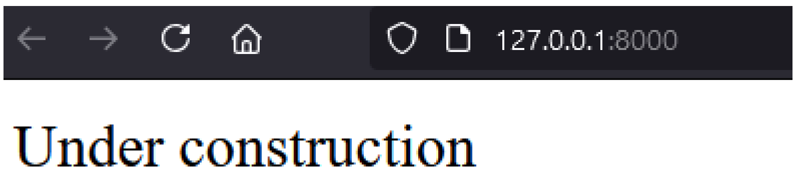
PHP Dossier

Práctica 01

> 🗁 Modificar el fichero web.php para que las peticiones GET (parecido al ejemplo anterior) al raíz de la aplicación: "/" muestren un mensaje que diga: "Under construction

```
  Route::get('/', function () {
     echo "Under construction";
 });
```

- Captura:



127.0.0.1:8000

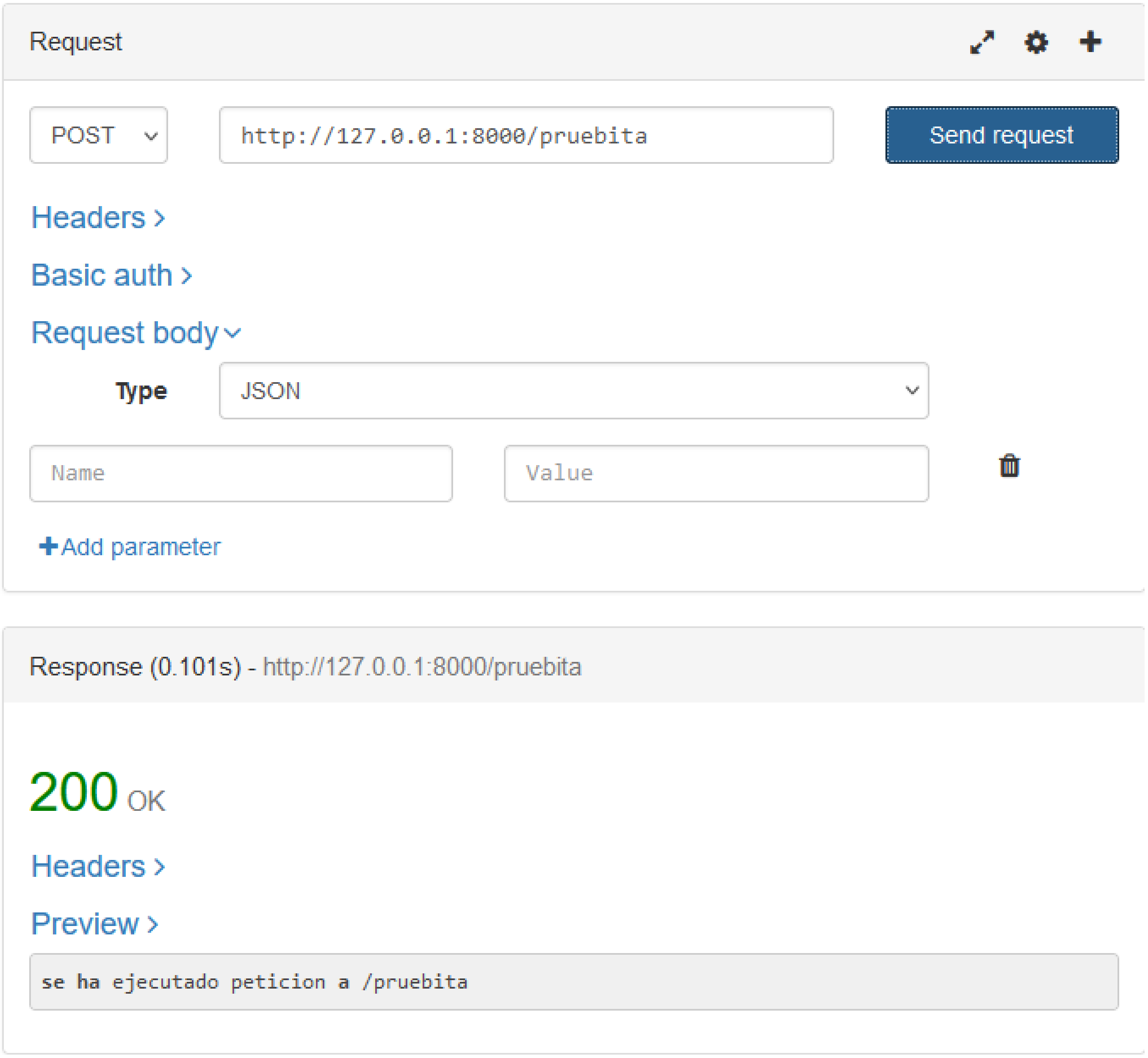
Under construction

Práctica 02

> 🗁 Modificar el fichero web.php para que las peticiones POST a: /pruebita muestren el mensaje: “se ha ejecutado una petición POST a la dirección: /pruebita ” Probar a hacer la petición POST ¿ muestra lo solicitado ? ¿ qué ocurre si se hace mediante una petición GET ? Volver a reestablecer la protección CSRF y hacer de nuevo la petición POST ¿ qué muestra ahora ?

```
Route::post('/pruebita', function () {
    echo "Se ha ejecutado una petición POST a la dirección: /pruebita";
});
```

- Captura:



http://127.0.0.1:8000/pruebita

Response (0.101s) - http://127.0.0.1:8000/pruebita

200 OK

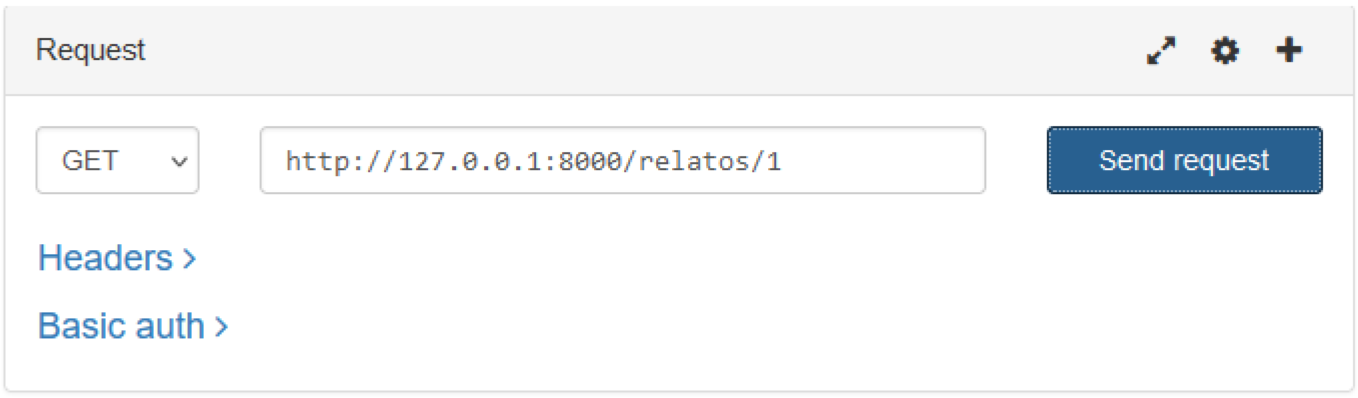
```
se ha ejecutado peticion a /pruebita
```

Práctica 03

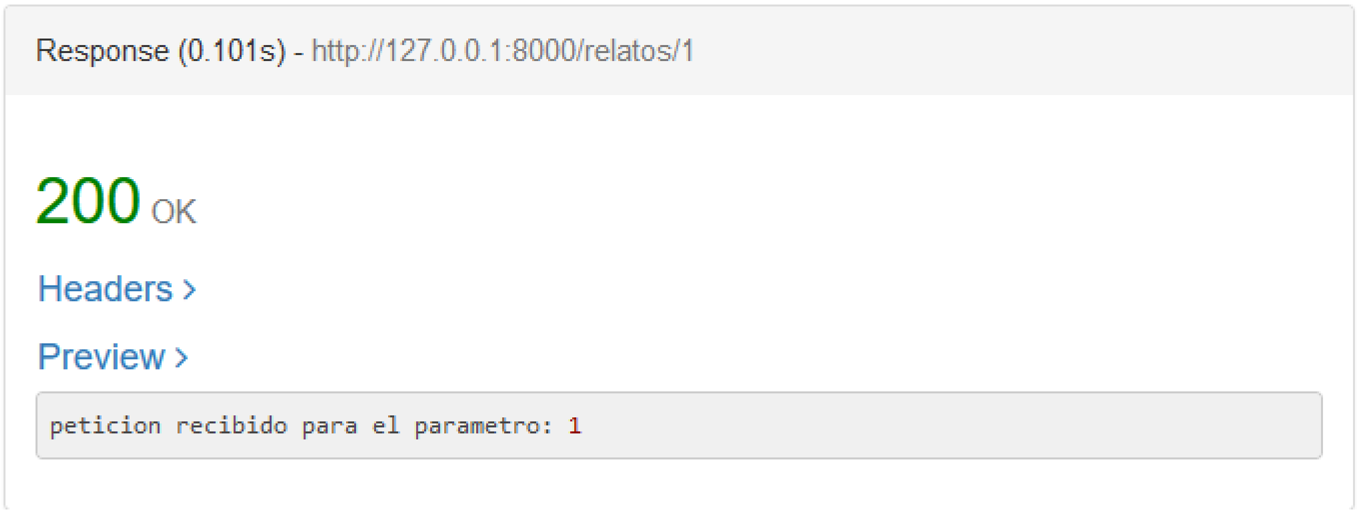
> 🗁 Crear una ruta para TODA petición (ya sea GET, POST, ...) hacia /relatos/numero (recordar que hemos visto una opción para recoger todo tipo de petición) De tal forma que numero deba ser un número y muestre el mensaje: “petición recibida para el parámetro: numero”

```
Route::any('/relatos/numeros/{num}', function ($num) {
    echo "Petición recivida para el parámetro: ". $num;
    exit();
})->where('num', '[0-9]+');
```

- Captura:

