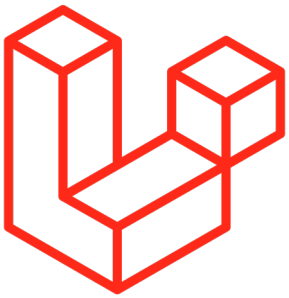
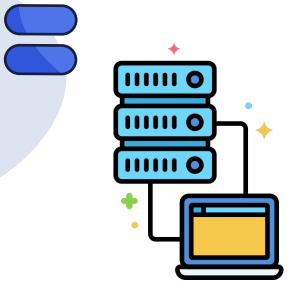
SMK Telkom Makassar

SERVER SIDE

FRAMEWORK





FRAMEWORK?

Framework bahasa pemrograman adalah sebuah kerangka kerja atau struktur yang dirancang untuk mempermudah pengembangan aplikasi dengan menyediakan sekumpulan alat, pustaka (library), dan panduan. Framework ini membantu pengembang dalam mengurangi upaya pengkodean manual dan mengatasi tugas-tugas umum yang sering dihadapi dalam pengembangan perangkat lunak.

Berikut adalah beberapa poin penting tentang framework bahasa pemrograman:

Struktur Terorganisir: Framework menyediakan struktur yang sudah terorganisir untuk kode Anda, sehingga Anda tidak perlu memulai dari nol. Struktur ini memandu pengembang dalam membangun aplikasi dengan cara yang konsisten dan terstandarisasi.

Komponen Siap Pakai: Framework biasanya dilengkapi dengan berbagai komponen siap pakai, seperti pengelolaan database, sistem autentikasi, manajemen sesi, dan lainnya, yang dapat digunakan langsung tanpa harus menulis ulang kode dari awal.

Penghematan Waktu: Dengan menggunakan framework, pengembang dapat menghemat waktu karena banyak fungsi dasar yang sudah disediakan dan hanya perlu diintegrasikan ke dalam aplikasi.

Keamanan: Banyak framework menyediakan fitur keamanan yang sudah teruji, seperti perlindungan terhadap serangan SQL Injection, Cross-Site Scripting (XSS), dan lainnya. Ini membantu menjaga keamanan aplikasi tanpa perlu memikirkan semua detailnya secara manual.

Dukungan Komunitas: Framework biasanya didukung oleh komunitas besar pengembang, yang berarti ada banyak dokumentasi, tutorial, dan sumber daya lain yang tersedia untuk membantu pengembangan dan pemecahan masalah.

Beberapa framework populer untuk pengembangan web termasuk Django (Python), Ruby on Rails (Ruby), Laravel (PHP), dan Spring (Java).
React Native (JavaScript), Flutter (Dart), dan Xamarin (C#) adalah contoh framework untuk pengembangan aplikasi mobile.
Angular, React, dan Vue.js adalah framework untuk pengembangan frontend aplikasi web

LARAVEL

Laravel adalah salah satu framework PHP yang paling populer dan digunakan untuk mengembangkan aplikasi web. Framework ini dirancang dengan tujuan untuk memudahkan pengembangan web yang elegan, efisien, dan terstruktur. Laravel mengikuti pola arsitektur MVC (Model-View-Controller), yang memisahkan logika bisnis, tampilan, dan kontrol alur aplikasi.

Fitur Utama Laravel:

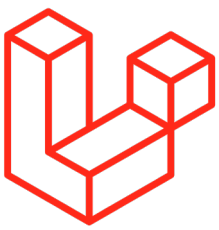
Routing yang Mudah: Laravel menyediakan sistem routing yang sangat sederhana dan fleksibel. Ini memungkinkan pengembang untuk mendefinisikan rute URL dengan mudah dan menghubungkannya dengan logika bisnis dalam aplikasi.

Blade Templating Engine: Laravel dilengkapi dengan Blade, mesin template bawaan yang memungkinkan pengembang untuk membuat tampilan dinamis dengan menggunakan sintaksis yang sederhana dan efisien.

Eloquent ORM (Object-Relational Mapping): Laravel menggunakan Eloquent ORM yang memudahkan interaksi dengan database. Eloquent memungkinkan pengembang untuk bekerja dengan database menggunakan objek PHP, sehingga mengurangi kebutuhan untuk menulis query SQL secara manual.

Artisan Console: Laravel memiliki alat command-line bernama Artisan yang membantu pengembang dalam berbagai tugas pengembangan, seperti migrasi database, seeding data, pembuatan controller, dan lainnya.

Migrations: Fitur ini memungkinkan pengelolaan skema database dengan mudah. Pengembang bisa melacak perubahan pada database dengan menggunakan migrasi, yang memungkinkan pengelolaan versi database yang lebih baik.



