:40:56",
7     "updated_at": "2021-05-03 03:40:56"
8 },
9 {
10    "subject_id": "2",
11    "subject_name": "Web Programming"

```

Jika kita scroll, maka akan terlihat semua data subject.

### 2. Menampilkan data tertentu

Masukkan URL <http://localhost:8081/subject/3> pada kolom URL di Postman. Pastikan method yang digunakan adalah GET dan tekan tombol Send.

Pada contoh di atas, kita mengambil subject yang memiliki subject\_id = 3. Hasil responsenya sebagai berikut:

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```

1
2   {
3     "subject_id": "3",
4     "subject_name": "Information Security System",
5     "sks": "2",
6     "created_at": "2021-05-03 03:40:56",
7     "updated_at": "2021-05-03 03:40:56"
8   }
9

```

### 3. Membuat data baru

Masukkan URL <http://localhost:8081/subject> pada kolom URL di Postman. Pastikan method yang digunakan adalah POST. Kemudian pilih tab Body dan klik pada bagian x-www-form-urlencoded lalu kita masukkan nilai untuk subject\_name dan sks di dalamnya seperti gambar di bawah.

POST  Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	subject_name	Basis Data			
<input checked="" type="checkbox"/>	sks	2			
	Key	Value	Description		

Tekan tombol Send

Hasil response seperti berikut:

Body Cookies Headers (11) Test Results 201 Created 110 ms 649 B

Pretty Raw Preview Visualize JSON

```

1
2   {
3     "status": 201,
4     "error": null,
5     "messages": {
6       "success": "Data Saved"
7     }
8

```

Jika kita cek di dalam databasenya, terlihat data baru telah masuk

subject_id	subject_name	sks
1	Object Oriented Programming	2
2	Web Programming	3
3	Information Security System	2
4	Algorithm and Data Structure	3
5	Human Computer Interaction	3
8	Basis Data	2

#### 4. Mengubah Data

Masukkan URL <http://localhost:8081/subject/8> sebagai contoh pada kolom URL di Postman. Pastikan method yang digunakan adalah PUT. Kemudian pilih tab Body dan klik pada bagian x-www-form-urlencoded lalu kita masukkan nilai untuk subject\_name dan sks di dalamnya seperti gambar di bawah.

KEY	VALUE	DESCRIPTION	...	Bulk Edit
subject_name	Praktikum Basis Data			
sks	3			
Key	Value	Description		

Tekan tombol Send

Hasil response seperti berikut:

```

1
2 "status": 200,
3 "error": null,
4 "messages": {
5     "success": "Data Updated"
6 }

```

Jika kita cek di dalam databasenya, terlihat data baru telah berubah

subject_id	subject_name	sks
1	Object Oriented Programming	2
2	Web Programming	3
3	Information Security System	2
4	Algorithm and Data Structure	3
5	Human Computer Interaction	3
8	Praktikum Basis Data	3

## 5. Menghapus Data

Masukkan URL <http://localhost:8081/subject/8> sebagai contoh pada kolom URL di Postman. Pastikan method yang digunakan adalah DELETE. Kemudian tekan tombol Send.

The screenshot shows the Postman interface. In the top bar, 'DELETE' is selected from a dropdown, and the URL 'http://localhost:8081/subject/8' is entered. On the right, a blue 'Send' button is visible. Below the URL input, tabs for 'Params', 'Authorization', 'Headers (6)', 'Body', 'Pre-request Script', 'Tests', 'Settings', and 'Cookies' are shown. The 'Headers' tab is active. At the bottom, the response section displays a JSON object:

```

1   "status": 200,
2   "error": null,
3   "messages": {
4     "success": "Data Deleted"
5   }
    
```

On the right side of the response panel, there are icons for copy, search, and refresh. Above the response, status information is shown: 200 OK, 112 ms, 646 B, and a 'Save Response' dropdown.

Jika kita cek di dalam databasenya, terlihat data baru telah terhapus

subject_id	subject_name	sks
1	Object Oriented Programming	2
2	Web Programming	3
3	Information Security System	2
4	Algorithm and Data Structure	3
5	Human Computer Interaction	3

### **Latihan**

Buatlah RESTful API untuk data berikut:

- a. Nama
- b. Tempat dan Tanggal Lahir
- c. Alamat
- d. No Telepon
- e. Jenis Kelamin
- f. Pendidikan

## DAFTAR PUSTAKA

Anonim. 2015. *Code Igniter*. Tutorials Point (1) Pvt.LTd.  
CodeIgniter. "CodeIgniter Documentation", <https://codeigniter.com/docs> [1 Mei 2021].