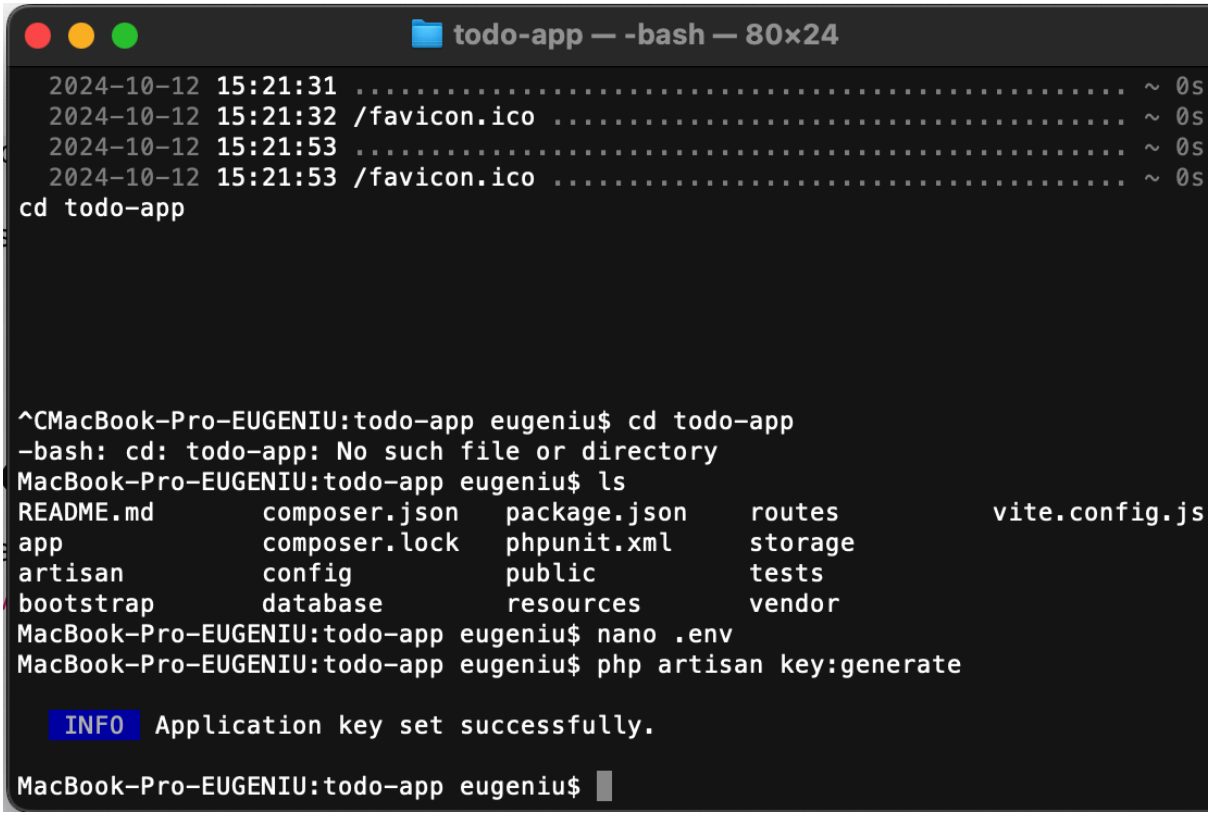
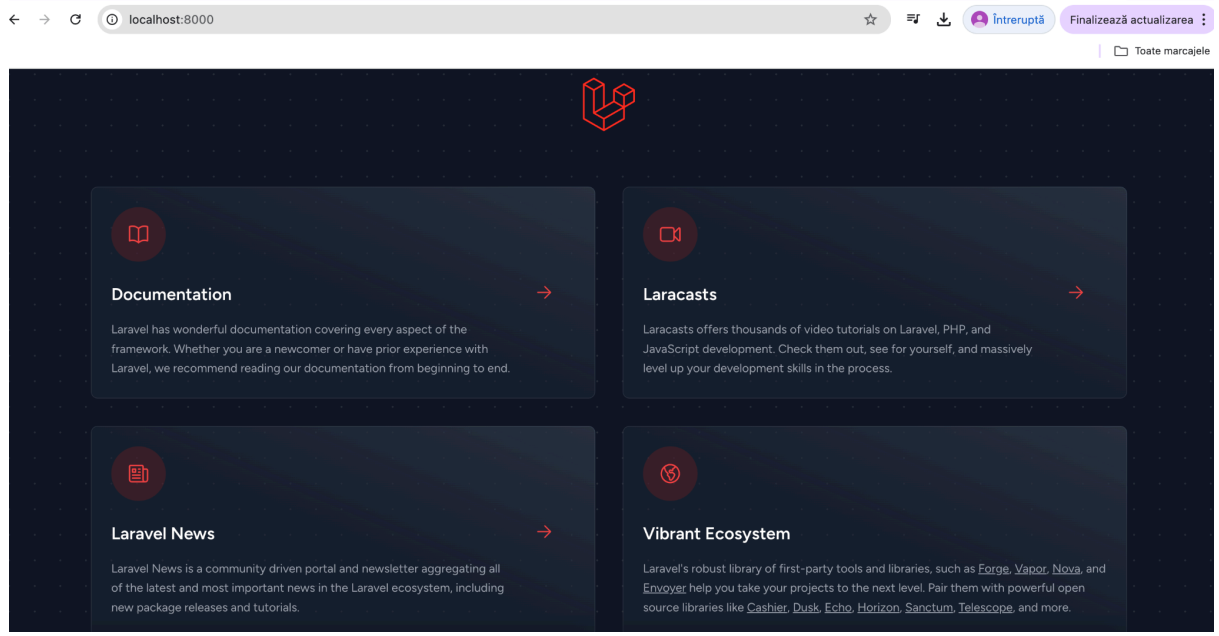
Sarcina nr.1. Pregătirea pentru lucru, instalarea Laravel

