



Universitas
Telkom



Fakultas
Informatika
School of Computing
Telkom University



informatics lab



FACULTY OF
SYSTEMS &
INFORMATION



BPPT



BJK



GOVERNMENT CERTIFICATION AUTHORITY



TECHtalk

Dhika Anbiya S.Kom M.T.

October 11th, 2020

Microservices

Penyelenggaraan
Sertifikasi Elektronik
PSrE iOTENTIK

Dalam Lingkungan Pemerintah
di Indonesia



Overview

01

Microservices Overview

A brief overview about microservices

02

Application Protocol Interface (API)

What and How is the API

03

Frameworks

The most popular framework

04

Coding

Let's get our hand dirty



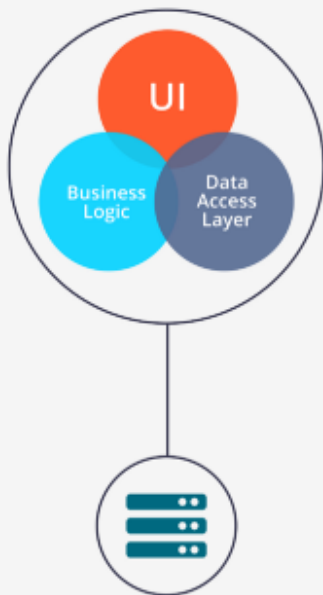
Lets Go!

Microservices

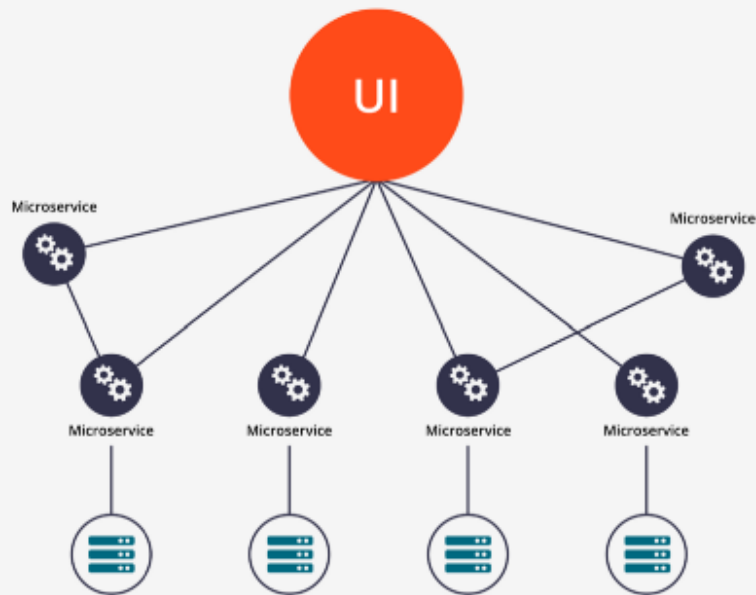
Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of services that are

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities
- Owned by a small team.