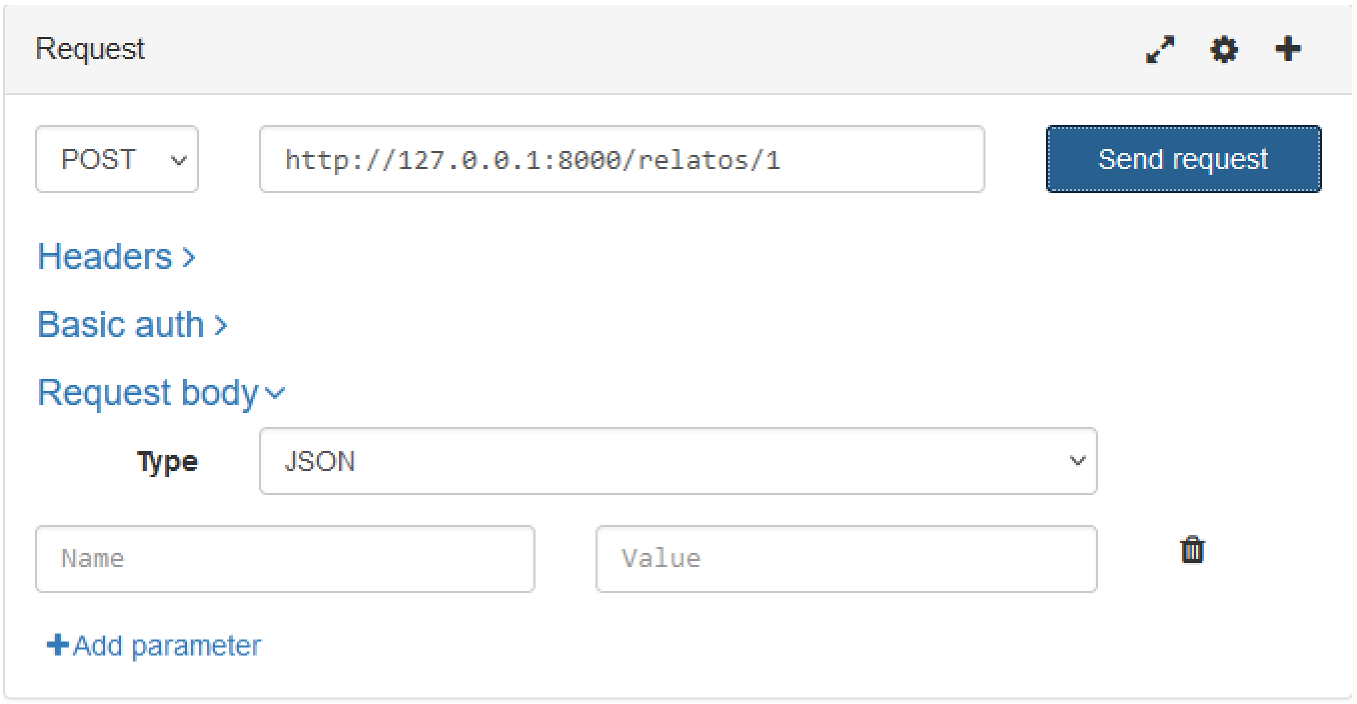

Request

GET http://127.0.0.1:8000/relatos/1


Response (0.101s) - http://127.0.0.1:8000/relatos/1

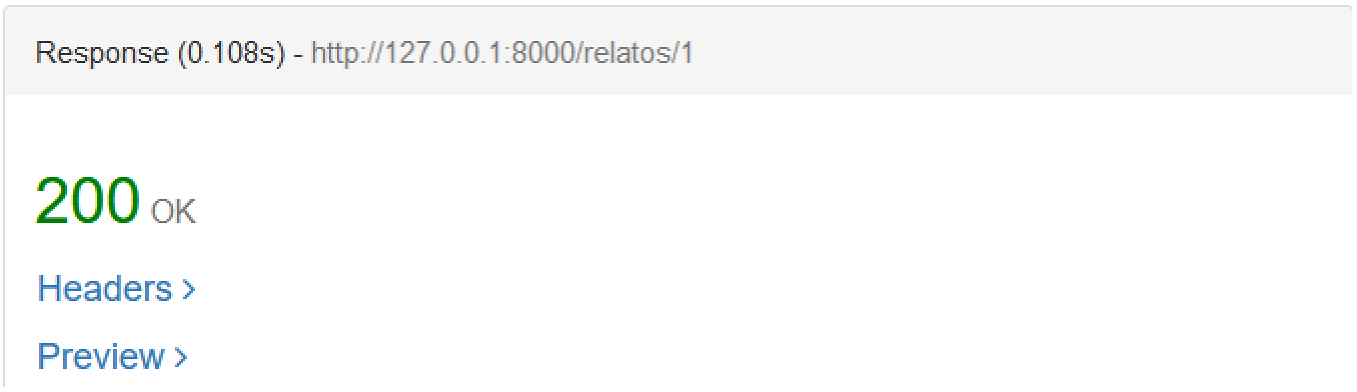
200 OK

```
peticion recibido para el parametro: 1
```


Request

POST http://127.0.0.1:8000/relatos/1

Type: JSON


Response (0.108s) - http://127.0.0.1:8000/relatos/1

200 OK

```
peticion recibido para el parametro: 1
```

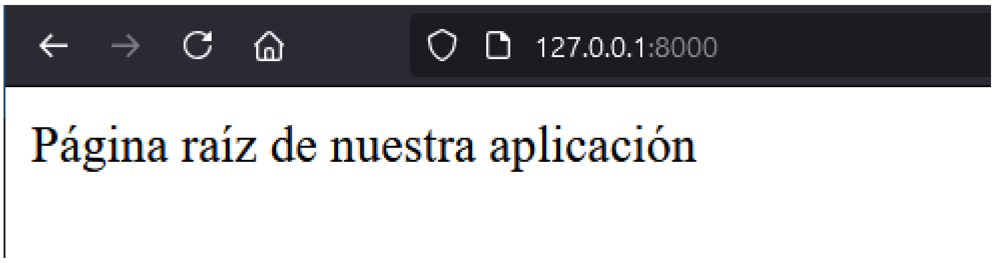
Práctica 04

> 🗁 Crear una ruta para el raíz: "/" En una primera implementación mostrará el mensaje: "página raíz de nuestra aplicación" que se resolverá en el propio web.php Haremos una segunda versión de esta actividad en la que redireccionará hacia el controlador y la función pertinente y allí se mostrará un mensaje que indique adicionalmente que se ha respondido desde el controlador

- Version 1:

```
Route::get('/', function (){
    echo "Página raíz de nuestra aplicación";
});
```

- Captura:



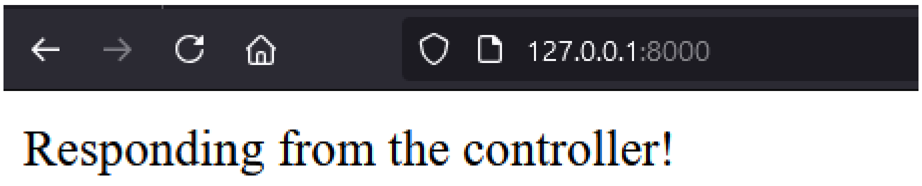
Página raíz de nuestra aplicación

- Version 2:

```
Route::get('/', [Practice04Controller::class, 'controllerResponse']);
```

```
class Practice04Controller extends Controller
{
    function controllerResponse(){
        echo "Responding from the controller!";
    }
}
```

- Captura:


127.0.0.1:8000

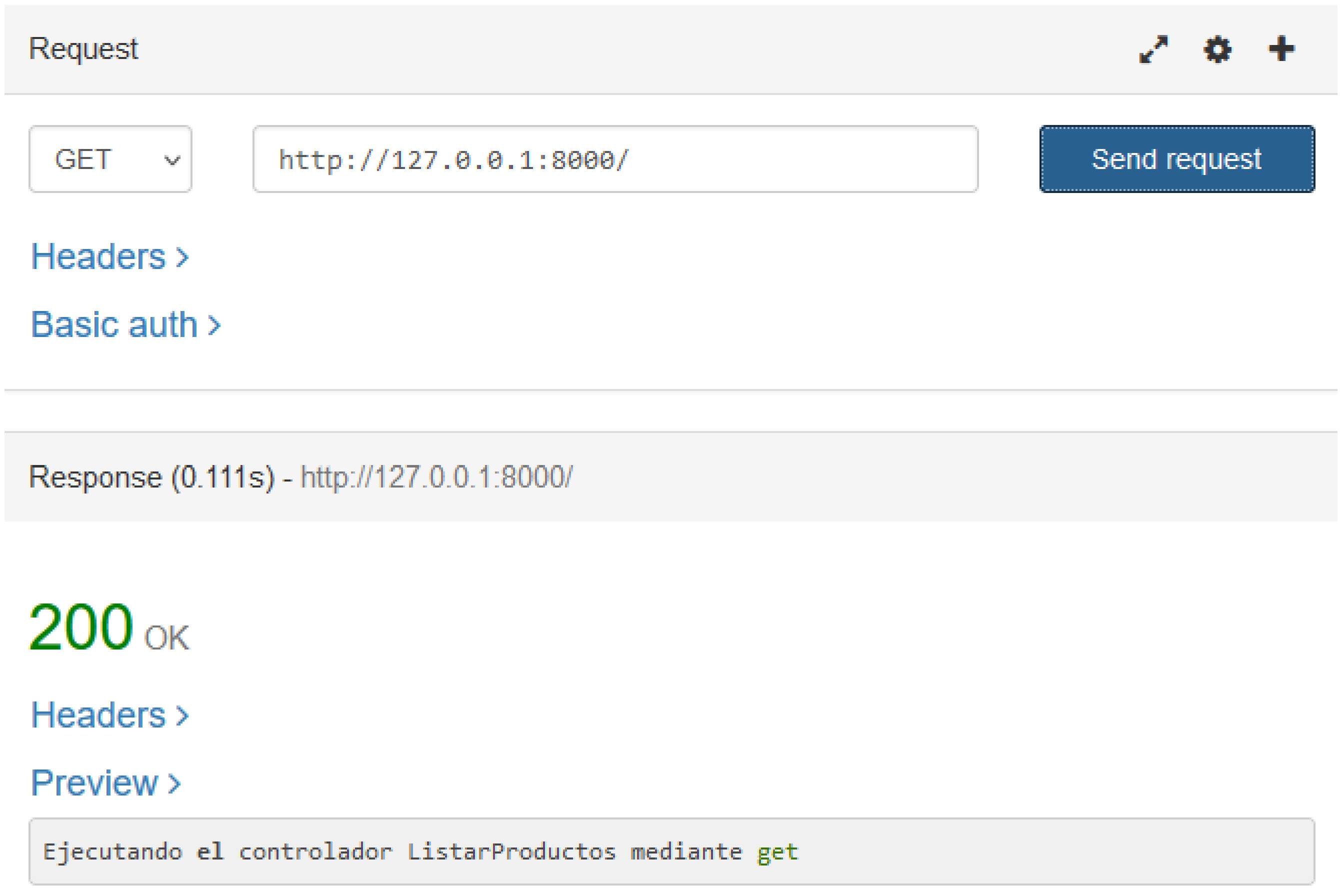
Responding from the controller!