Am deschis terminalul si am creat un proiect Laravel cu numele `todo-app`



Când deschid pagina http://localhost:8000/ în browser îmi apare pagina de bun venit Laravel



Sarcina nr.2. Configurarea mediului

Am deschis fisierul `.env` si am setat configurarile

```
MacBook-Pro-EUGENIU:todo-app eugeniu$ nano .env

  UW PICO 5.09                      File: .env                      Modified

APP_NAME=ToDoApp
APP_ENV=local
APP_KEY=
APP_DEBUG=true
APP_URL=http://localhost:8000
```

Am generat cheia aplicatiei , care este utilizata pentru criptarea datelor : `bash php artisan key:generate`

```
MacBook-Pro-EUGENIU:todo-app eugeniu$ php artisan key:generate

   INFO  Application key set successfully.

MacBook-Pro-EUGENIU:todo-app eugeniu$
```

Intrebare : Ce s-ar intimpla daca aceasta cheie ar ajunge pe mina unui raufacator ?

Daca cheia ajunge in mainile unui raufacator, pot aparea mai multe probleme serioase. In primul rand, acesta ar putea accesa date sensibile, de exemplu, ar putea decripta parolele utilizatorilor sau informatii confidentiale stocate in aplicatie. In al doilea rand, un atacator ar putea genera sesiuni false, obtinand astfel controlul asupra conturilor utilizatorilor. In plus, daca cheia este schimbata dupa criptarea datelor, aceste informatii devin inaccesibile, deoarece nu mai pot fi decriptate. Astfel, este esential sa protejezi aceasta cheie pentru a asigura securitatea aplicatiei tale.