Microservices vs Monolithic

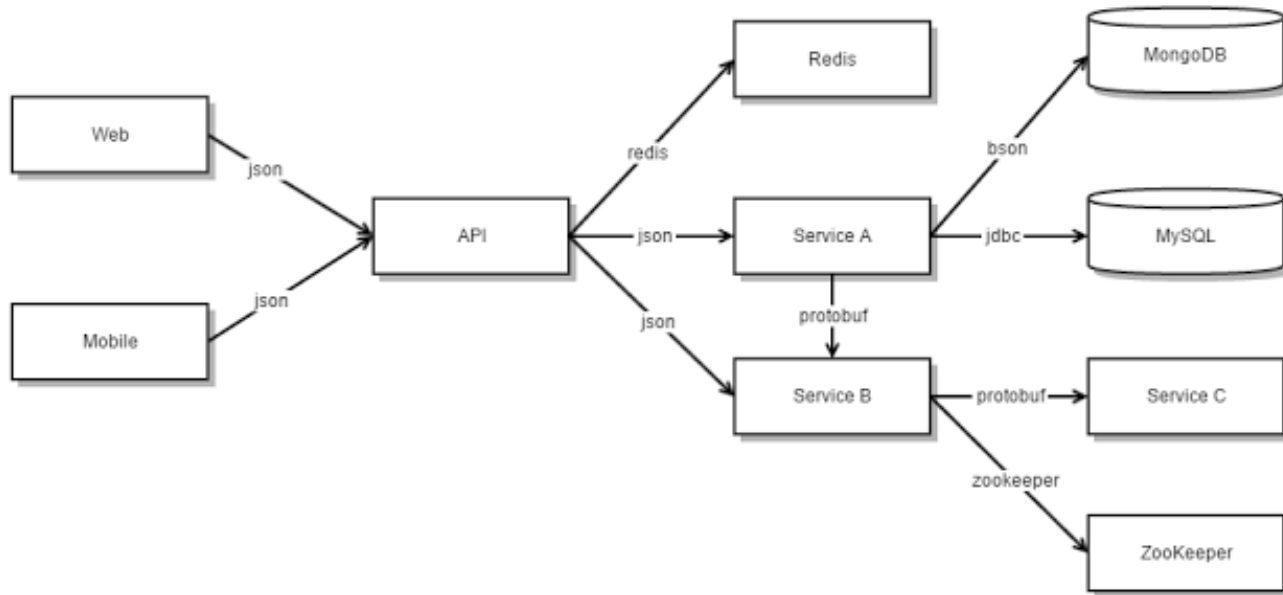


Monolithic Architecture

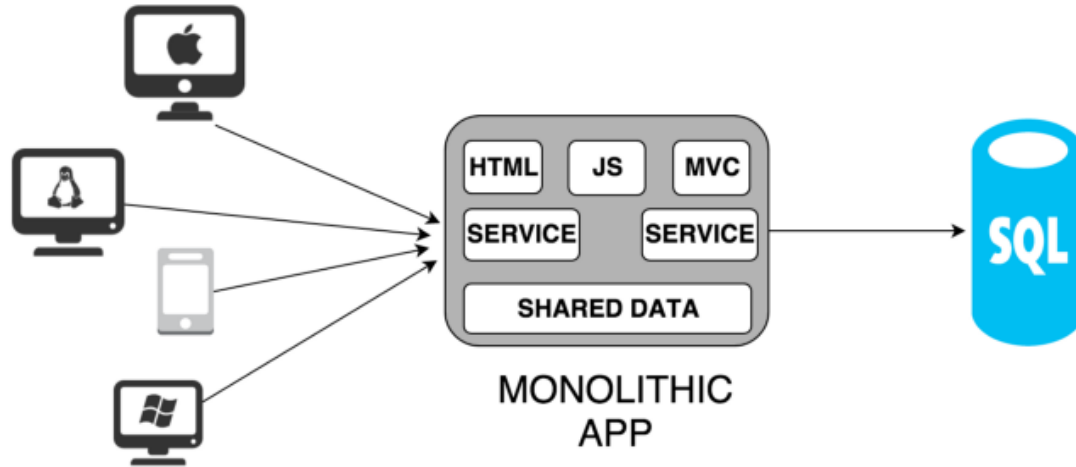


Microservice Architecture

Microservices Architecture



Monolithic Architecture





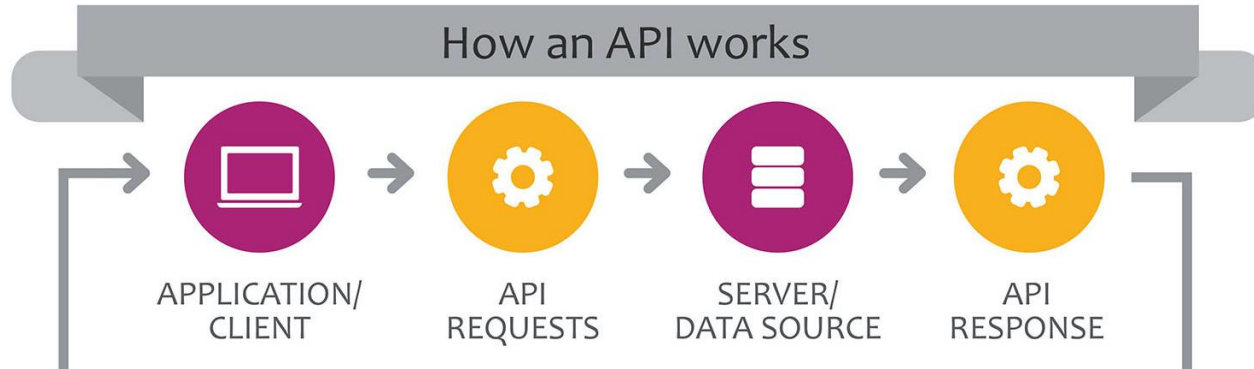
The API

Application Programming Interface

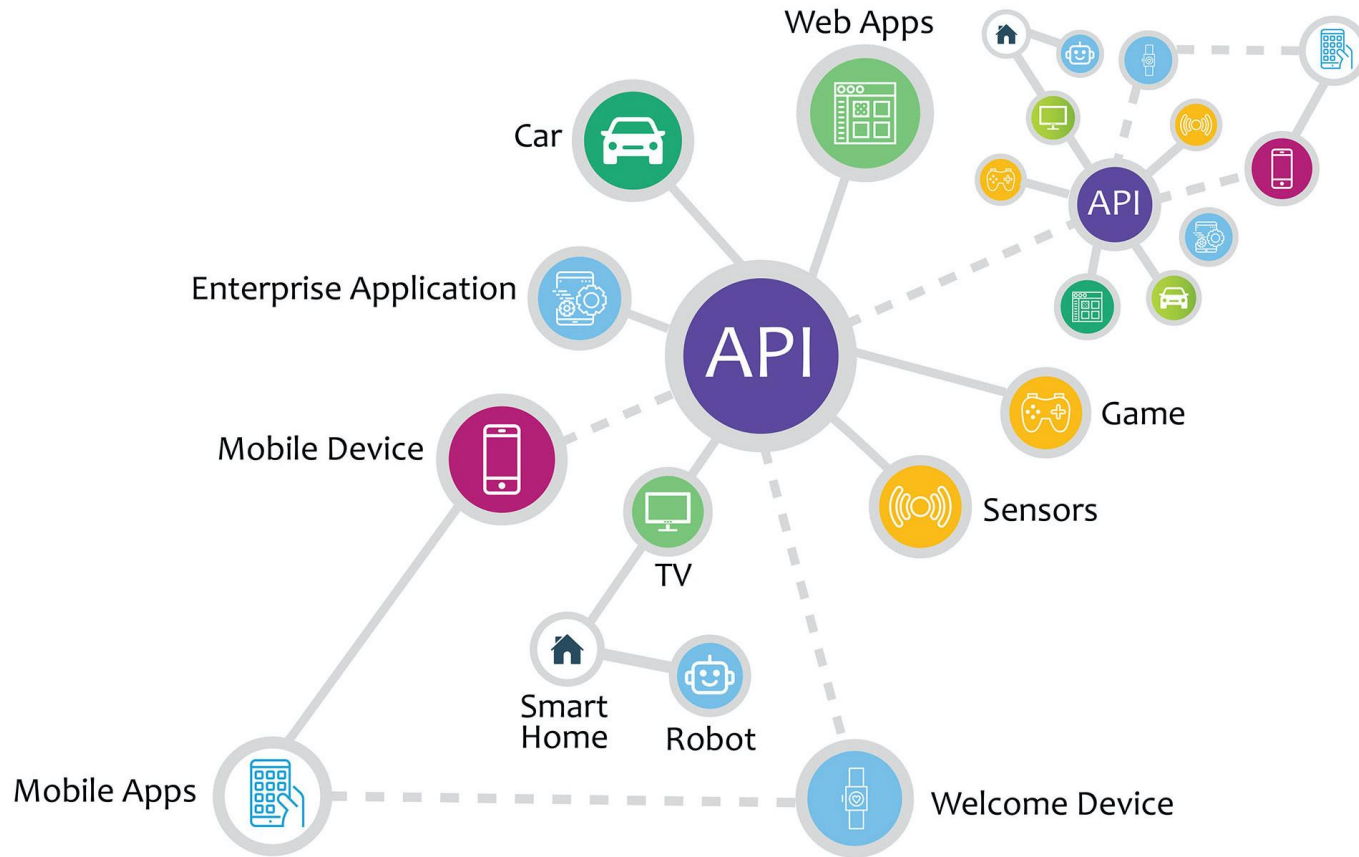


API Definition

An application program interface that provides a developer with programmatic access to a proprietary software application. A software intermediary that makes it possible for application programs to interact with each other and share data.



<https://www.jmbaxigroup.com/newsletter/issue-xxix/apis-application-programming-interfaces/>



REST API

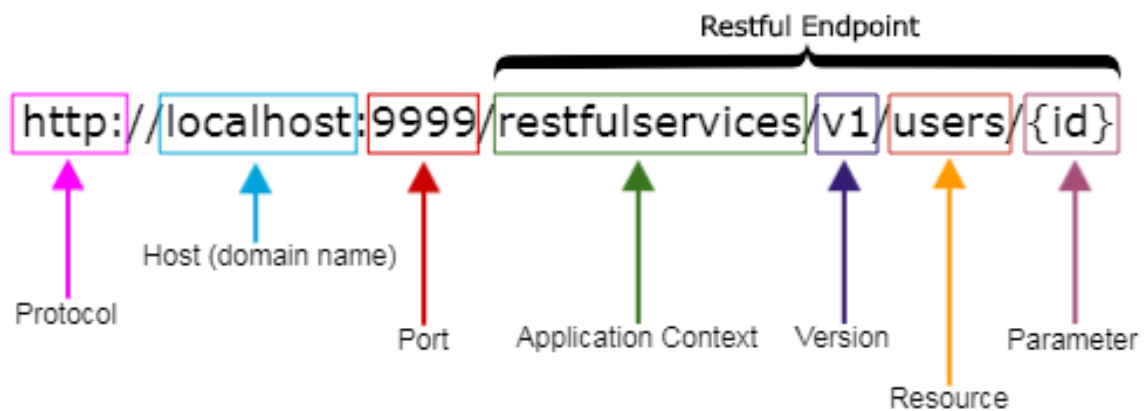
REST is acronym for **RE**presentational **St**ate **T**ransfer. It is architectural style for **distributed hypermedia systems** and was first presented by Roy Fielding in 2000 on his dissertation.

https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Guiding Principles

- Client - Server
- Stateless
- Cacheable
- Uniform interface
- Layered system
- Code on demand





<https://avaldes.com/best-practices-for-restful-api-design/>

Schemes

HTTP



HTTP Methods

Testing different HTTP verbs



DELETE