MVC

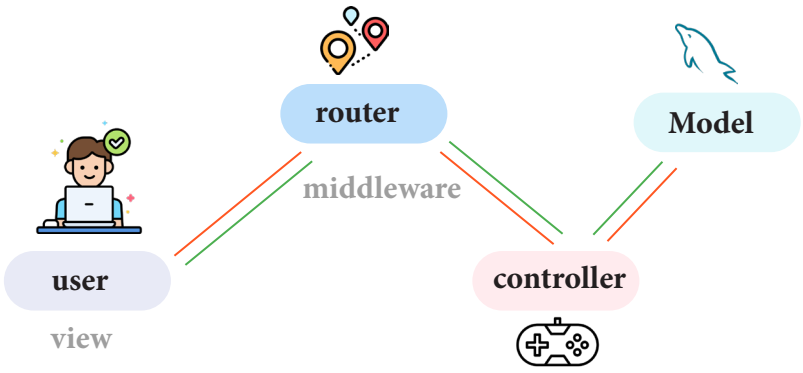
MVC (Model-View-Controller) adalah sebuah pola arsitektur perangkat lunak yang memisahkan aplikasi menjadi tiga komponen utama: Model, View, dan Controller. Pola ini digunakan untuk mengorganisir kode agar lebih terstruktur, terpisah dengan jelas, dan mudah dikelola

Model:
Model merupakan bagian dari aplikasi yang berhubungan langsung dengan data dan logika bisnis. Model mengelola data aplikasi, menghubungkan dengan database, dan mengimplementasikan aturan serta logika bisnis yang diperlukan.

View:
View adalah komponen yang bertanggung jawab untuk menampilkan data kepada pengguna. Ini adalah representasi visual dari data yang disediakan oleh Model. View menangani semua aspek antarmuka pengguna (UI).

Controller:
Controller bertindak sebagai penghubung antara Model dan View. Controller menerima input dari pengguna, memprosesnya, dan menentukan apa yang harus dilakukan dengan data tersebut. Setelah memproses input, Controller memperbarui Model dan memilih View yang sesuai untuk ditampilkan

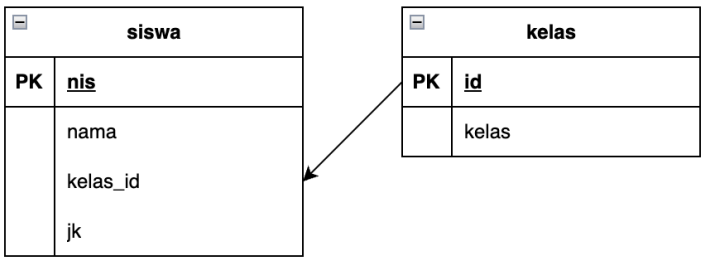
Middleware: Laravel mendukung penggunaan middleware, yang merupakan lapisan pengolahan tambahan yang dapat digunakan untuk memeriksa atau memodifikasi permintaan HTTP sebelum mencapai controller. Ini berguna untuk fungsi seperti autentikasi, logging, dan lainnya.
Autentikasi dan Otorisasi: Laravel menyediakan sistem autentikasi dan otorisasi yang lengkap, yang dapat diimplementasikan dengan sangat mudah. Ini termasuk manajemen pengguna, login, registrasi, reset password, dan otorisasi berbasis peran.
Queuing System: Laravel memiliki sistem antrian bawaan yang memungkinkan penundaan tugas seperti pengiriman email atau pemrosesan tugas berat lainnya, sehingga aplikasi dapat merespons permintaan dengan cepat.
API Development: Laravel memudahkan pembuatan API, dengan dukungan penuh untuk pengiriman JSON, verifikasi API token, dan lainnya. Laravel juga mendukung pengembangan API RESTful.



Alur Kerja:
Pengguna Mengakses URL: Pengguna mengunjungi URL tertentu di aplikasi Anda.
Router Menerima Permintaan: Router di Laravel menerima permintaan tersebut dan memutuskan Controller dan metode mana yang harus dipanggil.
Controller Memproses Permintaan:
Controller mengambil alih, memproses permintaan, dan mungkin membutuhkan data dari database. Controller kemudian menggunakan Model untuk mengambil data yang diperlukan.
Model Mengambil Data: Model berinteraksi dengan database, mengambil data yang diperlukan, dan mengembalikannya ke Controller.
Controller Mengirim Data ke View: Controller kemudian meneruskan data tersebut ke View untuk ditampilkan kepada pengguna.
View Menampilkan Data: View menerima data dari Controller dan menampilkannya dalam bentuk yang dapat dilihat dan dimengerti oleh pengguna (misalnya, halaman HTML).

