Laravel 8 CRUD

Laborator dedicate lui CRUD (Creare, Read, Update, and Delete), un ghid pas cu pas cu exemple care îi pot ajuta pe începători. Elementele de bază în Laravel sunt legate de creare, citire, actualizare și ștergere.

Pas1. Creare aplicație laravel prin scrierea în cmd a comenzii

```
composer create-project --prefer-dist laravel/laravel crud
```

Pas2. Intarea în aplicație cu comanda: `cd crud`

Pas3. In cmd se scrie comanda php artisan serve

Rezultat

Starting Laravel development server: http://127.0.0.1:8000

Pas4. Se lansează browserul și se scrie : http://127.0.0.1:8000

Pas5 : instalăm pachetul Laravel Collective, rulați următoarea comandă de mai jos:

```
composer require laravelcollective/html
```

Pas6. Creare în MySql a bazei de date exlara2021

Pas7. În fișierul .env se fac modificările care ne ajută se ne conectăm la baza de date *exlara2021* creată în mysql se fac modificările

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:RYmyRP/TkuB1T067j8papaI8mTSo9aVKuVgedJNV97Y=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=exlara2021
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
```

```
CACHE_DRIVER=file
FILESYSTEM_DRIVER=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120
```

Pas3. Se modifică fișierul database.php existent în subdirectorul /config astfel

Fișierul database.php

Obs. În fișierul database.php se modifică valorile din liniile

```
    'default' => env('DB_CONNECTION', 'mysql'),

    /*
    |--------------------------------------------------------------------------
    | Database Connections
    |--------------------------------------------------------------------------
    |
....
        'mysql' => [
            'driver' => 'mysql',
            'host' => env('DB_HOST', '127.0.0.1'),
            'port' => env('DB_PORT', '3306'),
            'database' => env('DB_DATABASE', 'exlara'),  //nume bază de date
            'username' => env('DB_USERNAME', 'root'),//usernameul Mysql
            'password' => env('DB_PASSWORD', ''),//parola Mysql
            'unix_socket' => env('DB_SOCKET', ''),
            'charset' => 'utf8',
```

```
        'collation' => 'utf8_unicode_ci',
        'prefix' => '',
        'strict' => true,
        'engine' => null,
    ],
....
```

Pas4. În fereastra cmd se execută comenzile

```
php artisan cache:clear
php artisan config:cache
```

Pas8. Configurare migrare

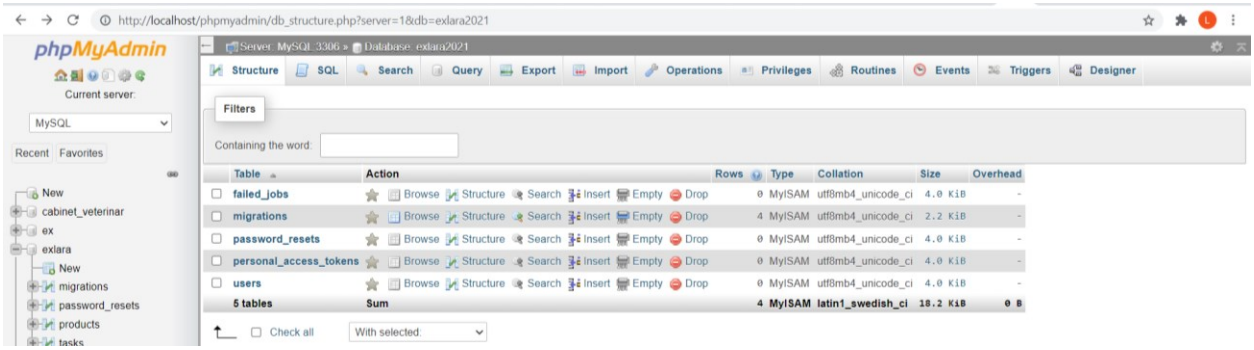
- personalizare tabelul Users, adăugare câmpul nume în autentificare. Iată migrațiile complete de mai jos:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name')->nullable();
            $table->string('email')->unique();
            $table->string('username')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
```

```
    */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

Pas9. `php artisan migrate`

`Efect:` Efectul pasului 9 este in MySql apare



În caz de eroare:

SQLSTATE[42S01]: Base table or view already exists: 1050 Table 'users' already exists (SQL: create table `users` (`id` bigint unsigned not null auto_increment primary key, `name` varchar(191) null, `email` varchar(191) not null, `username` varchar(191) not null, `email_verified_at` timestamp null, `password` varchar(191) not null, `remember_token` varchar(100) null, `created_at` timestamp null, `updated_at` timestamp null) default character set utf8mb4 collate 'utf8mb4_unicode_ci')

Se rezolvă astfel