/delete "The request's DELETE parameters."

GET

/get The request's query parameters.

PATCH

/patch The request's PATCH parameters.

POST

/post The request's POST parameters.

PUT

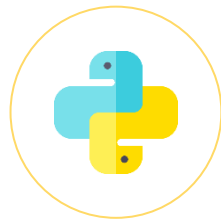
/put The request's PUT parameters.

<https://www.oreilly.com/>

Popular Frameworks



Java Springboot



Python Django



PHP Laravel Lumen



Nodejs Express



Let's get our hands dirty

Our Services

Insert the title of your subtitle Here

01

Your Text Here

You can simply impress your audience and add a unique zing and appeal to your Presentations. Easy to change colors, photos and Text.

02

Your Text Here

You can simply impress your audience and add a unique zing and appeal to your Presentations. Easy to change colors, photos and Text.

03

Your Text Here

You can simply impress your audience and add a unique zing and appeal to your Presentations. Easy to change colors, photos and Text.

04

Your Text Here

You can simply impress your audience and add a unique zing and appeal to your Presentations. Easy to change colors, photos and Text.



What we need



Laragon

Fast Forwarding
Development
Environment



POSTMAN



Visual Studio Code

What we will learn

- Installing Laravel Lumen
- Routes
- Database Connection
- Resource API
- Authentication/Authorization