Latihan CRUD

CRUD merupakan singkatan dari Create, Read, Update, dan Delete. Keempat operasi ini merupakan inti dari interaksi dengan data dalam sebagian besar aplikasi perangkat lunak.

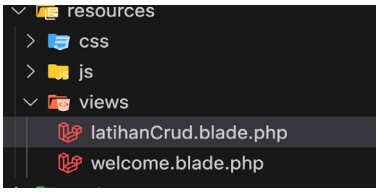


1. membuat proyek:
```
composer create-project laravel/laravel siswaApp
```

2. menghubungkan database dengan proyek applikasi dengan mengedit file .env:
```
...
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=
....
```
3. menjalankan artisan migrate pada terminal:
```
php artisan migrate
```

4. menjalankan service laravel
```
php artisan serve
```

5. membuat file **latihanCrud.blade.php** pada direktori resources/views



6. membuat controler dengan php artisan:
```
php artisan make:controller siswaController
```

7. mengedit file app/Http/Controllers/siswaController:

```php
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
class siswaController extends Controller
{
    public function index()
    {
        return view('latihanCrud');
    }
}
```

8. mengedit file routers/web.php:

```
<?php

use App\Http\Controllers\siswaController;
use Illuminate\Support\Facades\Route;

Route::get('/',[siswaController::class,'index']);
```

9. membuat layout tabel HTML pada file l**atihanCrud.blade.php** dengan hasil akhir sebagai berikut:

NIS	NAMA SISWA	KELAS	AKSI

10. untuk menampilkan data dari database edit file siswaController, pastikan data pada tb_siswa dan tb_kelas sudah terisi, untuk mengisi data silahkan menggunakan query mysql atau bisa menggunakan tool phpmyadmin:

```
MariaDB [db_siswa]> select * from tb_kelas;
+----+----------+
| id | kelas    |
+----+----------+
|  1 | XI RPL 1 |
|  2 | XI RPL 2 |
|  3 | XI RPL 3 |
|  4 | XI RPL 4 |
+----+----------+
```

```
MariaDB [db_siswa]> select * from tb_siswa;
+---------+------------+----------+-----------+
| nis     | nama       | kelas_id | jk        |
+---------+------------+----------+-----------+
| 5442311 | andi baso  |        4 | LAKI-LAKI |
| 5442312 | andi besse |        4 | PEREMPUAN |
| 5442313 | daeng kebo |        3 | LAKI-LAKI |
+---------+------------+----------+-----------+
```

siswaController:

```
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\View\View;
class siswaController extends Controller
{
    public function index():View
    {
        $data=DB::table('tb_siswa')
           ->leftJoin('tb_kelas','tb_siswa.kelas_id','=','tb_kelas.id')
           ->select('tb_siswa.*','tb_kelas.kelas')->get();
        return view('latihanCrud',[
            'data'=>$data
        ]);
    }
}
```

11. mengedit file latihanCrud.blade.php agar data dari database dapat tapil pada halaman aplikasi:

```
<tbody>
    @foreach ($data as $row )
        <tr>
           <td>{{ $row->nis }}</td>
           <td>{{ $row->nama }}</td>
           <td>{{ $row->kelas }}</td>
           <td>Edit / Hapus</td>
        </tr>
     @endforeach
</tbody>
```

NIS	NAMA SISWA	KELAS	AKSI
5442311	andi baso	XI RPL 4	Edit / Hapus
5442312	andi besse	XI RPL 4	Edit / Hapus
5442313	daeng kebo	XI RPL 3	Edit / Hapus

input data:

web.php:

```
<?php

use App\Http\Controllers\siswaController;
use Illuminate\Support\Facades\Route;

Route::get('/',[siswaController::class,'index']);
Route::get('/input-data',[siswaController::class,'input']);
Route::post('/proses-input',[siswaController::class,'proses']);
```

siswaController.php:

```
...
public function input():View
    {
        $kelas=DB::table('tb_kelas')->get();
        return view('input',[
            'kelas'=>$kelas
        ]);
    }
    public function proses(Request $request)
    {

        $request->validate([
            'nis'=>'required',
            'nama'=>'required',
            'kelas'=>'required',
        ]);
        DB::table('tb_siswa')->insert([
            'nis'=>$request->nis,
            'nama'=>$request->nama,
            'kelas_id'=>$request->kelas,
        ]);
        return redirect('/');
    }
....
```

input.blade.php:

```
<form action="/proses-input" method="post">
        @csrf
        <label>NIS:</label><br/>
        <input type="text" name="nis" placeholder="nis"/><br/>
        <label>NAMA LENGKAP:</label><br/>
        <input type="text" name="nama" placeholder="nama lengkap"/><br/>
        <label>KELAS:</label><br/>
        <select name="kelas" id="kelas">
            @foreach ($kelas as $kls )
              <option value="{{ $kls->id }}">{{ $kls->kelas }}</option>
            @endforeach
        </select>
```

```
        <br/>
        <input type="submit" value="simpan"/>
    </form>
```