Práctica 05

> 🗁 Crear un controlador llamado: ListarProductos que sea redireccionado en web.php cuando se acceda al raíz: “/” y muestre un mensaje que diga: “Ejecutando el controlador ListarProductos mediante get”. (si la llamada fue get. En el caso de que la llamada fuera post deberá decirlo)

```
Route::any('/', [ListarProductos::class, 'index']);

class ListarProductos extends Controller
{
    public function index(Request $request) {
        if ($request->isMethod('GET')){
            echo "Ejecutando el controlador ListarProductos mediante get";
        } elseif ($request->isMethod('POST')) {
            echo "Ejecutando el controlador ListarProductos mediante POST";
        }
    }
}
```

- Captura:



Request

GET `http://127.0.0.1:8000/` Send request

Headers >

Basic auth >

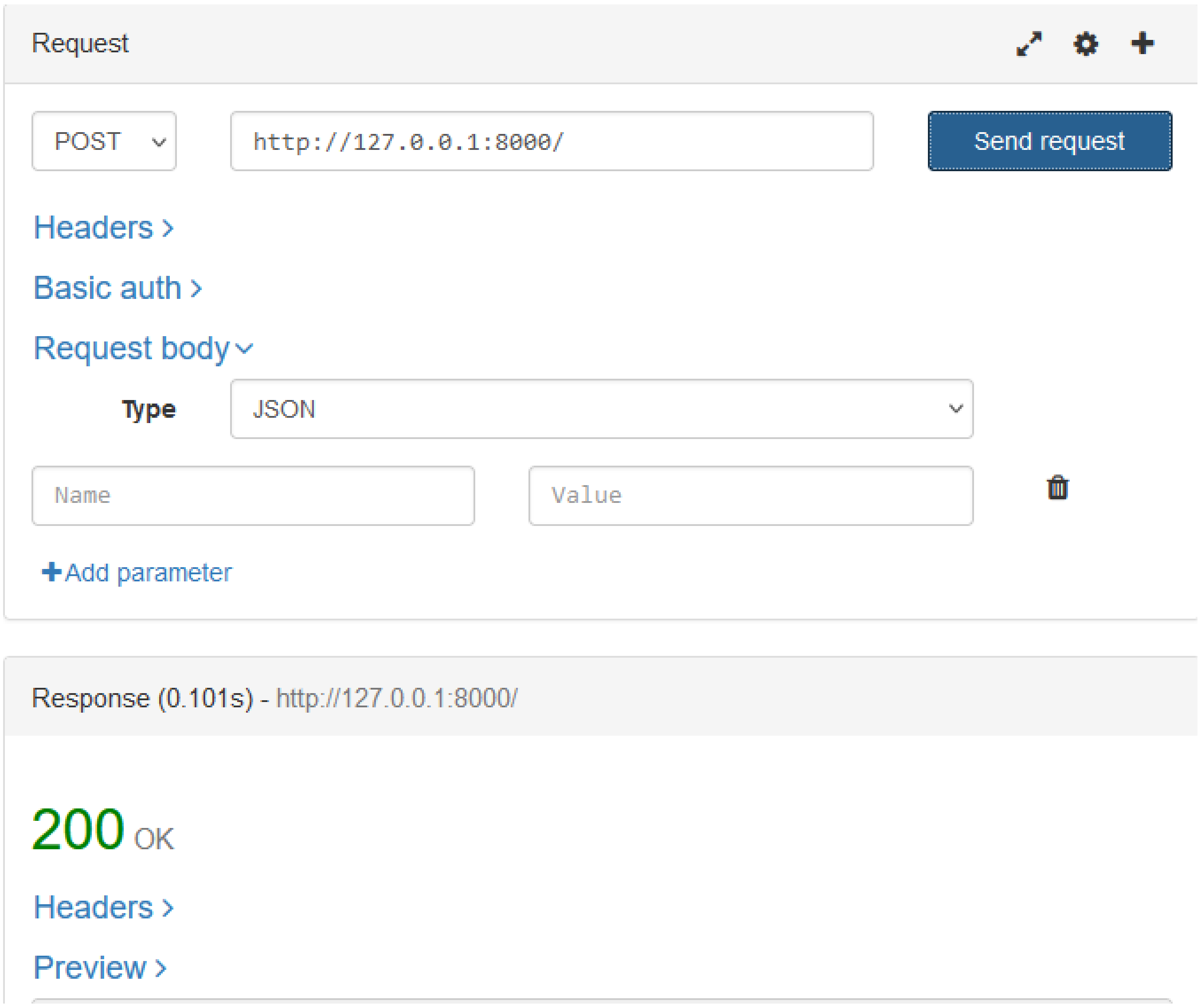
Response (0.111s) - http://127.0.0.1:8000/

200 OK

Headers >

Preview >

```
Ejecutando el controlador ListarProductos mediante get
```



Request

POST `http://127.0.0.1:8000/` Send request

Headers >

Basic auth >

Request body

Type: JSON

Name | Value

+Add parameter

Response (0.101s) - http://127.0.0.1:8000/

200 OK

Headers >

Preview >

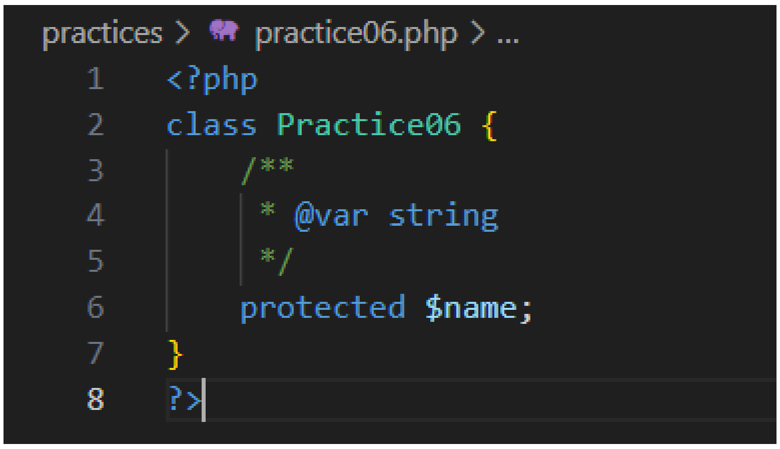
```
Ejecutando el controlador ListarProductos mediante POST
```

Práctica 06

> 🗁 Comprobar que la anotación @var para un objeto permite que el ide con inteliphense nos ayude con los atributos y los métodos

```
<?php
class Practice06 {
    /**
     * @var string
     */
    protected $name;
}
?>
```

- Captura:



practices > practice06.php > ...

```
<?php
class Practice06 {
    /**
     * @var string
     */
    protected $name;
}
?>
```

Práctica 07

> 🗁 Reproducir la vista descrita. Crear una tabla html por cada primo con un encabezado en la tabla que nos diga que campo estamos visualizando.

- Routes:

```
Route::any('/practice07', [Practice07Controller::class, 'primeNums']);
```

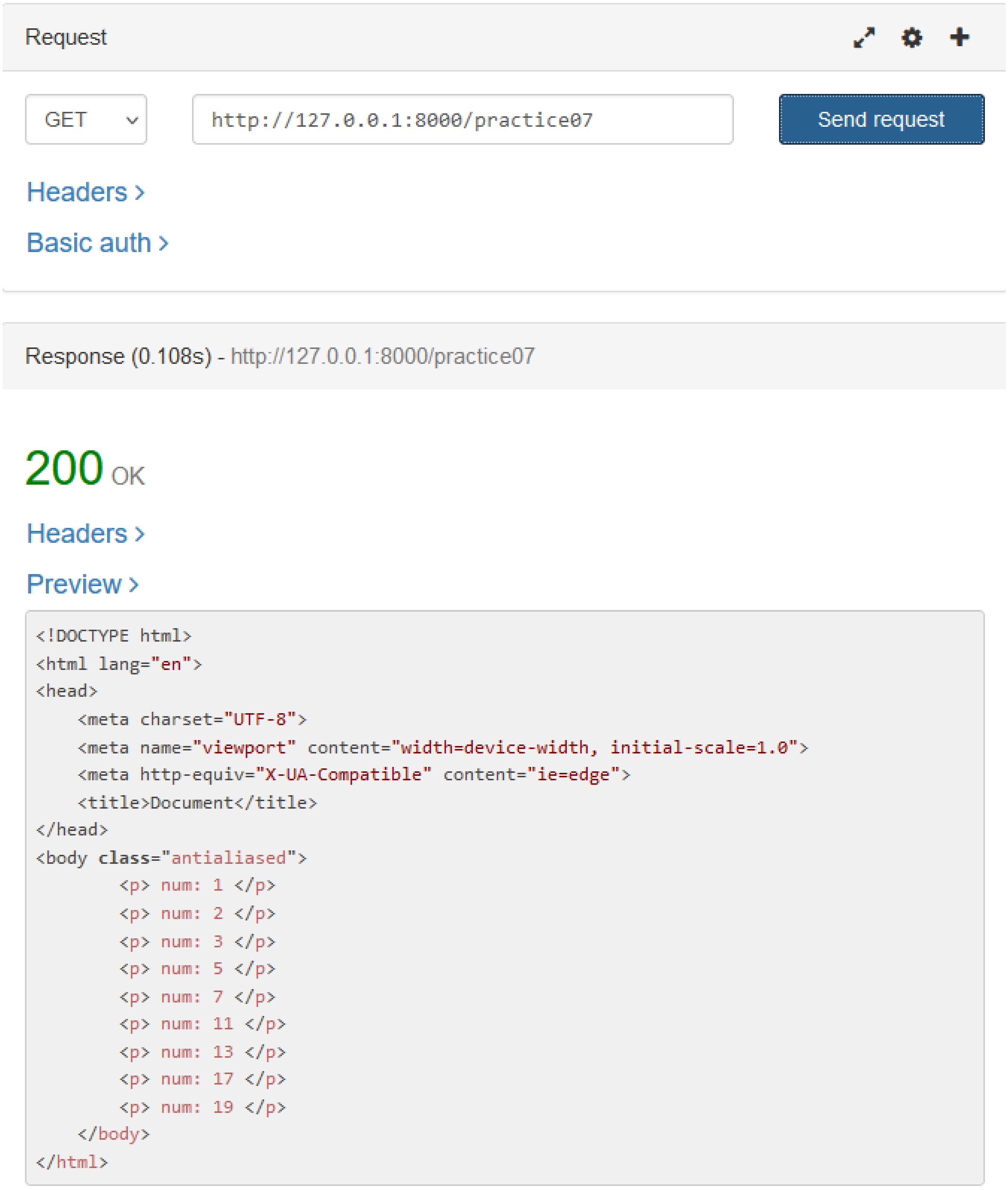
```
class Practice07Controller extends Controller
{
    public function primeNums()
    {
```

```
    $primeArray = collect([1,2,3,5,7,11,13,17,19]);
    return view('practice07',compact('primeArray'));
    }
}
```

- View:

```
<!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">
    <title>Document</title>
</head>
<body class="antialiased">
    @foreach ($primeArray as $num)
    <p> num: {{$num}} </p>
    @endforeach
</body>
</html>
```

- Captura:



```
http://127.0.0.1:8000/practice07
```

Response (0.108s) - http://127.0.0.1:8000/practice07

200 OK

```
<!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">
    <title>Document</title>
</head>
<body class="antialiased">
        <p> num: 1 </p>
        <p> num: 2 </p>
        <p> num: 3 </p>
        <p> num: 5 </p>
        <p> num: 7 </p>
        <p> num: 11 </p>
        <p> num: 13 </p>
        <p> num: 17 </p>
        <p> num: 19 </p>
    </body>
</html>
```

Práctica 08

> 🗁 Agregar al comienzo de la vista el mensaje(sustituye por la hora/día actual): Son las: 17:53 del día: 29-11-2020 Nota: buscar información y usar la función PHP date()

- Routes:

```
Route::get('/practice08', [Practice08Controller::class, 'date']);
```