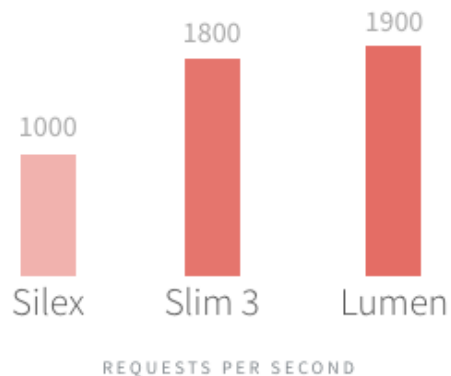




Lumen

Benchmark Breaking Speed

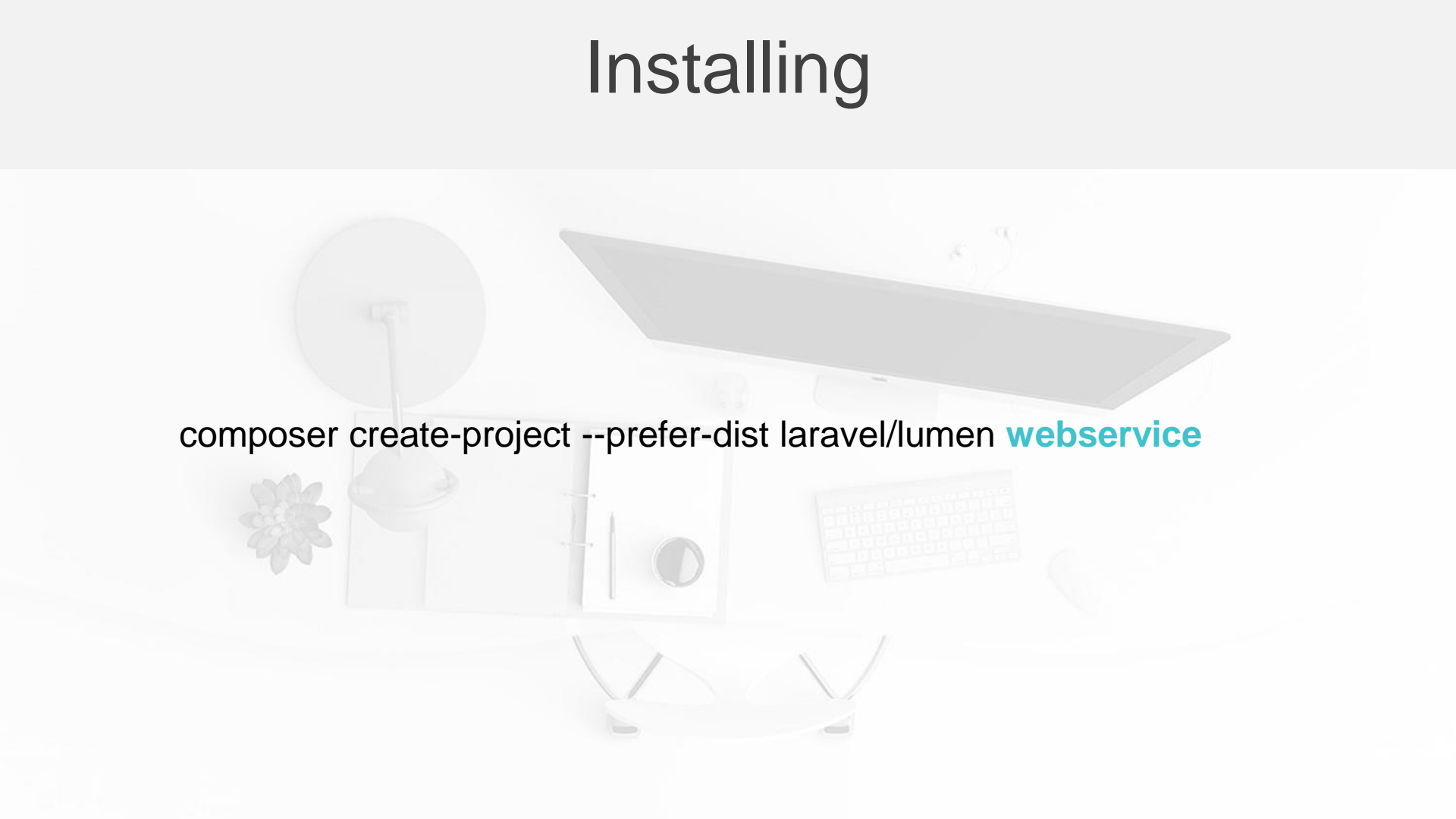
Lumen is the perfect solution for building Laravel based micro-services and blazing fast APIs. In fact, it's one of the fastest micro-frameworks available. It has never been easier to write stunningly fast services to support your Laravel applications.



All the source code are available on github
Just dig in to

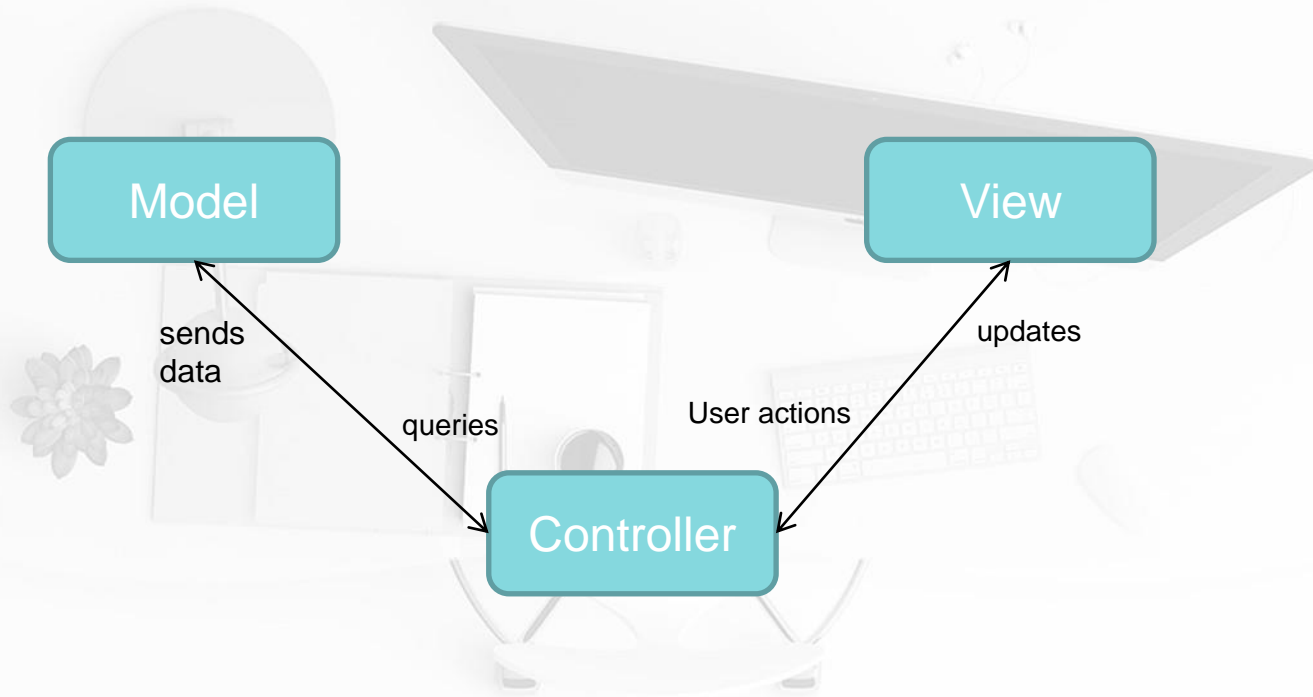
<https://github.com/dhikanbiya/webserviceapp>