Sarcina nr.3. Principiile de bază ale lucrului cu cererile HTTP

3.1. Crearea rutelor pentru pagina principală și pagina "Despre noi"

Am creat un controller `HomeController` pentru gestionarea cererilor catre pagina principala

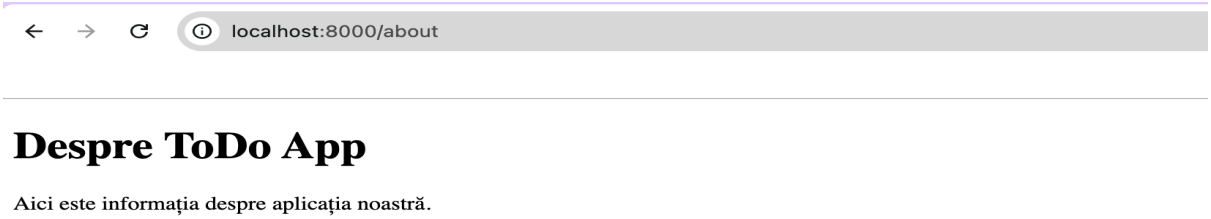
```
MacBook-Pro-EUGENIU:~ eugeniu$ cd ~/Downloads/todo-app
MacBook-Pro-EUGENIU:todo-app eugeniu$ php artisan make:controller HomeController

   INFO  Controller [app/Http/Controllers/HomeController.php] created successful
ly.

MacBook-Pro-EUGENIU:todo-app eugeniu$
```

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class HomeController extends Controller
{
    public function index()
    {
        return view('home');
    }

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

Am adaugat metoda index in HomeController , care va afisa pagina pricipala.
Am creat ruta pentru pagina principala in fisierul `routes/web.php`

```
php public function index() {
        return view('home')
```

In acelasi controller HomeController , am creat o metoda pentru pagina “Despre noi “
La fel am adaugat ruta pentru pagina “Despre noi” in fisierul `routes/web.php`



3.2. Crearea rutelor pentru sarcini

Am creat un controller ‘`TaskController`’ pentru gestionarea cererilor legate de sarcini si am adaugat urmatoarele metode

```
MacBook-Pro-EUGENIU:todo-app eugeniu$ php artisan make:controller TaskController

  INFO  Controller [app/Http/Controllers/TaskController.php] created successfully.

MacBook-Pro-EUGENIU:todo-app eugeniu$
```

```
  UW PICO 5.09          File: app/Http/Controllers/TaskController.php          Modified


namespace App\Http\Controllers;

use Illuminate\Http\Request;

class TaskController extends Controller
{
     public function index()
    {
        return 'Aceasta este o listă de sarcini';
    }

    public function create()
    {
    }

    public function store(Request $request)
    {
    }

    public function show($id)
    {
        return "Afișează sarcina cu ID-ul: $id";
    }

    public function edit($id)
    {
    }

    public function update(Request $request, $id)
    {
    }

    public function destroy($id)
    {
    }

}
```

Am utilizat gruparea rutelor pentru controllerul ‘TaskController’ cu prefixul ‘/task’ pentru a simplifica rutarea si a imbunatati lizibilitatea codului

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\HomeController;
use App\Http\Controllers\TaskController;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/

Route::get('/', [HomeController::class, 'index']);
Route::get('/about', [HomeController::class, 'about']);



Route::prefix('tasks')->group(function () {
    Route::get('/', [TaskController::class, 'index'])->name('tasks.index');
    Route::get('/create', [TaskController::class,
'create'])->name('tasks.create');
    Route::post('/', [TaskController::class, 'store'])->name('tasks.store');
    Route::get('/{id}', [TaskController::class,
'show'])->name('tasks.show')->where('id', '[0-9]+');
    Route::get('/{id}/edit', [TaskController::class,
'edit'])->name('tasks.edit')->where('id', '[0-9]+');
    Route::put('/{id}', [TaskController::class,
'update'])->name('tasks.update')->where('id', '[0-9]+');
    Route::delete('/{id}', [TaskController::class,
'destroy'])->name('tasks.destroy')->where('id', '[0-9]+');
});
```