```
// Modificare  /app/Providers/AppServiceProvider.php:

use Illuminate\Support\Facades\Schema;

public function boot()
{
    Schema::defaultStringLength(191);
}

//ON this error
//    PDOException::("SQLSTATE[42S01]: Base table or view already exists: 1050 Table 'users' already exists")
// After run ->  php artisan migrate:fresh  <- ! Note this will reset all tables in db
```

În directorul app/ a fișierului User.php în care se introduce codul sursă următor:

```php
<?php

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
   use HasApiTokens, HasFactory, Notifiable;

   /**
    * The attributes that are mass assignable.
    *
    * @var string[]
    */
   protected $fillable = [
      'name',
      'email',
'username',
      'password',
   ];

   /**
    * The attributes that should be hidden for serialization.
    *
    * @var array
    */
   protected $hidden = [
      'password',
      'remember_token',
   ];

   /**
    * The attributes that should be cast.
    *
    * @var array
    */
   protected $casts = [
      'email_verified_at' => 'datetime',
   ];
  public function setPasswordAttribute($value)
   {
```

```
        $this->attributes['password'] = bcrypt($value);
    }
}
```

Pas10. Rute de operare CRUD în fișierul **"routes/web.php"**

```
Route::group(['namespace' => 'App\Http\Controllers'], function()
{

    /**
    * User Routes
    */
    Route::group(['prefix' => 'users'], function() {
        Route::get('/', 'UsersController@index')->name('users.index');
        Route::get('/create', 'UsersController@create')->name('users.create');
        Route::post('/create', 'UsersController@store')->name('users.store');
        Route::get('/{user}/show', 'UsersController@show')->name('users.show');
        Route::get('/{user}/edit', 'UsersController@edit')->name('users.edit');
        Route::patch('/{user}/update', 'UsersController@update')->name('users.update');
        Route::delete('/{user}/delete', 'UsersController@destroy')->name('users.destroy');
    });
});
```

Pas 10. Creare Controler

php artisan make:controller UsersController

în directorul ..\crud\app\Http\Controllers apare fișierul UsersController în care se adaugă 7 metode care vor finaliza operațiunea de creare, citire, actualizare și ștergere în Laravel 8.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\User;
use App\Http\Requests\StoreUserRequest;
use App\Http\Requests\UpdateUserRequest;

class UsersController extends Controller
{
    /**
    * afisează toti users
```

```
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $users = User::latest()->paginate(10);

        return view('users.index', compact('users'));
    }

    /**
     * Afisare form pt creare user
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        return view('users.create');
    }

    /**
     * Memorare un user creat acum
     *
     * @param User $user
     * @param StoreUserRequest $request
     *
     * @return \Illuminate\Http\Response
     */
    public function store(User $user, StoreUserRequest $request)
    {
        //Numai în scopuri demonstrative. La crearea unui utilizator sau la invitarea unui utilizator
         // ar trebui să creați o parolă aleatorie generată și să o trimiteți prin e-mail utilizatorului
        $user->create(array_merge($request->validated(), [
            'password' => 'test'
        ]));

        return redirect()->route('users.index')
            ->withSuccess(__('User created successfully.'));
    }

    /**
     * Afisare date user
     *
     * @param User $user
     *
     * @return \Illuminate\Http\Response
     */
    public function show(User $user)
```

```
    {
        return view('users.show', [
            'user' => $user
        ]);
    }

    /**
     * Creare date user
     *
     * @param User $user
     *
     * @return \Illuminate\Http\Response
     */
    public function edit(User $user)
    {
        return view('users.edit', [
            'user' => $user
        ]);
    }

    /**
     * Update date user
     *
     * @param User $user
     * @param UpdateUserRequest $request
     *
     * @return \Illuminate\Http\Response
     */
    public function update(User $user, UpdateUserRequest $request)
    {
        $user->update($request->validated());

        return redirect()->route('users.index')
            ->withSuccess(__('User updated successfully.'));
    }

    /**
     * Stergere date user
     *
     * @param User $user
     *
     * @return \Illuminate\Http\Response
     */
    public function destroy(User $user)
    {
        $user->delete();

        return redirect()->route('users.index')
```

```
        ->withSuccess(__('User deleted successfully.'));
    }
}
}
```