Installing



```
composer create-project --prefer-dist laravel/lumen webservice
```

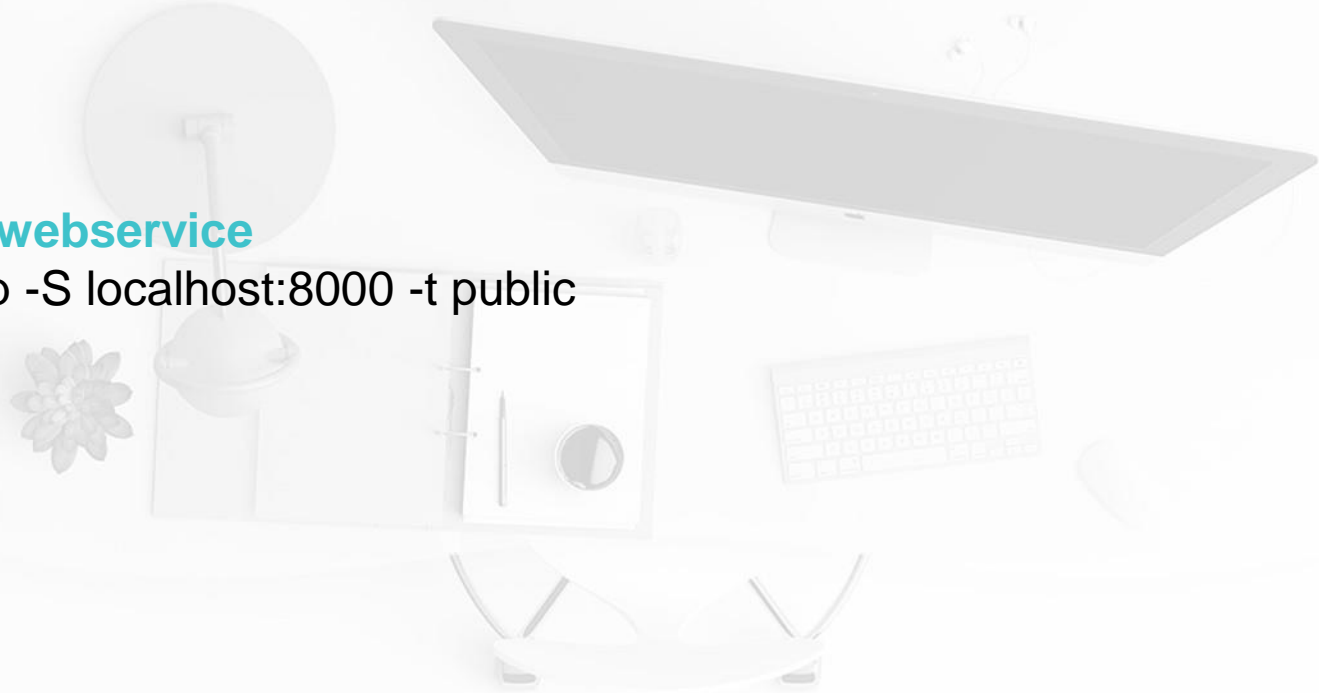
The MVC



Running your first webservice

cd **webservice**

php -S localhost:8000 -t public



Routes

Let's create our first route

Go to routes/web.php

```
$router->get('/', function () use ($router) {  
    return "hello world!";  
});
```