```
MacBook-Pro-EUGENIU:todo-app eugeniu$ php artisan route:list

  GET|HEAD  / ..................................................... HomeController@index
  POST      _ignition/execute-solution ignition.executeSolution › Spatie\LaravelIgnit…
  GET|HEAD  _ignition/health-check ignition.healthCheck › Spatie\LaravelIgnition › He…
  POST      _ignition/update-config ignition.updateConfig › Spatie\LaravelIgnition › …
  GET|HEAD  about ................................................. HomeController@about
  GET|HEAD  api/user ..............................................................
  GET|HEAD  sanctum/csrf-cookie sanctum.csrf-cookie › Laravel\Sanctum › CsrfCookieCon…
  GET|HEAD  tasks ..................................... tasks.index › TaskController@index
  POST      tasks ..................................... tasks.store › TaskController@store
  GET|HEAD  tasks/create ........................... tasks.create › TaskController@create
  GET|HEAD  tasks/{id} ................................ tasks.show › TaskController@show
  PUT       tasks/{id} ............................. tasks.update › TaskController@update
  DELETE    tasks/{id} ........................... tasks.destroy › TaskController@destroy
  GET|HEAD  tasks/{id}/edit ........................... tasks.edit › TaskController@edit

                                                                  Showing [14] routes
```

Am definit nume corecte pentru rutele controllerrului ‘TaskController’:

- **tasks.index**: Afiseaza lista de sarcini.
- **tasks.create**: Afiseaza formularul pentru crearea unei sarcini.
- **tasks.store**: Salveaza o sarcina noua.
- **tasks.show**: Afiseaza detaliile unei sarcini individuale pe baza ID-ului.
- **tasks.edit**: Afiseaza formularul pentru editarea unei sarcini existente.
- **tasks.update**: Actualizeaza o sarcina existenta.
- **tasks.destroy**: Sterge o sarcina existenta.

Am incercat ca in loc sa creez manual rute pentru fiecare metoda , sa folosesc “un controller de resurse” care sa creeze rute pentru toate operatiunile “CRUD”

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\HomeController;
use App\Http\Controllers\TaskController;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/

Route::get('/', [HomeController::class, 'index']);
Route::get('/about', [HomeController::class, 'about']);



Route::resource('tasks', TaskController::class)
    ->parameters(['tasks' => 'id'])
    ->where(['id' => '[0-9]+']);
```

```
MacBook-Pro-EUGENIU:todo-app eugeniu$ php artisan route:list

  GET|HEAD        / ..................................... HomeController@index
  POST            _ignition/execute-solution ignition.executeSolution › Spatie\Larave…
  GET|HEAD        _ignition/health-check ignition.healthCheck › Spatie\LaravelIgnitio…
  POST            _ignition/update-config ignition.updateConfig › Spatie\LaravelIgnit…
  GET|HEAD        about ................................... HomeController@about
  GET|HEAD        api/user ...............................................................
  GET|HEAD        sanctum/csrf-cookie sanctum.csrf-cookie › Laravel\Sanctum › CsrfCoo…
  GET|HEAD        tasks ........................... tasks.index › TaskController@index
  POST            tasks ........................... tasks.store › TaskController@store
  GET|HEAD        tasks/create .................. tasks.create › TaskController@create
  GET|HEAD        tasks/{id} ...................... tasks.show › TaskController@show
  PUT|PATCH       tasks/{id} .................... tasks.update › TaskController@update
  DELETE          tasks/{id} .................. tasks.destroy › TaskController@destroy
  GET|HEAD        tasks/{id}/edit ................... tasks.edit › TaskController@edit

                                                                  Showing [14] routes
```

Intrebare: Explicati diferenta intre crearea manuala a rutelor si utilizarea unui controller de resurse . Ce rute si ce nume de rute vor fi create automat ?

Crearea manuala a rutelor implica definirea fiecarei rute individual, ceea ce poate duce la un cod mai lung si mai greu de intretinut . Fiecare metoda a controller-ului necesita o linie separata de cod, ceea ce face ca fisierul de rute sa devina aglomerat si mai putin lizibil.