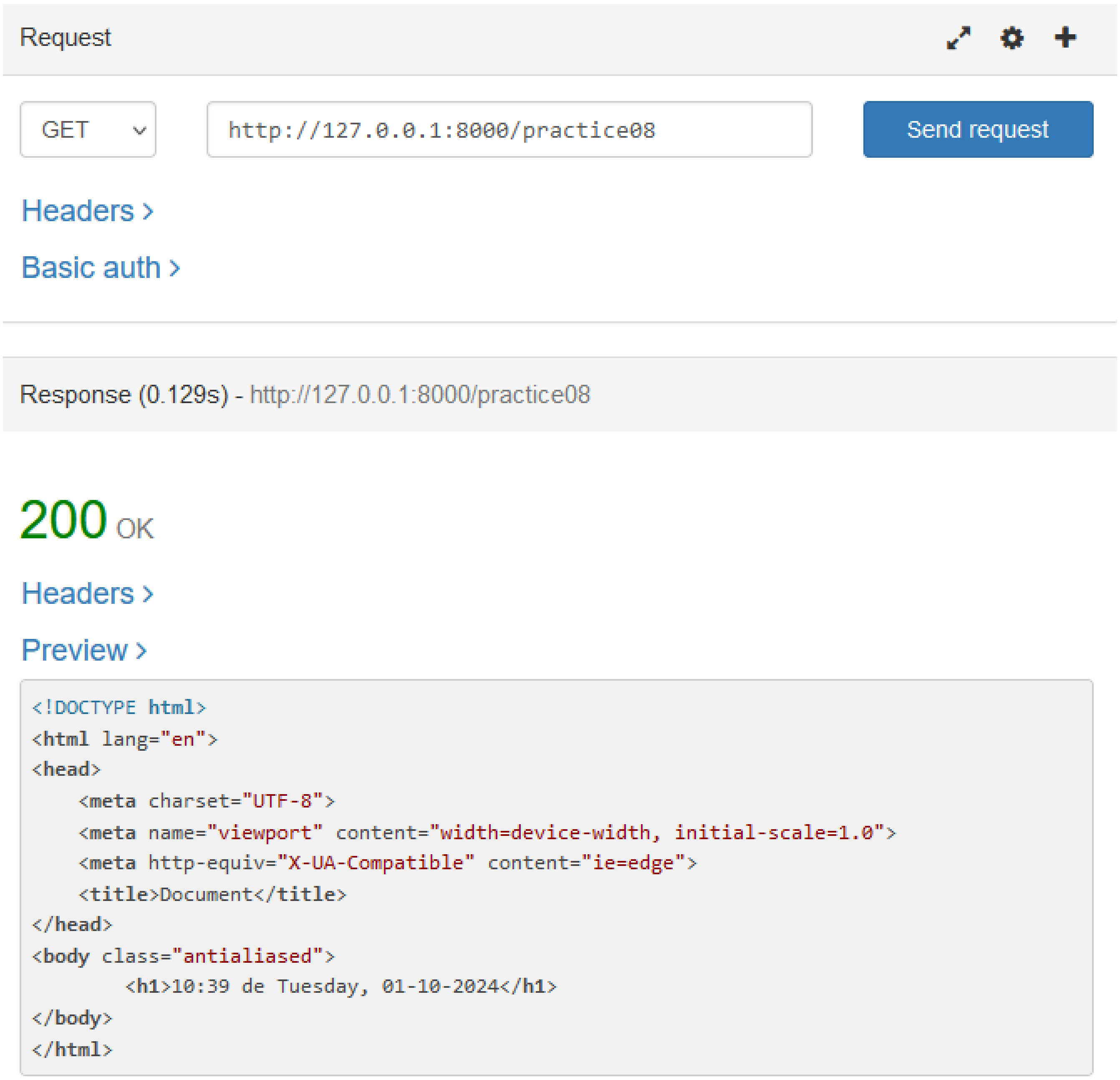
- Controller:

```
class Practice08Controller extends Controller
{
    public function date() {
        $currentDateTime = date('H:i \d\e l, d-m-Y');
        return view('practice08', compact('currentDateTime'));
    }
}
```

- View:

```
<!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">
    <title>Document</title>
</head>
<body class="antialiased">
        <h1>{{$currentDateTime}}</h1>
</body>
</html>
```

- Captura:



```
http://127.0.0.1:8000/practice08
```

Response (0.129s) - http://127.0.0.1:8000/practice08

200 OK

```html
<!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">
    <title>Document</title>
</head>
<body class="antialiased">
        <h1>10:39 de Tuesday, 01-10-2024</h1>
</body>
</html>
```

Práctica 09

> 🗁 El comando sleep() en php permite pausar la ejecución la cantidad de segundo especificada como parámetro. Modificar el ejemplo anterior para que lo muestre 3 veces con una espera de 1 segundo entre una iteración y la siguiente, mostrando de forma actualizada la información de los segundos desde 1970

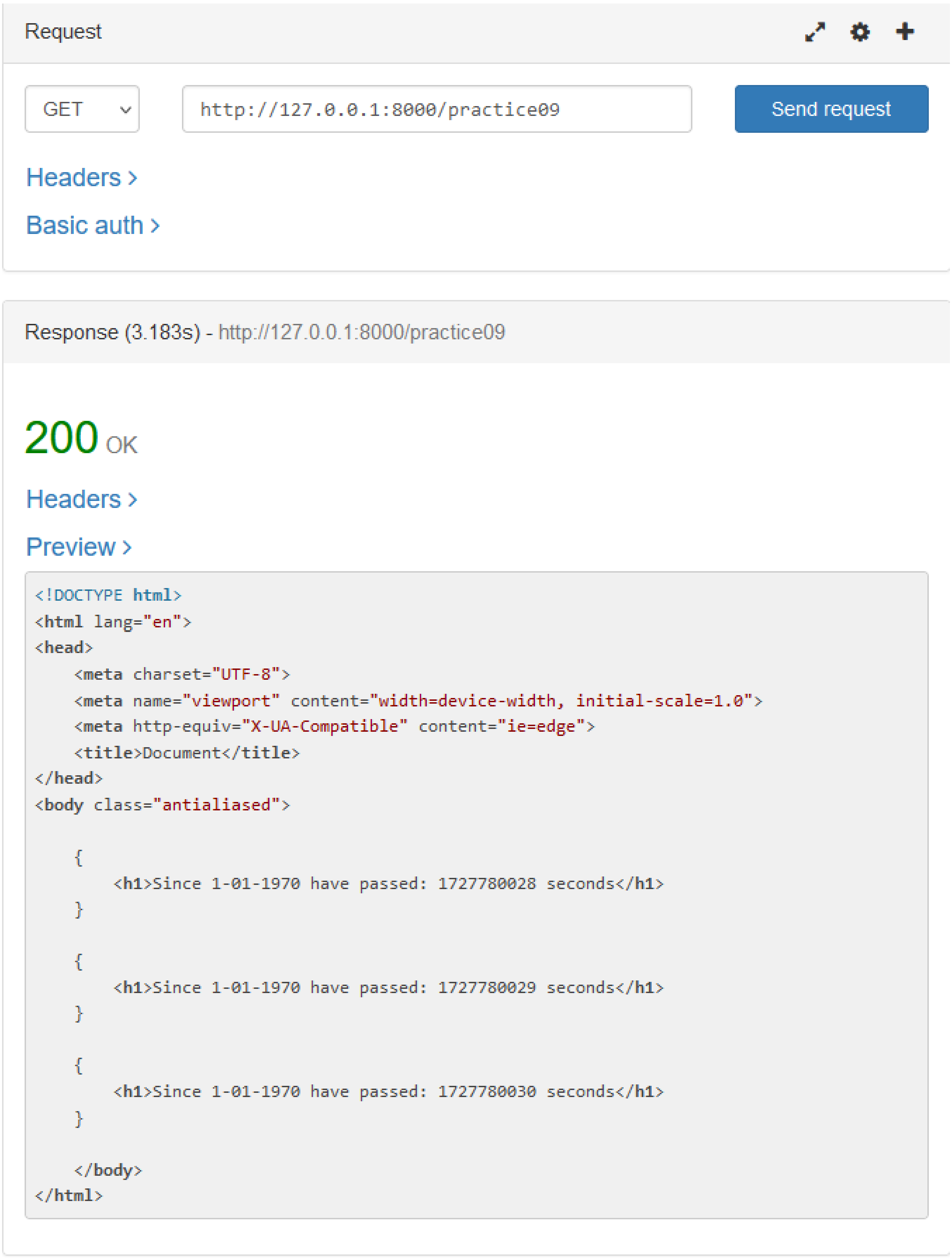
- Routes:

```
Route::get('/practice09', function(){
    return view('practice09');
});
```

- View:

```
    @php
        $dataSecondsArray = [];

        for($i=0; $i < 3; $i++){
            $seconds = time();
            $dataSecondsArray[$i] = $seconds;
            sleep(1);
        }

    @endphp

    @foreach ($dataSecondsArray as $seconds ){
        <h1>Since 1-01-1970 have passed: {{$seconds}} seconds</h1>
    }

    @endforeach
```

- Captura:


Request
GET
http://127.0.0.1:8000/practice09
Send request
Headers >
Basic auth >
Response (3.183s) - http://127.0.0.1:8000/practice09
200 OK
Headers >
Preview >

```
<!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">
    <title>Document</title>
</head>
<body class="antialiased">

    {
        <h1>Since 1-01-1970 have passed: 1727780028 seconds</h1>
    }

    {
        <h1>Since 1-01-1970 have passed: 1727780029 seconds</h1>
    }

    {
        <h1>Since 1-01-1970 have passed: 1727780030 seconds</h1>
    }

    </body>
</html>
```

Práctica 10

> 🗁 Generar una lista de números aleatorios de 0 a100 en el controlador. Desde nuestra plantilla blade mostraremos primero la lista de números obtenidos menores de 50 y un poco más abajo en la página

> los mayores que 50. Hacer uso de las directivas @if para que al mostrar aquellos que sean mayores de 50

- Routes:

```
Route::get('/practice10', [Practice10Controller::class, 'rndNum']);
```

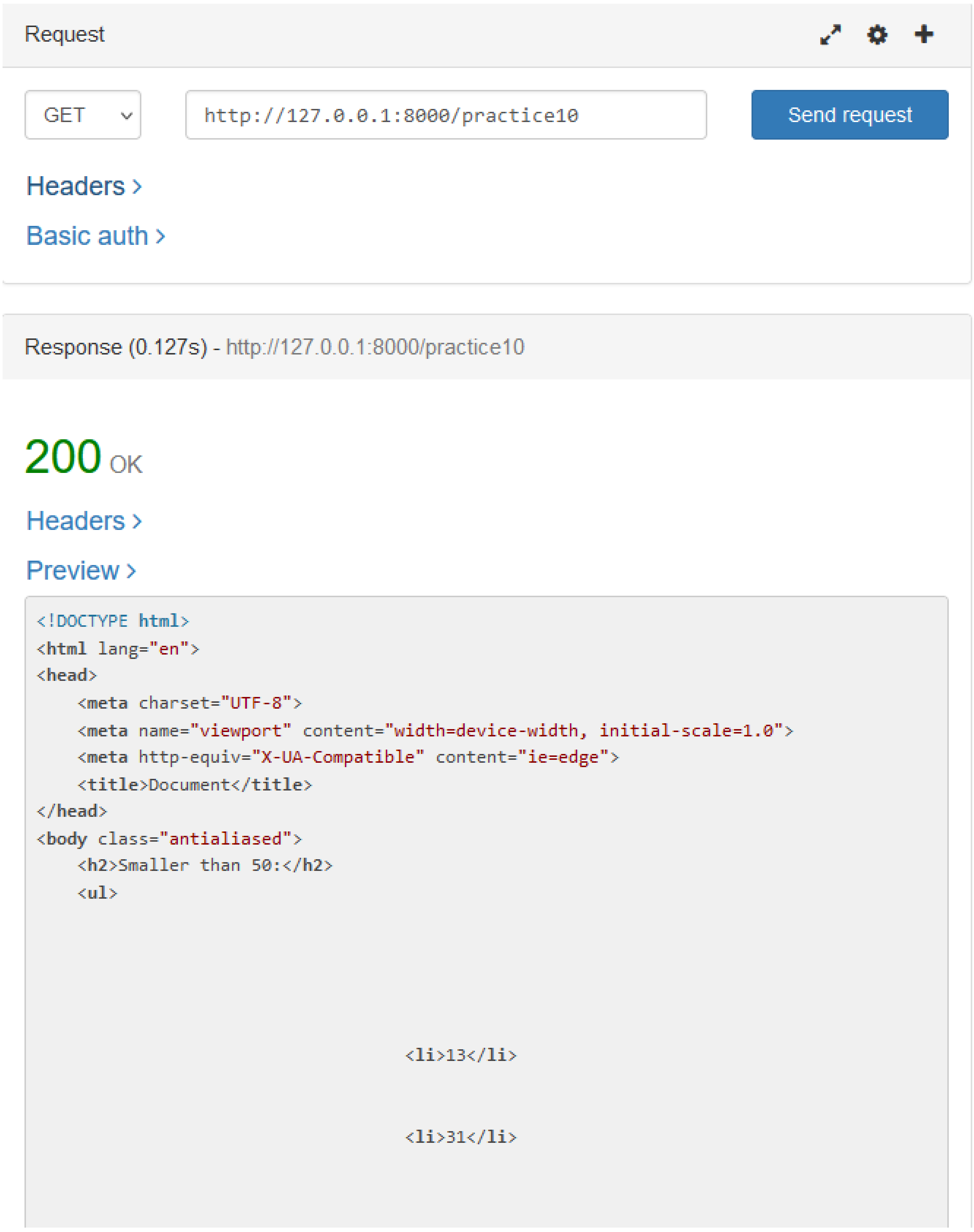
- Controller

```
    public function rndNum() {
        $array = [];
        for ($i = 0; $i < 15; $i++) {
            $array[] = rand(1, 100);
        }
        return view('practice10', compact('array'));
    }
```