Database Connection

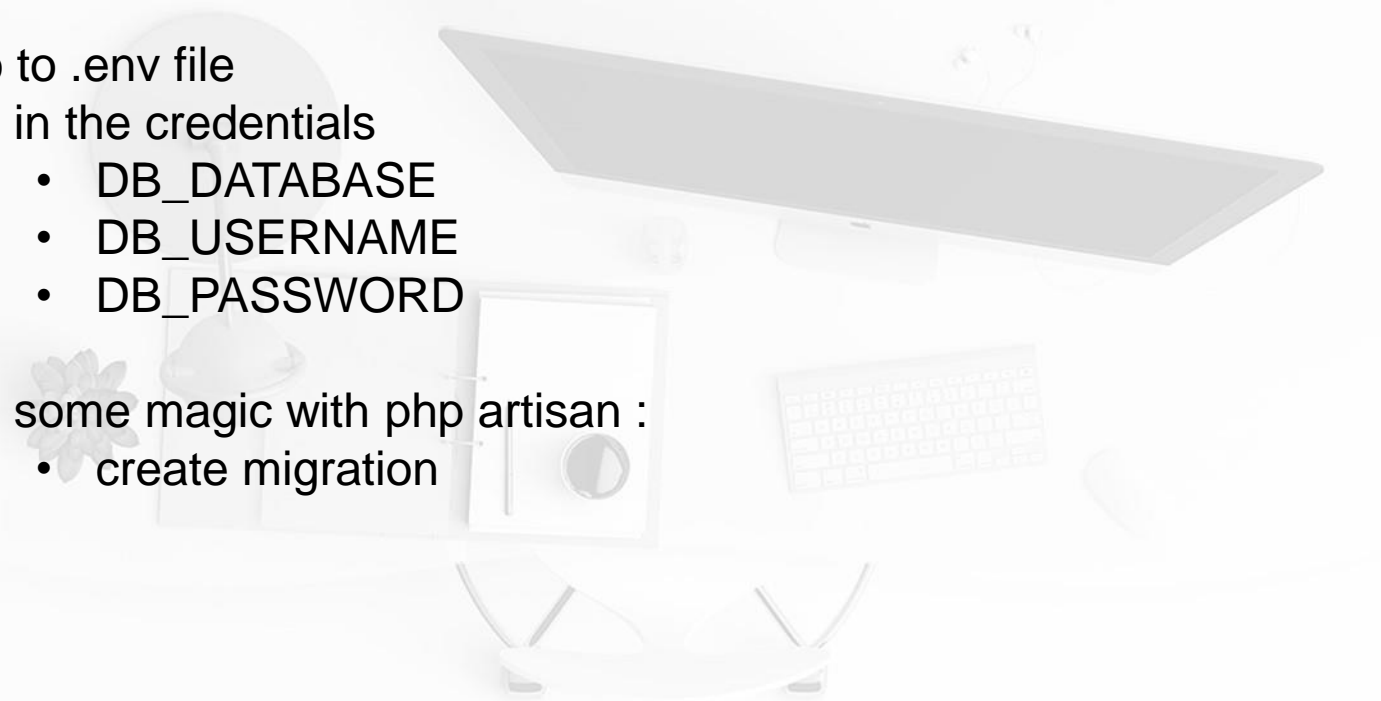
Go to .env file

Fill in the credentials

- DB_DATABASE
- DB_USERNAME
- DB_PASSWORD

Do some magic with php artisan :

- create migration



Database Connection

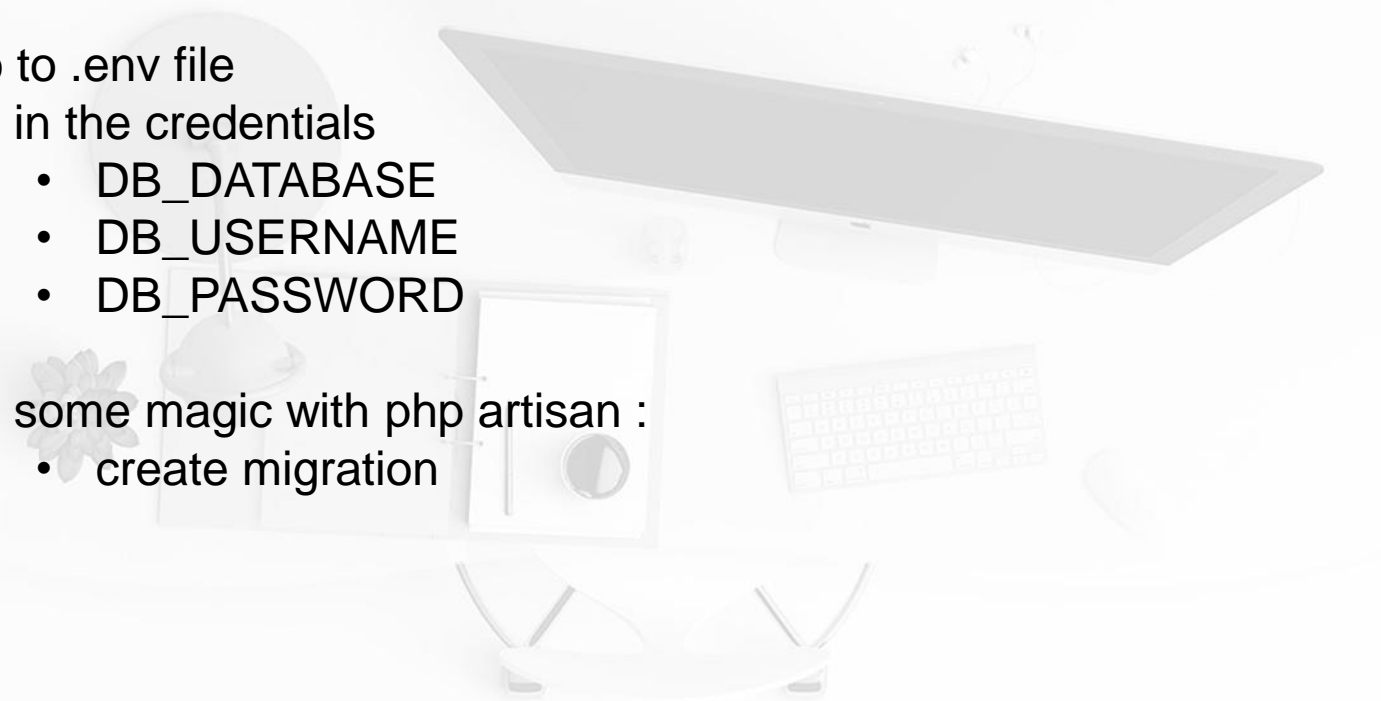
Go to .env file

Fill in the credentials

- DB_DATABASE
- DB_USERNAME
- DB_PASSWORD

Do some magic with php artisan :

- create migration



Simple Hacks in Database

- Enable Eloquent and Facades
 - Simply uncomment in bootstrap/app.php

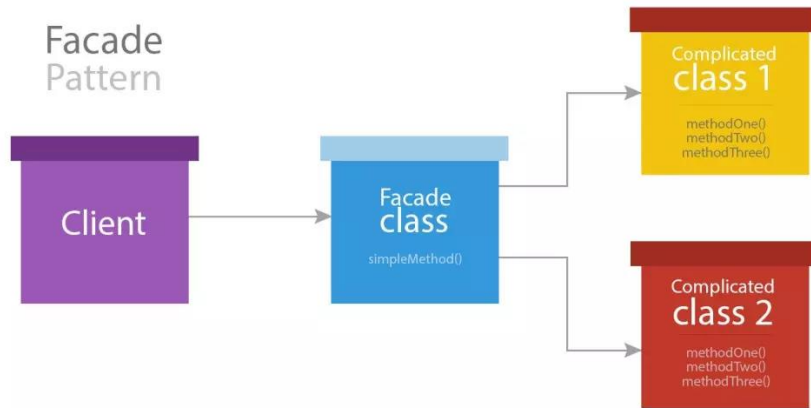
```
$app->withFacades();  
$app->withEloquent();
```

Eloquent

The Eloquent ORM included with Laravel provides a beautiful, simple ActiveRecord implementation for working with your database. Each database table has a corresponding "Model" which is used to interact with that table.

Facades

A **Laravel facade** is a class which provides a static-like interface to services inside the container.