Pe de alta parte, dupa observatiile din imaginile de mai sus ,rutele vor functiona la fel dar utilizarea unui controller de resurse simplifica acest proces. Printr-o singura comanda, Laravel genereaza

automat rutele necesare pentru toate operatiunile CRUD (Create, Read, Update, Delete). Aceasta metoda nu doar ca reduce cantitatea de cod scris, dar si imbunatateste organizarea si claritatea acestuia

Rutele create automat printr-un controller de resurse includ:

- tasks.index: pentru afisarea listei de sarcini (GET /tasks)
- tasks.create: pentru afisarea formularului de creare a unei sarcini (GET /tasks/create)
- tasks.store: pentru salvarea unei noi sarcini (POST /tasks)
- tasks.show: pentru afisarea detaliilor unei sarcini pe baza ID-ului (GET /tasks/{id})
- tasks.edit: pentru afisarea formularului de editare a unei sarcini (GET /tasks/{id}/edit)
- tasks.update: pentru actualizarea unei sarcini existente (PUT/PATCH /tasks/{id})
- tasks.destroy: pentru stergerea unei sarcini (DELETE /tasks/{id})

Am verificat rutele create cu ajutorul comenzii `php artisan route:list`

```
MacBook-Pro-EUGENIU:todo-app eugeniu$ php artisan route:list

  GET|HEAD        / ..................................... HomeController@index
  POST            _ignition/execute-solution ignition.executeSolution › Spatie\Larave…
  GET|HEAD        _ignition/health-check ignition.healthCheck › Spatie\LaravelIgnitio…
  POST            _ignition/update-config ignition.updateConfig › Spatie\LaravelIgnit…
  GET|HEAD        about ................................. HomeController@about
  GET|HEAD        api/user ..............................................
  GET|HEAD        sanctum/csrf-cookie sanctum.csrf-cookie › Laravel\Sanctum › CsrfCoo…
  GET|HEAD        tasks ........................ tasks.index › TaskController@index
  POST            tasks ....................... tasks.store › TaskController@store
  GET|HEAD        tasks/create ................. tasks.create › TaskController@create
  GET|HEAD        tasks/{id} ..................... tasks.show › TaskController@show
  PUT|PATCH       tasks/{id} .................. tasks.update › TaskController@update
  DELETE          tasks/{id} ................. tasks.destroy › TaskController@destroy
  GET|HEAD        tasks/{id}/edit ................. tasks.edit › TaskController@edit

                                                          Showing [14] routes
```

Sarcina nr.4. Șablonizarea folosind Blade

4.1. Crearea unui layout pentru pagini

Am creat un `layout` pentru paginile principale cu urmatoarele elemente comune ale paginii :

1. Titlul paginii;
2. Meniu de navigare;
3. Conținutul paginii;

Am folosit directiva `@yield` pentru a defini zona in care va fi inserat continutul paginii

4.2. Utilizarea șabloanelor Blade

Am creat un layout pentru paginile principale layouts/app.blade.php cu urmatoarele elemente :

1. Titlul paginii;
2. Meniu de navigare;
3. Conținutul paginii.

Am folosit directiva @yielda pentru a defini zona in care va fi inserat continutul diferitor pagini .

```blade
<!DOCTYPE html>
<html lang="ro">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@yield('title', 'To-Do App')</title>
</head>
<body>
    <nav>
        <ul>
            <li><a href="{{ route('home') }}">Acasă</a></li>
            <li><a href="{{ route('about') }}">Despre noi</a></li>
            <li><a href="{{ route('tasks.index') }}">Lista de sarcini</a></li>
            <li><a href="{{ route('tasks.create') }}">Crearea unei sarcini</a>
            </li>
        </ul>
    </nav>
    <div>
        @yield('content')
    </div>
</body>
</html>
```