+ Tambah siswa baru

NIS	NAMA SISWA	KELAS	AKSI
544987	adit	XI RPL 4	Edit / Hapus
5442311	andi baso	XI RPL 4	Edit / Hapus
5442312	andi besse	XI RPL 4	Edit / Hapus
5442313	daeng kebo	XI RPL 3	Edit / Hapus
5442333	arif rahman	XI RPL 3	Edit / Hapus
98987655	marmud akbar	XI RPL 4	Edit / Hapus

Kembali

NIS:

nis

NAMA LENGKAP:

nama lengkap

KELAS:

XI RPL 1

simpan

edit data:

web.php:

```
.....
Route::get('/edit-data/{id}',[siswaControl-
ler::class,'edit']);
Route::post('/proses-update',[siswaControl-
ler::class,'update']);
...
```

siswaController.php:

```
public function edit($id):View
    {
        $kelas=DB::table('tb_kelas')->get();
        $data=DB::table('tb_siswa')
            ->where('nis', $id)
            ->get();
        return view('edit',[
            'edit'=>$data,
            'kelas'=>$kelas
        ]);
    }

public function update(Request $request)
    {
        DB::table('tb_siswa')
            ->where('nis', $request->nis)
            ->update([
                'nama'=>$request->nama,
                'kelas_id'=>$request->kelas,
            ]);
        return redirect('/');
    }
```

edit.blade.php:

```
@foreach ($edit as $ed )
    <form action="/proses-update
method="post">
        @csrf
        <label>NIS:</label><br/>
        <input type="text" name="nis"
value="{{ $ed->nis }}" placehold-
er="nis"/><br/>
        <label>NAMA LENGKAP:</label><br/>
        <input type="text" name="nama"
value="{{ $ed->nama }}" placeholder="nama
lengkap"/><br/><br/>
        <label>KELAS:</label><br/>
        <select name="kelas" id="kelas">
            @foreach ($kelas as $kls )
              <option value="{{ $kls->id
}}">{{ $kls->kelas }}</option>
            @endforeach
        </select>
        <br/><br/>
        <input type="submit" value="up-
date"/>
```

Kembali

NIS:

5442313

NAMA LENGKAP:

daeng kebo

KELAS:

XI RPL 4

update

+ Tambah siswa baru

NIS	NAMA SISWA	KELAS	AKSI
544987	aditos	XI RPL 3	Edit / Hapus
5442311	andi baso	XI RPL 4	Edit / Hapus
5442312	andi besse	XI RPL 4	Edit / Hapus
5442313	daeng kebo	XI RPL 3	Edit / Hapus
5442333	arif rahman	XI RPL 3	Edit / Hapus
98987655	marmud akbar	XI RPL 4	Edit / Hapus

hapus data:
web.php:

```
Route::get('/hapus-data/{id}',[siswaCon-
troller::class,'hapus']);
```

siswaController.php:

...

```
public function hapus($id)
    {
        DB::table('tb_siswa')-
>where('nis', $id)->delete();
        return redirect('/');
    }
```

....

latihanCrud.php:

....

```
<td>
<a href="/edit-data/{{ $row->nis
}}">Edit</a>/
<a href="/hapus-data/{{ $row->nis
}}">Hapus</a>
</td>
```

....

+ Tambah siswa baru

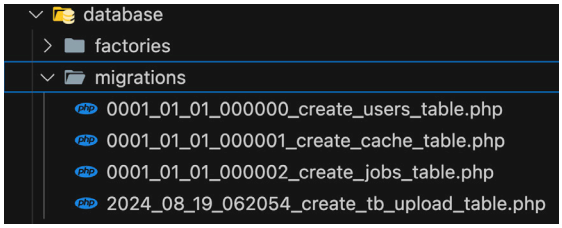
NIS	NAMA SISWA	KELAS	AKSI
5442311	andi baso	XI RPL 4	Edit / Hapus
5442312	andi besse	XI RPL 4	Edit / Hapus
5442313	daeng kebo	XI RPL 3	Edit / Hapus