Resource API

Create API

What we will make:

- List articles
- Create article
- Update article
- Delete Article
- Login
- Register



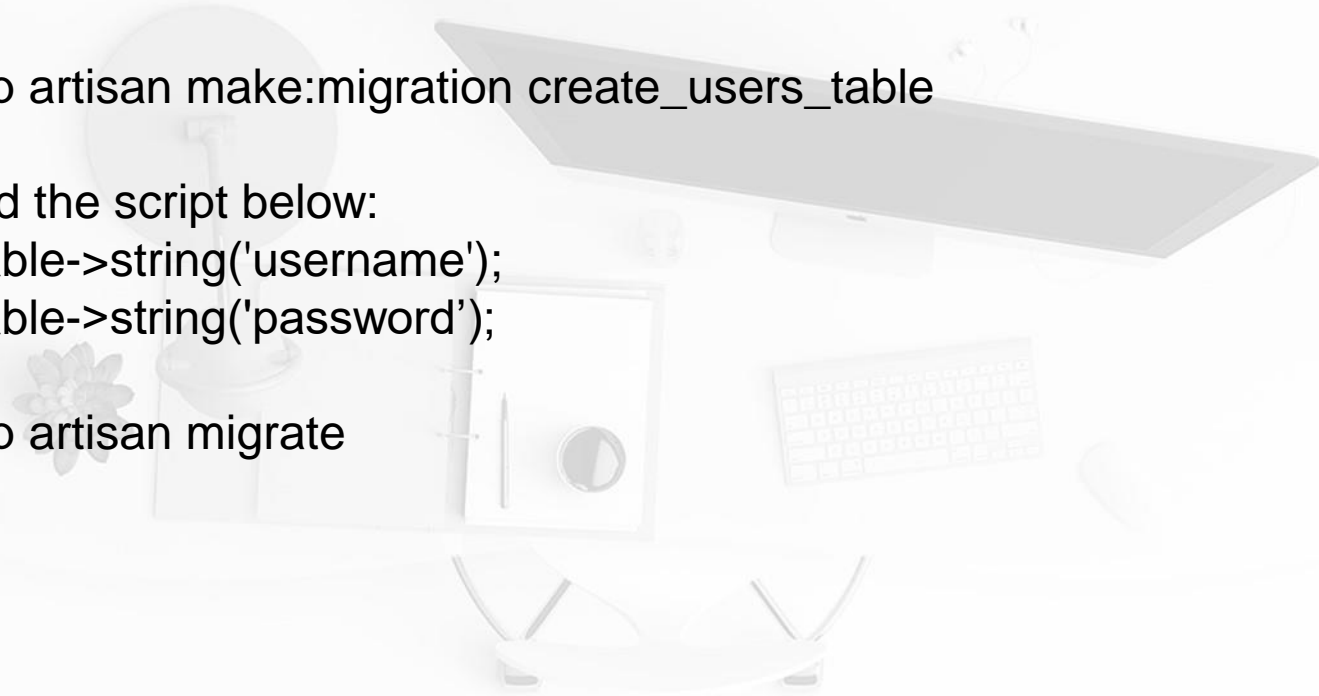
Create Users Table

```
php artisan make:migration create_users_table
```

Add the script below:

```
$table->string('username');  
$table->string('password');
```

```
php artisan migrate
```



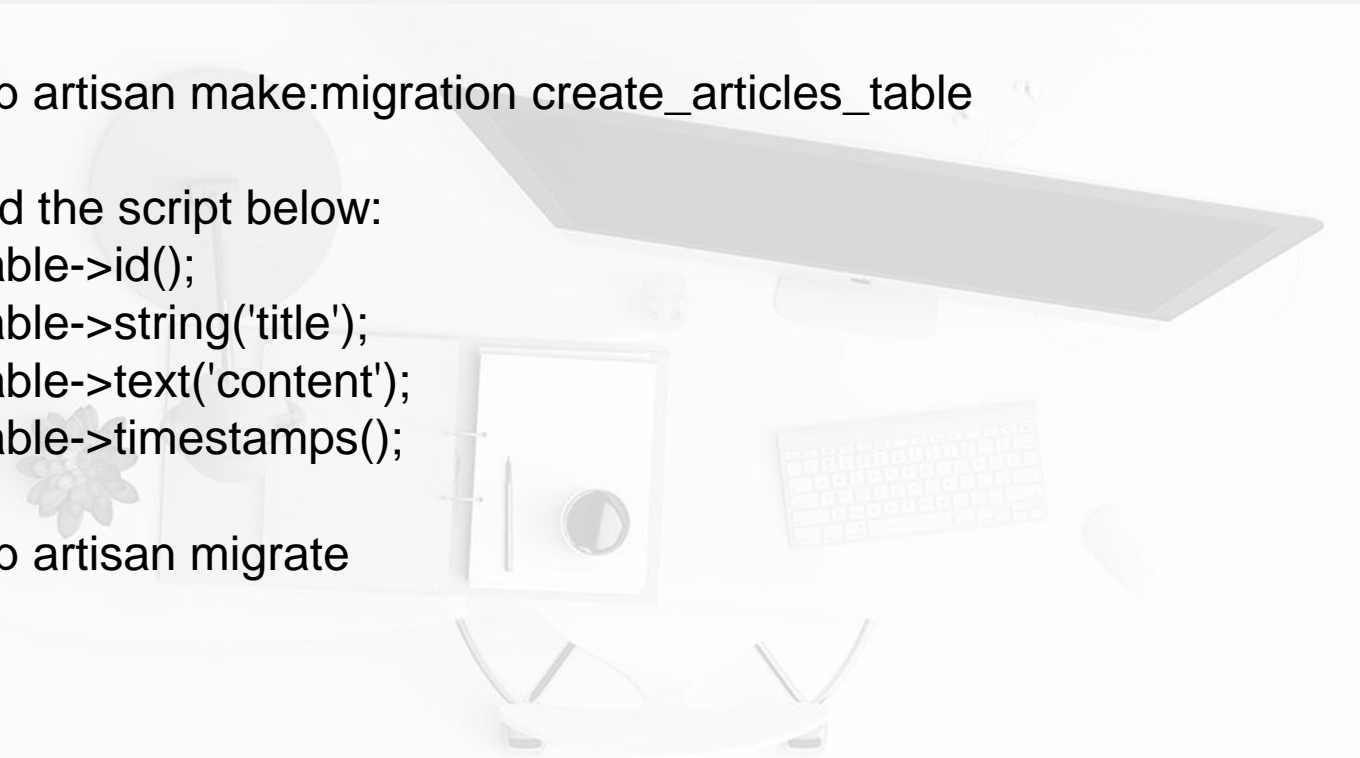
Create Articles Table

```
php artisan make:migration create_articles_table
```