- View:

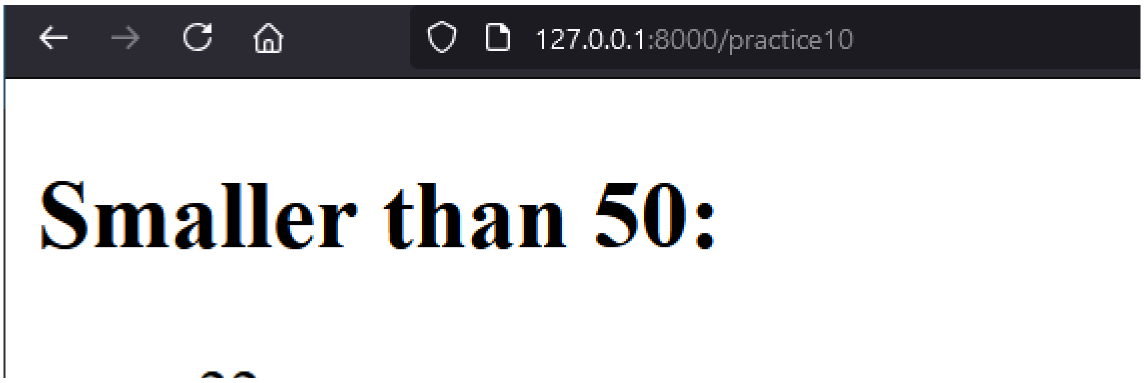
```
    <h2>Smaller than 50:</h2>
    <ul>
        @foreach ($array as $num )
            @if ($num < 50)
                <li>{{$num}}</li>

            @endif

        @endforeach
    </ul>

    <h2>Greater than 50:</h2>
    <ul>
        @foreach ($array as $num )
            @if ($num > 50)
                <li>{{$num}}</li>

            @endif

        @endforeach
    </ul>
```

- Captura:

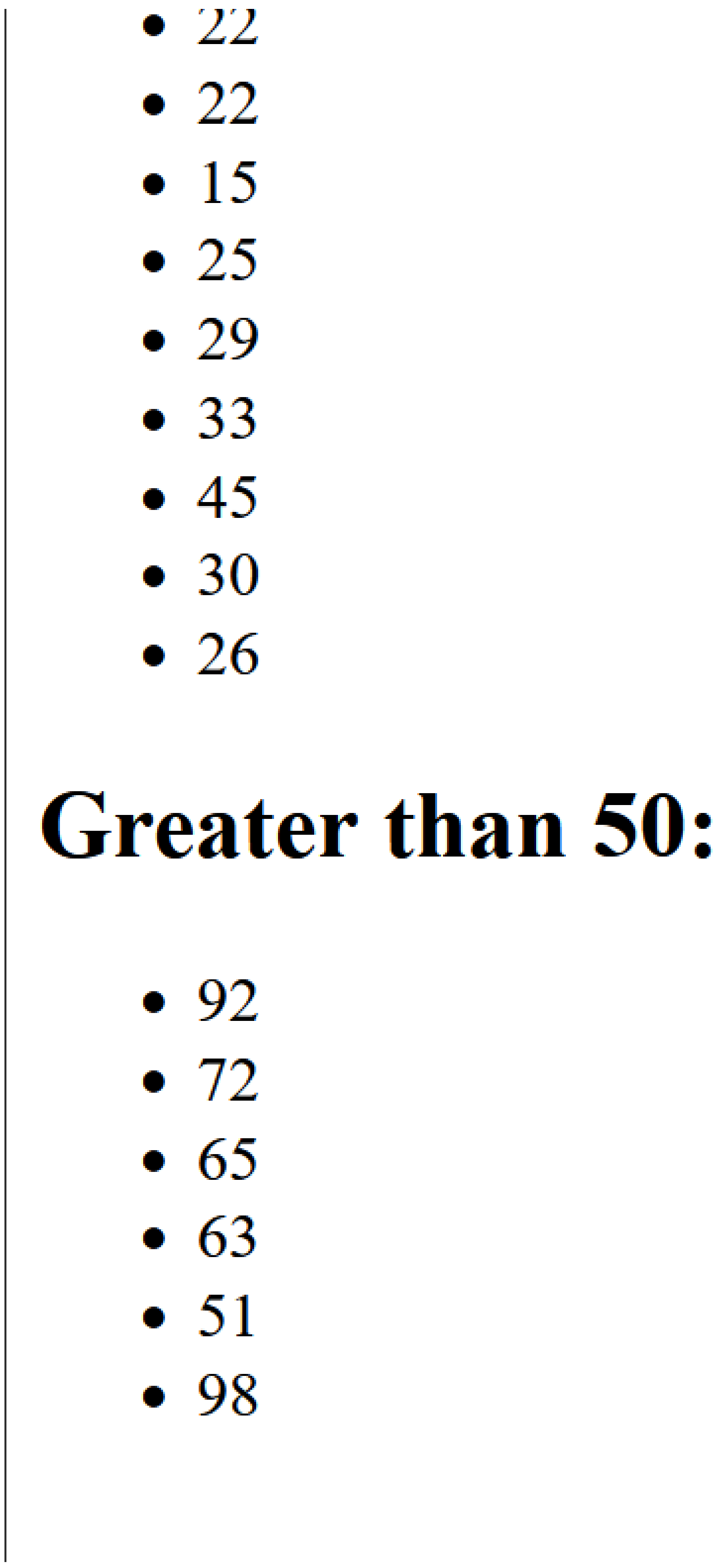


http://127.0.0.1:8000/practice10

```
<!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">
    <title>Document</title>
</head>
<body class="antialiased">
    <h2>Smaller than 50:</h2>
    <ul>



                                        <li>13</li>


                                        <li>31</li>

```



Smaller than 50:



- 22
- 22
- 15
- 25
- 29
- 33
- 45
- 30
- 26

Greater than 50:

- 92
- 72
- 65
- 63
- 51
- 98

Práctica 11

> 🗁 Enviar en un textarea una lista de palabras separadas por comas. Mostrar en una lista html esas palabras recibidas (una palabra por cada
>
> - de la lista) convertidas todas a mayúsculas. Para ello se usará el bucle: @for (cuidado! No el foreach) Observar que eso implicará "contar" el número de elementos que tiene la colección de palabras

- Routes:

```
Route::get('/processwords', [Practice11Controller::class, 'processWords']);
```

- Controller

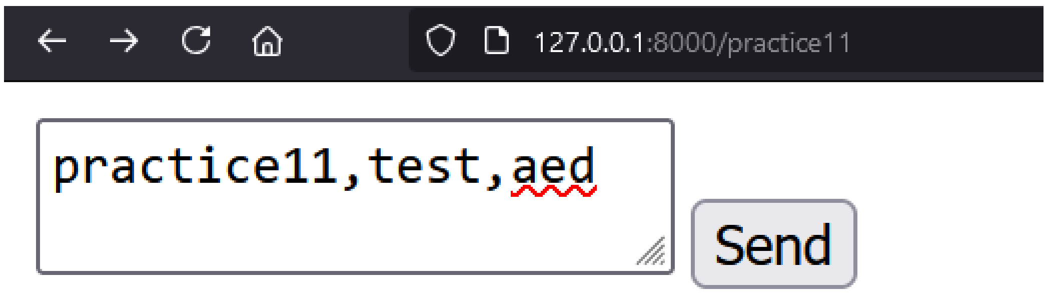
```
class Practice11Controller extends Controller
{
    public function processWords(Request $request) {
        $words = explode(',',  $request->input('words')??null);

        return view('practice11result', [
            'words' => $words,
        ]);
    }
}
```

- Views:

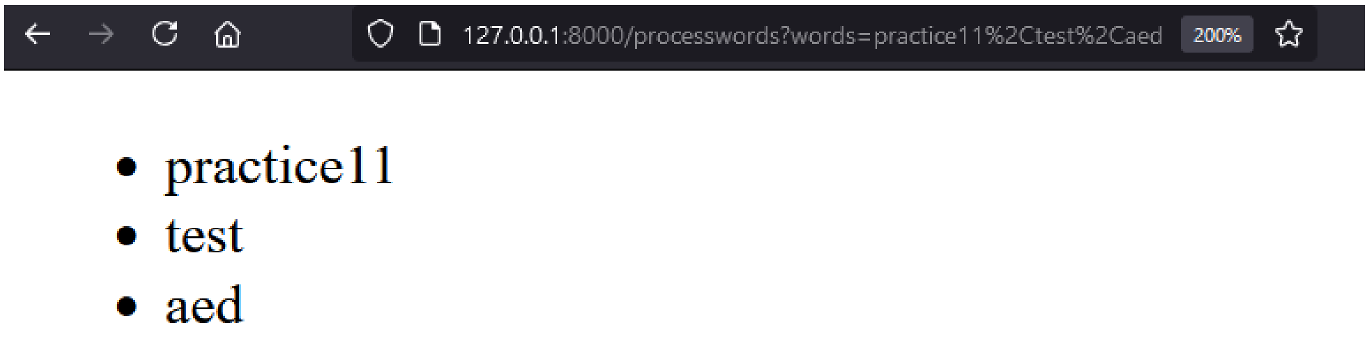
```
    <form action="processwords" method="GET">
        <textarea name="words" placeholder="Word's list">ç</textarea>
        <input type="submit" value="Send">
    </form>
```

```
    @php
    $length = count($words);
    @endphp

    <ul>
        @for ($i=0; $i<$length; $i++)
            <li>{{$words[$i]}}</li>
        @endfor
    </ul>
```

- Captura:


127.0.0.1:8000/practice11

practice11,test,aed


127.0.0.1:8000/processwords?words=practice11%2Ctest%2Caed

- practice11
- test
- aed

Práctica 12

> 🗁 Ubicar imágenes en la carpeta descrita para las imágenes que quieras mostrar (mínimo 5). Hacer que se visualicen en el navegador las imágenes en nuestra vista