Upload

fitur upload gambar adalah hal yang wajib ada dalam sebuah aplikasi berikut langkah langkah menambahkan fitur upload gambar pada aplikasi laravel

1. membuat tabel database dengan fitur migration:
`php artisan make:migration create_tb_upload_table`

perintah diatas akan membuat file migration:

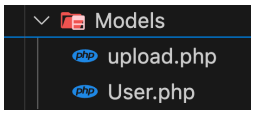



```
public function up(): void
  {
    Schema::create('tb_upload', function (Blueprint $table) {
       $table->id();
       $table->timestamps();
     });
   }
```

```
public function up(): void
    {
        Schema::create('tb_upload', func-
tion (Blueprint $table) {
            $table->id();
            $table->text('file');
            $table->string('nama_file');
            $table->timestamps();
        });
    }
```

2. mensingkronkan file migration dengan database:
`php artisan migrate`

3. membuat model:
`php artisan make:model upload`



...

```
<?php
namespace App\Models;
use   Illuminate\Database\Eloquent\Factories\Has-
Factory;
use Illuminate\Database\Eloquent\Model;
class upload extends Model
{
    protected $table='tb_upload';
    protected $fillable=['file','nama_file'];
}
```

...

upload.blade.php:

```
<form action="/upload/proses" method="POST"
enctype="multipart/form-data">
@csrf
<label>Nama File</label>
<br/>
<input type="text" name="nama_file"/>
<input type="file" name="image">
<button type="submit">Upload</button>
</form>
```

web.php:

```
Route::get('/upload',[siswaController::-
class,'upload']);
Route::post('/upload/proses',[siswaControl-
ler::class,'prosesUpload']);
```

siswaController.php:

```
public function upload():View
    {
        $data=upload::get();
        return view('upload',[
            'gambar'=>$data
        ]);
    }
    public function prosesUpload(Request $request)
    {
        $request->validate([
            'nama_file'=>'required',
            'image' => 'required|image|mimes:jpeg,png,jpg,gif|max:2048',
        ]);
        $gambar=$request->file('image')->store('gambar','public');
        upload::create([
            'nama_file'=>$request->nama_file,
            'file'=>$gambar
        ]);
        return redirect('/upload');
    }
```

4. Apa kepanjangan dari CRUD dalam konteks pengembangan web?
a. Create, Retrieve, Update, Delete
b. Create, Read, Update, Delete
c. Create, Rename, Update, Delete
d. Create, Read, Upgrade, Delete

5. Bagaimana cara mengatur route untuk menyimpan data baru di Laravel?
a. Route::get('/store', [storeController::class,'data']);
b. Route::post('/store', [storeController::class,'data']);
c. Route::put('/store', [storeController::class,'data']);
d. Route::patch('/store', [storeController::class,'data']);

6. Metode mana yang digunakan dalam Laravel untuk memperbarui data yang ada dalam tabel?
a. update()
b. edit()
c. modify()
d. change()

7. Bagaimana cara menambahkan kondisi WHERE dalam Query Builder di Laravel?
a. ->where('column', 'value')
b. ->filter('column', 'value')
c. ->condition('column', 'value')
d. ->match('column', 'value')

8. Bagaimana cara mengambil satu baris pertama dari hasil query menggunakan Query Builder di Laravel?
a. ->first()
b. ->fetchOne()
c. ->getOne()
d. ->one()

9. Di Laravel, file environment (.env) digunakan untuk:
a. Menyimpan konfigurasi aplikasi dan pengaturan environment -
b. Menyimpan template view
c. Menyimpan log aktivitas pengguna
d. Menyimpan data sementara

10. Perintah artisan apa yang digunakan untuk membuat controller baru di Laravel?
a. php artisan make:controller
b. php artisan create:controller
c. php artisan build:controller
d. php artisan new:controller

EXCERCISE?

1. Dalam arsitektur MVC, komponen mana yang bertanggung jawab untuk mengelola logika dan data?

a. Model
b. Controller
c.View
d.Database

2. Komponen View dalam MVC berfungsi untuk:

a. Mengontrol aliran data antara model dan view
b. Menyimpan data dan aturan bisnis
c. Mengatur tampilan dan interaksi dengan pengguna
d. Mengelola rute aplikasi

3. Bagaimana Controller berinteraksi dengan Model dan View dalam arsitektur MVC

a. Controller memperbarui View dan kemudian meminta data dari Model
b. Controller menerima input pengguna, mengirimkannya ke Model, dan memperbarui View-
c. Controller hanya bertanggung jawab untuk menyimpan data di Model
d. Controller hanya berfungsi untuk mengatur View tanpa berinteraksi dengan Model