Add the script below:

```
$table->id();  
$table->string('title');  
$table->text('content');  
$table->timestamps();
```

```
php artisan migrate
```



Create First Controller

app/http/Controllers/ArticleController.php

```
public function index()
{
    return response()->json([
        'code' => 200,
        'message' => 'this is article index'
    ]);
}
```

routes/web.php

```
$router->get('/articles', ['uses' => 'ArticleController@index', 'as' => 'articles']);
```

Create a Model

app/Article.php

```
protected $fillable = [  
    'title', 'content',  
];
```

```
protected $hidden = [  
    '',  
];
```



Create insert Article

app/http/Controllers/ArticleController.php

```
public function create(Request $request){
    $article = new Article;
    $article->title = $request->title;
    $article->content= $request->content;

    if($article->save()){
        return response()->json([
            'code' => 200,
            'message' => 'new article created'
        ]);
    }

    return response()->json([
        'code' => 400,
        400,
        'message' => 'error creating article'
    ]);
}
```

routes/web.php

```
$router->post('/create', ['uses' => 'ArticleController@create', 'as' => 'create']);
```

use App\Article;

```
class ArticleController extends Controller
{
    .....
}
```


Add the validation

```
app/http/Controllers/ArticleController.php
```

```
public function create(Request $request){  
  
    $this->validate($request,[  
        'title' => 'required|min:20',  
        'content' => 'required'  
    ]);  
}
```

List All Articles

app/http/Controllers/ArticleController.php

```
public function index()
{
    $article = Article::all();
    return response()->json([
        'code' => 200,
        'message' => 'this is article index'
        'message' => 'articles list',
        'data' => $article
    ]);
}
```

Update Article

app/http/Controllers/ArticleController.php

```
public function update($id, Request $request){
    $article = Article::findOrFail($id);
    $update = $article->update($request->all());

    if($update){
        return response()->json([
            'code' => 200,
            'message' => 'article updated'
        ]);
    }

    return response()->json([
        'code' => 400,
        400,
        'message' => 'error creating article'
    ]);
}
```

routes/web.php

```
$router->put('/update/{id}',[ 'uses'=>'ArticleController@update','as'=>'update']);
```

Delete Article

app/http/Controllers/ArticleController.php

```
public function delete($id){
    $delete = Article::findOrFail($id)->delete();
    if($delete){
        return response()->json([
            'code' => 200,
            'message' => 'article deleted'
        ]);
    }

    return response()->json([
        'code' => 400,
        400,
        'message' => 'error creating article'
    ]);
}
```

routes/web.php

```
$router->delete('/delete/{id}',[ 'uses'=>'ArticleController@delete','as'=>'delete']);
```



Authorization & Authentication

Add api_key

create a migration file

```
php artisan make:migration add_api_key_to_users --table=users
```

Add this script on **function up**

```
$table->string('api_key')->after('password');
```

Add this script on **function down**

```
$table->dropColumn(['api_key']);
```

Do the migration script

```
php artisan migrate
```

Create a seeder

```
php artisan make:seeder UsersTableSeeder
```

Add dependencies

```
use App\User;  
use Illuminate\Support\Str;  
use Illuminate\Support\Facades\Hash;
```

Add this script on function run

```
User::create([  
    'username' => 'user',  
    'password' => Hash::make('secret'),  
    'api_key' => Str::random(32)  
]);
```

Create Login

app/http/Controllers/AuthController.php

```
public function login(Request $request){  
  
    $this->validate($request,[  
        'username' => 'required',  
        'password' => 'required'  
    ]);  
  
    $user = User::where('username',$request->username)->first();  
  
    if(!$user){  
        return response()->json([  
            "code" => "401",  
            "message" => "Bad Credentials",  
            401  
        ]);  
    }  
  
    if(Hash::check($request->password,$user->password)){  
        return response()->json([  
            "code" => "200",  
            "message" => "Login Success",  
            "data" => [  
                "api-key" => $user->api_key  
            ]  
        ]);  
    }  
  
    return response()->json([  
        "code" => "401",  
        "message" => "Bad Credentials",  
        401  
    ]);  
}
```

routes/web.php

```
$router->post('/login',[ 'uses'=>'AuthController@login','as'=>'login']);
```


Create Login

Comment out the lines

```
bootstrap/app.php
```

```
$app->routeMiddleware([  
    'auth' => App\Http\Middleware\Authenticate::class,  
]);
```

```
$app->register(App\Providers\AuthServiceProvider::class);
```

Header Authorization

app/Providers/AuthServiceProvider.php

```
$this->app['auth']->viaRequest('api', function ($request) {  
    if ($request->input('api_token')) {  
        return User::where('api_token', $request->input('api_token'))->first();  
    }  
    if($request->header('Authorization')){  
        $sex = explode(' ', $request->header('Authorization'));  
  
        return User::where('api_key',$sex[1])->first();  
    }  
});
```

Implement the Authorization

app/Http/Controllers/ArticleController.php

```
public function __construct(){  
    $this->middleware('auth',['only'=>[  
        'create','update','delete'  
    ]]);  
}
```



Thank you