- Routes:

```
Route::get('/practice12', [Practice12Controller::class, 'showImgs']);
```

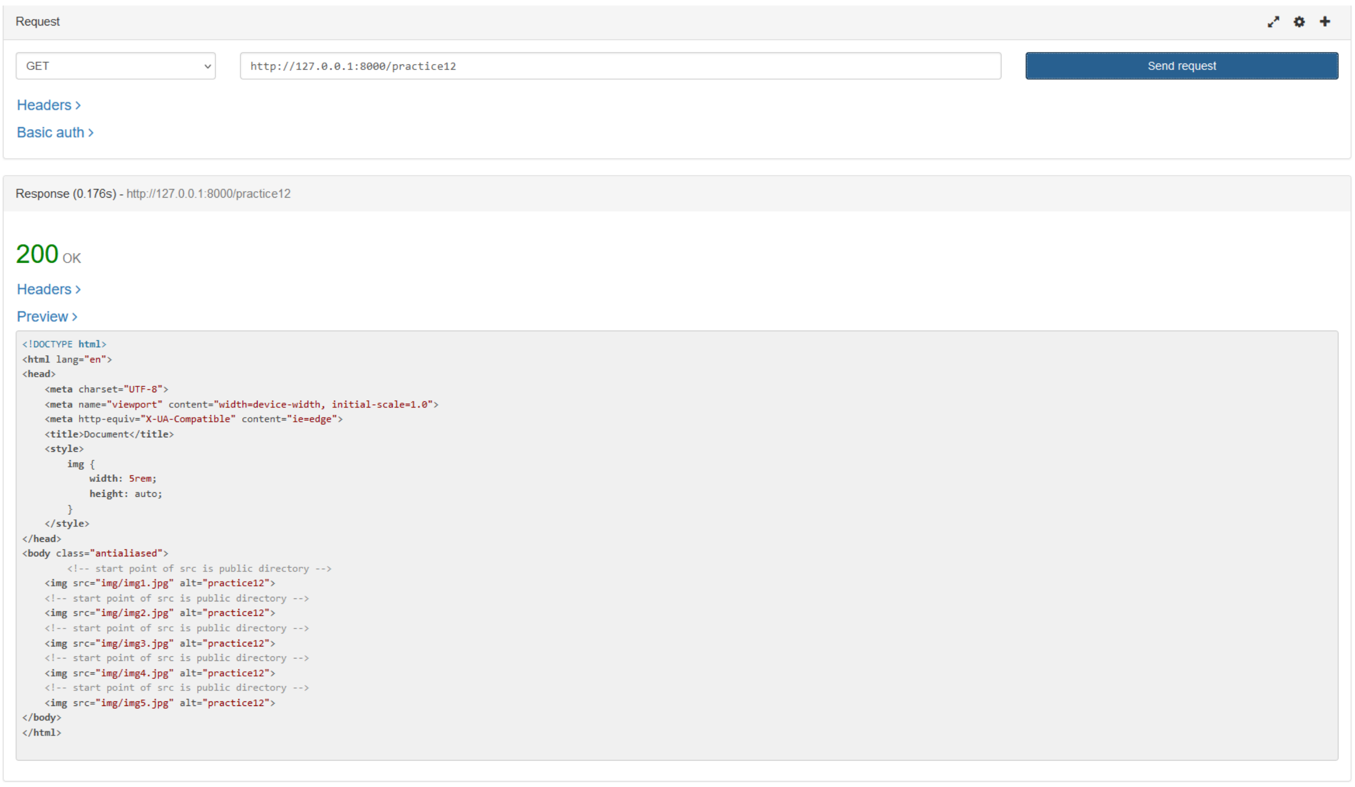
- Controller:

```
class Practice12Controller extends Controller
{
    function showImgs(){
        $imgArray =['img1.jpg', 'img2.jpg', 'img3.jpg', 'img4.jpg', 'img5.jpg'];

        return view('practice12', compact('imgArray' ));
    }
}
```

- View:

```
    @foreach ($imgArray as $img)
        <img src="img/{{$img}}" alt="practice12">
    @endforeach
```

- Captura:

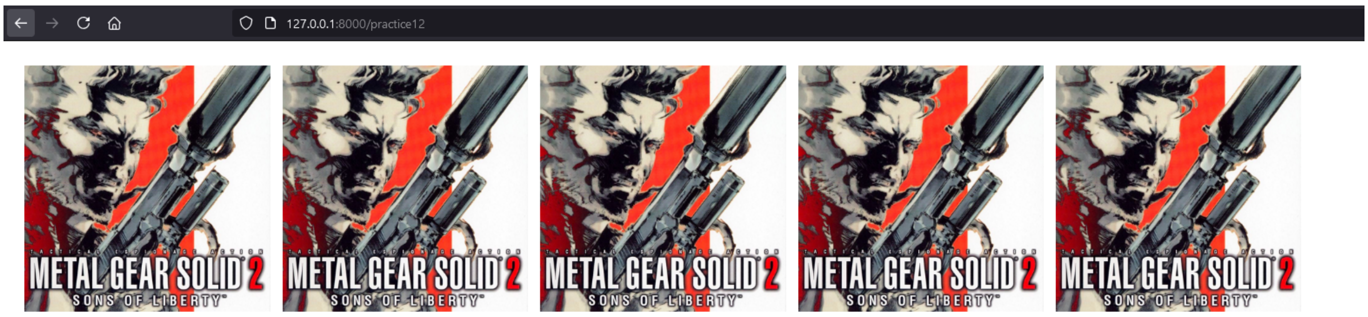


http://127.0.0.1:8000/practice12

Response (0.176s) - http://127.0.0.1:8000/practice12

200 OK

```
<!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">
    <title>Document</title>
    <style>
        img {
            width: 5rem;
            height: auto;
        }
    </style>
</head>
<body class="antialiased">
        <!-- start point of src is public directory -->
    <img src="img/img1.jpg" alt="practice12">
    <!-- start point of src is public directory -->
    <img src="img/img2.jpg" alt="practice12">
    <!-- start point of src is public directory -->
    <img src="img/img3.jpg" alt="practice12">
    <!-- start point of src is public directory -->
    <img src="img/img4.jpg" alt="practice12">
    <!-- start point of src is public directory -->
    <img src="img/img5.jpg" alt="practice12">
</body>
</html>
```



Práctica 13

> 🗁 Crear un formulario que envíe nombres de colores en cada ejecución del usuario. Obtendrá por respuesta una página con la lista de colores que ha ido introduciendo (usar session() para almacenar la lista de colores) (es un formulario post tendremos que tener en cuenta @csrf leer más abajo)

- Routes:

```
Route::get('/practice13', [Practice13Controller::class, 'getColors']);
Route::post('/add-color', [Practice13Controller::class, 'addColor']);
Route::post('/delete-color/{id}', [Practice13Controller::class, 'deleteColor']);
```

- Controller:

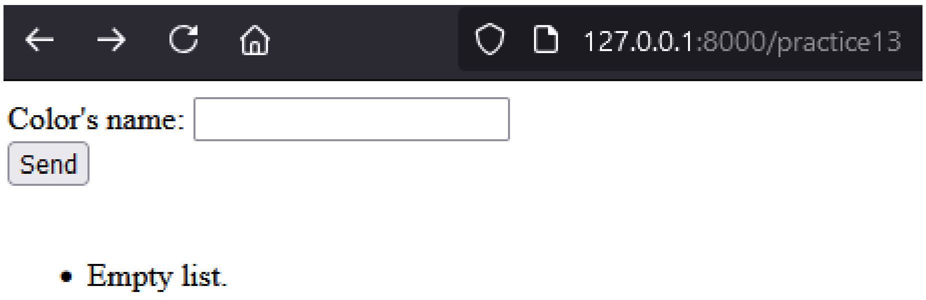
```
class Practice13Controller extends Controller
{
    public function getColors() {
        $colors = session()->get('colors', []);

        if(!isset($colors)){
            $colors = [];
            session()->put('colors', $colors);
        }

        return view('practice13', compact('colors'));
    }

    public function addColor(Request $request){
        $colors = session()->get('colors', []);

        $name = $request->input('color')??null;
        $id = count($colors) + 1;


        $newColor = new Color( $id, $name);
        $colors[] = $newColor;

        session()->put('colors', $colors);

        return redirect('/practice13');
    }

    public function deleteColor(Request $request){
        $colors = session()->get('colors', []);
        $id = $request->input('id');

        foreach($colors as $key => $item){
            if($item->getId() == $id){
                unset($colors[$key]);
                break;
            }
        }

        session()->put('colors', array_values($colors));
        return redirect('/practice13');
    }

}
```

- View:

```

     <form method="POST" action="{{ url('/add-color')}}">
          @csrf
          @if(isset($color))
              <input type="hidden" name="id" value="{{ $color->id }}">
          @endif
          <label for="color">Color's name: </label>
          <input type="text" name="color" id="color">
```

```
        <br>
        <input type="submit" name="submit" id="submit" value="Send">
    </form>
    <br>
    <div class="history">
        <ul>
            @if(!empty($colors))
                @foreach ($colors as $color)
                <li>
                    {{ $color->getName() }}
                    <form method="POST" action="{{ url('/delete-color/'.$color-
>id) }}">
                        @csrf
                        <input type="hidden" name="id" value="{{ $color->id }}">
                        <button type="submit">Delete</button>
                    </form>
                </li>
                @endforeach
            @else
                <li>Empty list.</li>
            @endif
        </ul>
    </div>
```

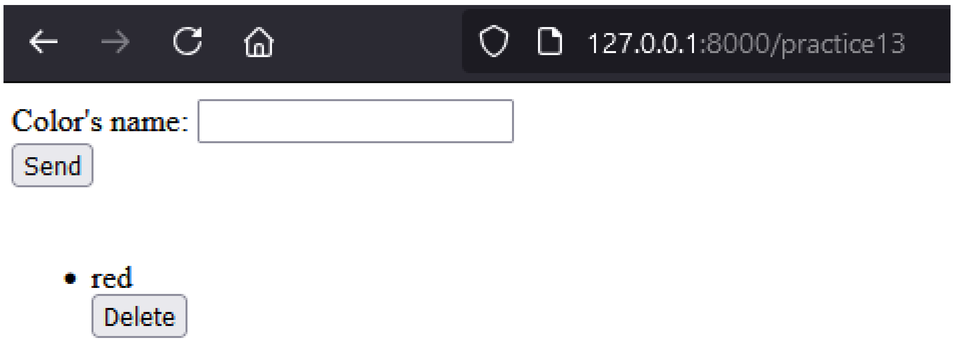
- Captura:


127.0.0.1:8000/practice13

Color's name:

Send

- Empty list.


127.0.0.1:8000/practice13

Color's name:

Send

- red

Delete

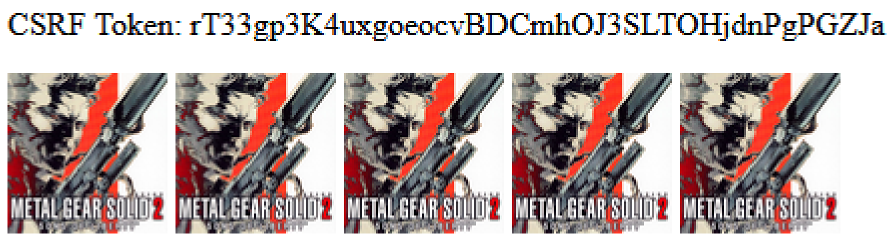
Práctica 14

> 🗁 Una forma fácil de visualizar el token csrf es mediante: {{ csrf_token() }} Introducir en la práctica 12 ese código y comprobar que está activo.