Pas11. Creare layouts de tip crud blades. View are următoare directoare și fișiere în resources\views

layouts/app-master.blade.php

layouts/partials/navbar.blade.php

layouts/partials/messages.blade.php

users/create.blade.php

users/edit.blade.php

users/index.blade.php

users/show.blade.php

Pagina principală este resources/views /layouts/app-master.blade.php

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Lista Sarcini</title>
     <!-- Bootstrap CSS File  -->
          <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- The above 2 meta tags *must* come first in the head; any other head content must come *after*
these tags -->
    <meta name="description" content="">
    <meta name="author" content="">
   <link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.0.0/css/bootstrap.min.css">
</head>
<body>
 <div class="container">
     <nav class="navbar navbar-default">
       <div class="container-fluid">
          <div class="navbar-header">
            <a class="navbar-brand" href="#">Exemplu aplicatie-Laravel CRUD </a>
```

```
        </div>
        <ul class="nav navbar-nav">
         </ul>
      </div>
   </nav>

   <main class="container mt-5">
      @yield('content')
   </main>

   <script src="{!! url('assets/bootstrap/js/bootstrap.bundle.min.js') !!}"></script>

  </body>
</html>
```

resources/views/layouts/partials/navbar.blade.php

```
<header class="p-3 bg-dark text-white">
  <div class="container">
   <div class="d-flex flex-wrap align-items-center justify-content-center justify-content-lg-start">
    <a href="/" class="d-flex align-items-center mb-2 mb-lg-0 text-white text-decoration-none">
     <svg class="bi me-2" width="40" height="32" role="img" aria-label="Bootstrap"><use
xlink:href="#bootstrap"/></svg>
    </a>

    <ul class="nav col-12 col-lg-auto me-lg-auto mb-2 justify-content-center mb-md-0">
     <li><a href="{{ route('users.index') }}" class="nav-link px-2 text-white">Home</a></li>

   </div>
  </div>
</header>
```

resources/views/layouts/partials/messages.blade.php

```
@if(Session::get('success', false))
   <?php $data = Session::get('success'); ?>
   @if (is_array($data))
      @foreach ($data as $msg)
         <div class="alert alert-success" role="alert">
            <i class="fa fa-check"></i>
            {{ $msg }}
         </div>
      @endforeach
   @else
      <div class="alert alert-success" role="alert">
```

```
        <i class="fa fa-check"></i>
        {{ $data }}
      </div>
    @endif
@endif
```

resources/views/users/index.blade.php

```
@extends('layouts.app-master')

@section('content')
  <div class="bg-light p-4 rounded">
    <h1>Users</h1>
    <div class="lead">
      Manage your users here.
      <a href="{{ route('users.create') }}" class="btn btn-primary btn-sm float-right">Add new user</a>
    </div>

    <div class="mt-2">
      @include('layouts.partials.messages')
    </div>

    <table class="table table-striped">
      <thead>
      <tr>
        <th scope="col" width="1%">#</th>
        <th scope="col" width="15%">Name</th>
        <th scope="col">Email</th>
        <th scope="col" width="10%">Username</th>
        <th scope="col" width="1%" colspan="3"></th>
      </tr>
      </thead>
      <tbody>
        @foreach($users as $user)
          <tr>
            <th scope="row">{{ $user->id }}</th>
            <td>{{ $user->name }}</td>
            <td>{{ $user->email }}</td>
            <td>{{ $user->username }}</td>
            <td><a href="{{ route('users.show', $user->id) }}" class="btn btn-warning btn-sm">Show</a></td>
            <td><a href="{{ route('users.edit', $user->id) }}" class="btn btn-info btn-sm">Edit</a></td>
            <td>
              {!! Form::open(['method' => 'DELETE','route' => ['users.destroy', $user->id],'style'=>'display:inline']) !!}
```

```
                    {!! Form::submit('Delete', ['class' => 'btn btn-danger btn-sm']) !!}
                    {!! Form::close() !!}
                  </td>
                </tr>
              @endforeach
            </tbody>
          </table>

          <div class="d-flex">
            {!! $users->links() !!}
          </div>

        </div>
@endsection
```

resources/views/users/create.blade.php

```
@extends('layouts.app-master')

@section('content')
  <div class="bg-light p-4 rounded">
    <h1>Add new user</h1>
    <div class="lead">
      Add new user and assign role.
    </div>

    <div class="container mt-4">
      <form method="POST" action="">
        @csrf
        <div class="mb-3">
          <label for="name" class="form-label">Name</label>
          <input value="{{ old('name') }}"
            type="text"
            class="form-control"
            name="name"
            placeholder="Name" required>

          @if ($errors->has('name'))
            <span class="text-danger text-left">{{ $errors->first('name') }}</span>
          @endif
        </div>
        <div class="mb-3">
          <label for="email" class="form-label">Email</label>
          <input value="{{ old('email') }}"
            type="email"
            class="form-control"
            name="email"
```

```
                placeholder="Email address" required>
            @if ($errors->has('email'))
                <span class="text-danger text-left">{{ $errors->first('email') }}</span>
            @endif
        </div>
        <div class="mb-3">
            <label for="username" class="form-label">Username</label>
            <input value="{{ old('username') }}"
                type="text"
                class="form-control"
                name="username"
                placeholder="Username" required>
            @if ($errors->has('username'))
                <span class="text-danger text-left">{{ $errors->first('username') }}</span>
            @endif
        </div>

        <button type="submit" class="btn btn-primary">Save user</button>
        <a href="{{ route('users.index') }}" class="btn btn-default">Back</a>
    </form>
  </div>

 </div>
@endsection
```

resources/views/users/show.blade.php

```
@extends('layouts.app-master')

@section('content')
 <div class="bg-light p-4 rounded">
  <h1>Show user</h1>
  <div class="lead">

  </div>

  <div class="container mt-4">
    <div>
      Name: {{ $user->name }}
    </div>
    <div>
      Email: {{ $user->email }}
    </div>
```

```
        <div>
            Username: {{ $user->username }}
        </div>
    </div>

</div>
<div class="mt-4">
    <a href="{{ route('users.edit', $user->id) }}" class="btn btn-info">Edit</a>
    <a href="{{ route('users.index') }}" class="btn btn-default">Back</a>
</div>
@endsection
```

```
@extends('layouts.app-master')

@section('content')
  <div class="bg-light p-4 rounded">
    <h1>Update user</h1>
    <div class="lead">

    </div>

    <div class="container mt-4">
      <form method="post" action="{{ route('users.update', $user->id) }}">
        @method('patch')
        @csrf
        <div class="mb-3">
          <label for="name" class="form-label">Name</label>
          <input value="{{ $user->name }}"
            type="text"
            class="form-control"
            name="name"
            placeholder="Name" required>

          @if ($errors->has('name'))
            <span class="text-danger text-left">{{ $errors->first('name') }}</span>
          @endif
        </div>
        <div class="mb-3">
          <label for="email" class="form-label">Email</label>
          <input value="{{ $user->email }}"
            type="email"
            class="form-control"
            name="email"
            placeholder="Email address" required>
          @if ($errors->has('email'))
```

```
                <span class="text-danger text-left">{{ $errors->first('email') }}</span>
            @endif
          </div>
          <div class="mb-3">
            <label for="username" class="form-label">Username</label>
            <input value="{{ $user->username }}"
              type="text"
              class="form-control"
              name="username"
              placeholder="Username" required>
            @if ($errors->has('username'))
              <span class="text-danger text-left">{{ $errors->first('username') }}</span>
            @endif
          </div>

          <button type="submit" class="btn btn-primary">Update user</button>
          <a href="{{ route('users.index') }}" class="btn btn-default">Cancel</button>
        </form>
      </div>

    </div>
@endsection
```

În /app/Providers/AppServiceProvider.php se fac modificările

```
<?php

namespace App\Providers;
use Illuminate\Pagination\Paginator;
use Illuminate\Support\ServiceProvider;
use Illuminate\Support\Facades\Schema;


class AppServiceProvider extends ServiceProvider
{
    /**
     * Register any application services.
     *
     * @return void
     */
    public function register()
    {
        //
    }
```

```
    /**
     * Bootstrap any application services.
     *
     * @return void
     */
    public function boot()
    {


    Schema::defaultStringLength(191);
Paginator::useBootstrap();


    }
}
```

În caz de eroare

```
php artisan config:cache
php artisan cache:clear
php artisan route:cache
```

Pasul 12: Adăugați solicitări

În UsersController, s-a implementat o cerere specifică pentru fiecare acțiune, aceasta va ajuta să se scurteze codul și definire funcții și verificarea într-o altă clasă.

Se rulează im cmd următoarele comenzi:

php artisan make:request StoreUserRequest

php artisan make:request UpdateUserRequest

efect apar în ..\app\Http\Requests fișierele

StoreUserRequest.php

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class StoreUserRequest extends FormRequest
{
```

```
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'name' => 'required',
            'email' => 'required|email:rfc,dns|unique:users,email',
            'username' => 'required|unique:users,username',
        ];
    }
}
```

UpdateUserRequest.php

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class UpdateUserRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
```

```
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {

            $user = request()->route('user');

        return [
            'name' => 'required',
            'email' => 'required|email:rfc,dns|unique:users,email,'.$user->id,
            'username' => 'required|unique:users,username,'.$user->id,

        ];
    }
}
```