- View (práctica 12 modificada):

```
<body class="antialiased">
    <p>CSRF Token: {{ csrf_token() }}</p>
    @foreach ($imgArray as $img)
        <img src="img/{{$img}}" alt="practice12">
    @endforeach
</body>
```

- Captura:


CSRF Token: rT33gp3K4uxgoeocvBDCmhOJ3SLTOHjdnPgPGZJa

Práctica 15

> 🗁 Crear un formulario POST Con los datos de un posible usuario (nombre, edad, gustos, etc) En cada ejecución de este formulario se le muestra al usuario la información almacenada del usuario en session() Observar que si se envía el formulario sin rellenar algún campo, se mantendrá la información anterior respecto a ese campo

- Routes:

```
Route::get('/practice15', [Practice15Controller::class, 'showForm']);

Route::post('practice15/update', [Practice15Controller::class, 'handleForm']);
```

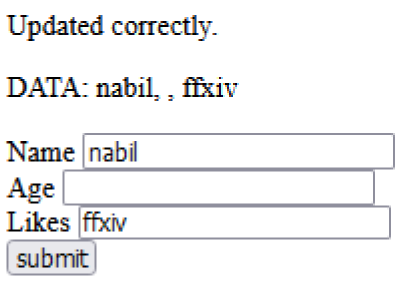
- Controller:

```
    public function showForm(Request $request) {

        $name = session()->get('name');
        $age = session()->get('age');
        $likes = session()->get('likes');

        return view('practice15', compact('name', 'age', 'likes'));
    }

    public function handleForm(Request $request)
    {
        $nameSession = session()->get('name', '');
        $ageSession = session()->get('age', '');
        $likesSession = session()->get('likes', '');

        $nameUpdate = $request->get('name', $nameSession);
        $ageUpdate = $request->get('age', $ageSession);
        $likesUpdate = $request->get('likes', $likesSession);

        $request->session()->put('name', $nameUpdate);
        $request->session()->put('age', $ageUpdate);
        $request->session()->put('likes', $likesUpdate);

        return redirect('practice15')->with('success', 'Updated correctly.');
    }
```

- View:

```
<div class="main-container">
    @if(session('success'))
        <p> {{session('success')}}</p>
    @endif

    <p>DATA: {{session('name')}}, {{session('age')}}, {{session('likes')}}</p>
    <form action="{{ url('/practice15/update')}}" method="POST">
        @csrf
            <label for="name">Name</label>
            <input type="text" name="name" id="name" value="{{session('name')}}"
/>
            <br>
            <label for="age">Age</label>
            <input type="text" name="age" id="age" value="{{session('age')}}" />
            <br>
            <label for="likes">Likes</label>
            <input type="text" name="likes" id="likes" value="
{{session('likes')}}" />
            <br>
            <input type="submit" name="submit" value="submit">
    </form>
    <br>
</div>
```

- Captura:


Updated correctly.

DATA: nabil, , ffxiv

Name nabil
Age
Likes ffxiv

Práctica 16

> 🗂 Crear un fichero con nombre y dirección de correo por fila (en formato csv) almacenado en Storage Leer el fichero y mostrarlo en pantalla

- Routes:

```
 Route::get('/practice16', [Practice16Controller::class, 'readCsv']);
```

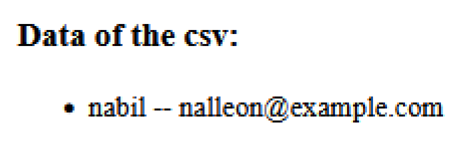
- Controller:

```
public function readCsv (){
    $csvContent = Storage::get('users.csv');
    $rows = explode("\n", $csvContent);
    $data = [];

    foreach ($rows as $row) {
        $columns = explode(',', $row);
        $data[] = [
            'name' => $columns[0],
            'email' => $columns[1],
        ];
    }

    return view('practice16', compact('data'));
}
```

- View:

```
<div class="main-container">

    <h3>Data of the csv:</h3>
    <ul>
        @foreach($data as $row)
        <li>
            {{ $row['name'] }} -- {{ $row['email'] }}
        </li>
    @endforeach
    </ul>
</div>
```

- Captura:



Data of the csv:

- nabil -- nalleon@example.com

Práctica 17

> 🗁 Crear un formulario que se introduzca un nombre y cree un directorio en storage con ese nombre

- Routes:

```
Route::get('/practice17', function (){
    return view('/practice17');
});

Route::post('/create-directory', [Practice17Controller::class 'createDirectory']);
```

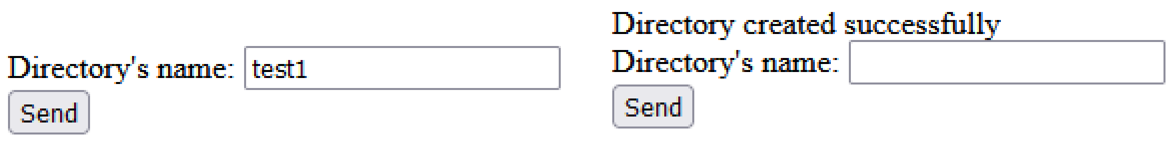
- Controller:

```
class Practice17Controller extends Controller
{
    public function createDirectory(Request $request){
        $directory = $request->input('directory');
        if($directory ==! null){
            Storage::makeDirectory('/' . $directory, 700, true);
            echo "Directory created successfully";
        }
        return view('/practice17');
    }

}
```

- View:

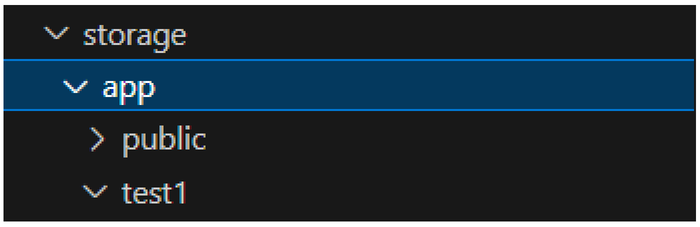
```
     <form method="POST" action="{{ url('/create-directory')}}">
        @csrf
        @if(isset($directory))
            <input type="hidden" name="id" value="{{ $color->id }}">
        @endif
        <label for="directory">Directory's name: </label>
        <input type="text" name="directory" id="directory">
        <br>
        <input type="submit" name="submit" id="submit" value="Send">
    </form>
    <br>
    <div class="history">
    </div>
```

- Captura:



Directory's name: test1
Send

Directory created successfully
Directory's name:
Send



Práctica 18

> 🗀 Crear un formulario que se introduzca un nombre y cree un directorio en storage con ese nombre

- Routes:

```
Route::get('/practice18', function (){
    return view('/practice18');
});

Route::post('/read-file', [Practice18Controller::class, 'readFile']);
```

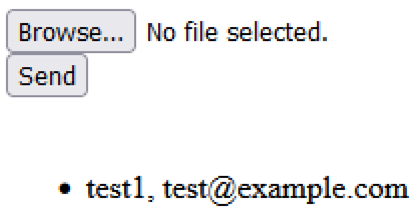
- Controller:

```
class Practice18Controller extends Controller
class Practice18Controller extends Controller
{
    public function readFile(Request $request) {
        if (!$request->hasFile('myFile')) {
            return back()->withErrors(['myFile' => 'No se ha subido ningún
archivo.']);
        }


        $file = $request->file('myFile');

        $content = [];

        $fileOriginalName = $file->getClientOriginalName();

        $path =$file->storeAs("/", $fileOriginalName);
        if (($open = fopen(storage_path('app/' . $path), "r")) !== FALSE) {
            while (($data = fgetcsv($open, 1000, ",")) !== FALSE) {
                $content[] = $data;
            }
            fclose($open);
        }

        return redirect('/practice18')->with('content', $content);
    }
}
```

- View:

```
    <form method="POST" action="{{ url('/read-file')}}" enctype='multipart/form-
data' >
        @csrf
        <input type="file" name="myFile" id="myFile">
        <br>
        <input type="submit" name="submit" id="submit" value="Send">
    </form>
    <br>
    <div class="history">
        @if (session('content') && count(session('content')) > 0)
            <ul>
                @foreach (session('content') as $row)
                    <li>{{ implode(', ', $row) }}</li>
                @endforeach
            </ul>
        @endif
    </div>
```

- Captura:



- test1, test@example.com

Práctica 19

> 🗁 Mostrar en una página una lista de ficheros de una carpeta en storage. Cuando se pulse en el nombre del fichero se descargará

- Routes:

```
Route::get('/practice19', [Practice19Controller::class, 'showFiles']);
Route::get('/practice19/download/{filename}', [Practice19Controller::class,
'downloadFile']);
```

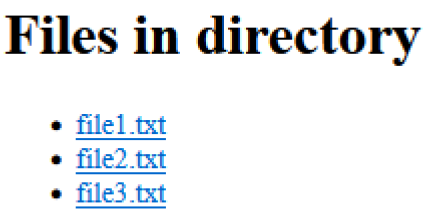
- Controller:

```
    public function showFiles(){
        $files = Storage::files('practice19');
        return view('practice19', compact('files'));
    }


    public function downloadFile($filename){
        return Storage::download('practice19/' . $filename);
    }
```

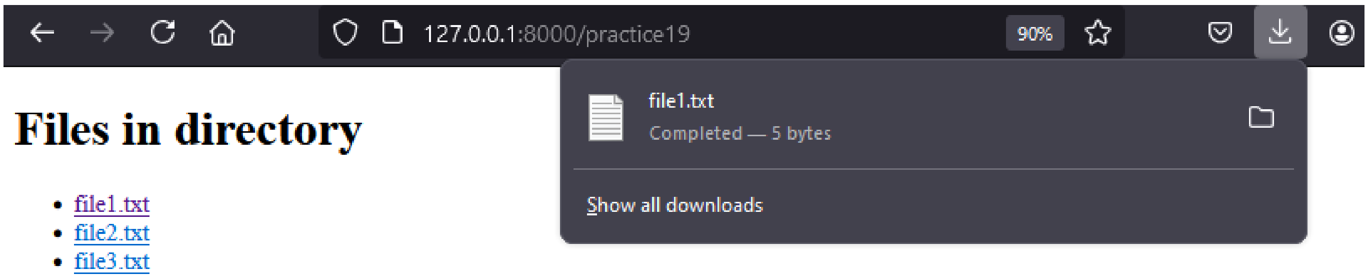
- View:

```
    <div class="main-container">
        <h1>Files in directory</h1>
        <ul>
            @foreach ($files as $file)
                <li>
                    <a href="{{ url('practice19/download/' . basename($file)) }}">
{{ basename($file) }}</a>
                </li>
            @endforeach
        </ul>

        @if ($files === null)
            <p>There are no files in this directory.</p>
        @endif
    </div>
```

- Captura:


Files in directory

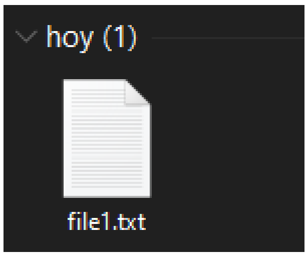
- file1.txt
- file2.txt
- file3.txt


127.0.0.1:8000/practice19

Files in directory

- file1.txt
- file2.txt
- file3.txt

file1.txt
Completed — 5 bytes
Show all downloads



Práctica 20

> 🗁 Continuando con el ejemplo anterior crear una opción para borrar los ficheros listados

- Routes:

```
Route::get('/practice20', [Practice20Controller::class, 'showFiles']);
Route::get('/practice20/download/{filename}', [Practice20Controller::class,
'downloadFile']);
Route::post('/practice20/delete/{filename}', [Practice20Controller::class,
'deleteFile']);
```

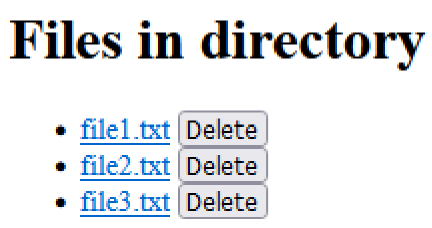
- Controller:

```
    public function showFiles(){
        $files = Storage::files('practice19');
        return view('practice19', compact('files'));
    }


    public function downloadFile($filename){
        return Storage::download('practice19/' . $filename);
    }

    public function deleteFile($filename){
        Storage::delete('practice20/' . $filename);
        return redirect('/practice20');
    }
```

- View:

```
<div class="main-container">
    <h1>Files in directory</h1>
    <ul>
        @foreach ($files as $file)
            <li>
                <a href="{{ url('practice20/download/' . basename($file)) }}">{{
basename($file) }}</a>
                <form action="{{ url('practice20/delete', basename($file)) }}"
method="POST" style="display:inline;">
                    @csrf
                    <button type="submit">Delete</button>
                </form>
            </li>
        @endforeach
    </ul>

    @if ($files === null)
        <p>There are no files in this directory.</p>
    @endif
</div>
```

- Captura:



Files in directory

- file1.txt Delete
- file2.txt Delete
- file3.txt Delete

Extra:

To do list - Session

- Routes:

```
Route::get('/', [FormController::class, 'show']);
Route::get('/task', [FormController::class, 'getTask']);

Route::post('/task/create', [FormController::class, 'createTask']);
```

```
Route::post('/task/delete', [FormController::class, 'deleteTask']);

Route::post('/task/update', [FormController::class, 'updateForm']);
```

Esta es la única clase del modelo.

- Task:

```
  /**
     * @var int
     */
    public $id;

    /**
     * @var string
     */
    public $subject;

    /**
     * @var string
     */
    public $description;

    /**
     * @var bool
     */
    public $finished;

    /**
     * Task constructor.
     *
     * @param string $subject
     * @param string $description
     *
     *
     * */

     public function __construct(string $subject = "", int $id = 0, string 
$description = "", bool $finished = false){
        $this->subject = $subject;
        $this->id = $id;
        $this->description = $description;
        $this->finished = $finished;
    }

    // getters and setters
```

- Controller:

```
 public function show(){
        $todolist = session()->get('todolist');

        if (!isset($todolist)) {
            $todolist = [];
            session()->put('todolist', $todolist);
        }

        return view('startpage', compact('todolist'));
    }


    public function getTask(Request $request){
        $id = $request->input('id');

        $todolist = session()->get('todolist');
        $auxTask = null;


        foreach ($todolist as $item) {
            if($item->getId() == $id){
                $auxTask = $item;
                break;
            }
        }

        return view('tasks', compact('auxTask'));
    }

    public function createTask(Request $request){
            $todolist = session()->get('todolist', []);

            $subject = $request->input('subject')??null;
            $id = count($todolist) + 1;
            $description=$request->input('description')??null;
            $finished = $request->input('finished') === 'Closed' ? true : false;

            $newTask = new Task($subject, $id, $description, $finished);
            $todolist[] = $newTask;

            session()->put('todolist', $todolist);

        return redirect('/');
    }

    public function updateForm(Request $request){
        $todolist = session()->get('todolist', []);

        $subject = $request->input('subject');
        $id = $request->input('id');
        $description=$request->input('description');
        $finished = $request->input('finished') === 'Closed' ? true : false; //
important to declare
```

```
        foreach($todolist as $key => $item){
            if($item->getId() == $id){
                $item->setSubject($subject);
                $item->setDescription($description);
                $item->setFinished($finished);
                $todolist[$key] = $item;
                break;
            }
        }

        session()->put('todolist', $todolist);
        return redirect('/');
    }


    public function deleteTask(Request $request){
        $todolist = session()->get('todolist', []);
        $id = $request->input('id');

        foreach($todolist as $key => $item){
            if($item->getId() == $id){
                unset($todolist[$key]);
                break;
            }
        }

        session()->put('todolist', array_values($todolist));
        return redirect('/');
    }
```

- View (página principal):

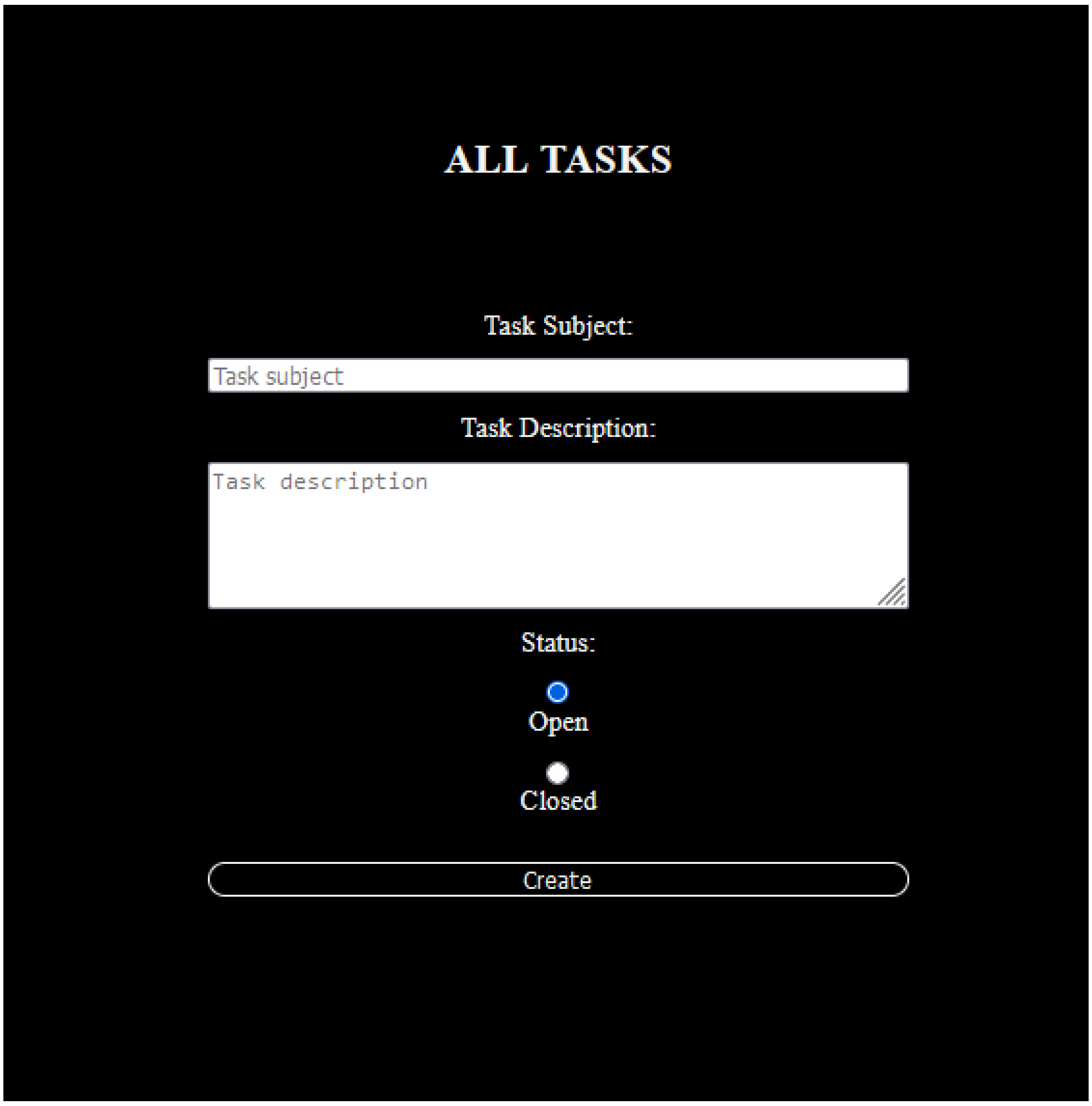
```
<div class="main-container">

    <h2>ALL TASKS</h2>
    <ul>
        @foreach ($todolist as $task)
            <li>
                <a href="./task?id={{$task->id}}">{{ $task->subject }}</a>
                <a href="./task?id={{$task->id}}">{{ $task->description }}</a>
                <a href="./task?id={{$task->id}}">{{ $task->finished ? 'Finished'
: 'Not finished' }}</a>

                <form action="{{ url('task/delete') }}" method="POST"
style="display:inline;">
                    @csrf
                    <input type="hidden" name="id" value="{{ $task->id }}">
                    <input type="submit" name="delete" id="delete" value="Delete">
                </form>
            </li>
        @endforeach
    </ul>

    <form action="{{ url('task/create') }}" method="POST">
        @csrf
        <label for="subject">Task Subject:</label>
        <input type="text" name="subject" id="subject" placeholder="Task subject"
/>
        <br>
        <label for="description">Task Description:</label>
        <textarea cols="50" rows="5" name="description" id="description"
placeholder="Task description"></textarea>
        <br>
        <label for="finished">Status:</label><br>
        <div class="status-container">
            <input type="radio" value="Open" name="finished" id="finishedOpen"
checked>
            <label for="finishedOpen">Open</label><br>
            <input type="radio" value="Closed" name="finished"
id="finishedClosed">
            <label for="finishedClosed">Closed</label><br>
        </div>
        <br>
        <input type="submit" name="create" value="Create">
    </form>
</div>
```

- View (Task seleccionada):

```
<div class="main-container">
    <h2>TASK TO UPDATE</h2>
    <form action="{{ url('task/update')}}" method="POST" id="formTasks">
        @csrf
        <label for="subject">Task ID: {{$auxTask->id}}</label>
        <br>
        <input type="hidden" name="id" value="{{ $auxTask->id }}">
        <input type="text" name="subject" id="subject" placeholder="Task subject"
value="{{$auxTask->subject}}" />
        <textarea cols="200" rows="5" name="description" placeholder="Task
description" id="description">{{$auxTask->description}}</textarea>

        <label for="finished">Status:</label><br>
        <div class="status-container">
            @if ($auxTask->finished)
                <input type="radio" value="Open" name="finished"
id="finishedOpen">
                <label for="finishedOpen">Open</label>
                <input type="radio" value="Closed" name="finished"
id="finishedClosed" checked>
                <label for="finishedClosed">Close</label>
            @else
                <input type="radio" value="Open" name="finished" id="finishedOpen"
checked>
```

```
                <label for="finishedOpen">Open</label>
                <input type="radio" value="Closed" name="finished"
id="finishedClosed">
                <label for="finishedClosed">Close</label>
            @endif
        </div>

        <input type="submit" name="submit" id="submit" value="Update">
    </form>
</div>
```

- Captura:



ALL TASKS

Task Subject:

Task subject

Task Description:

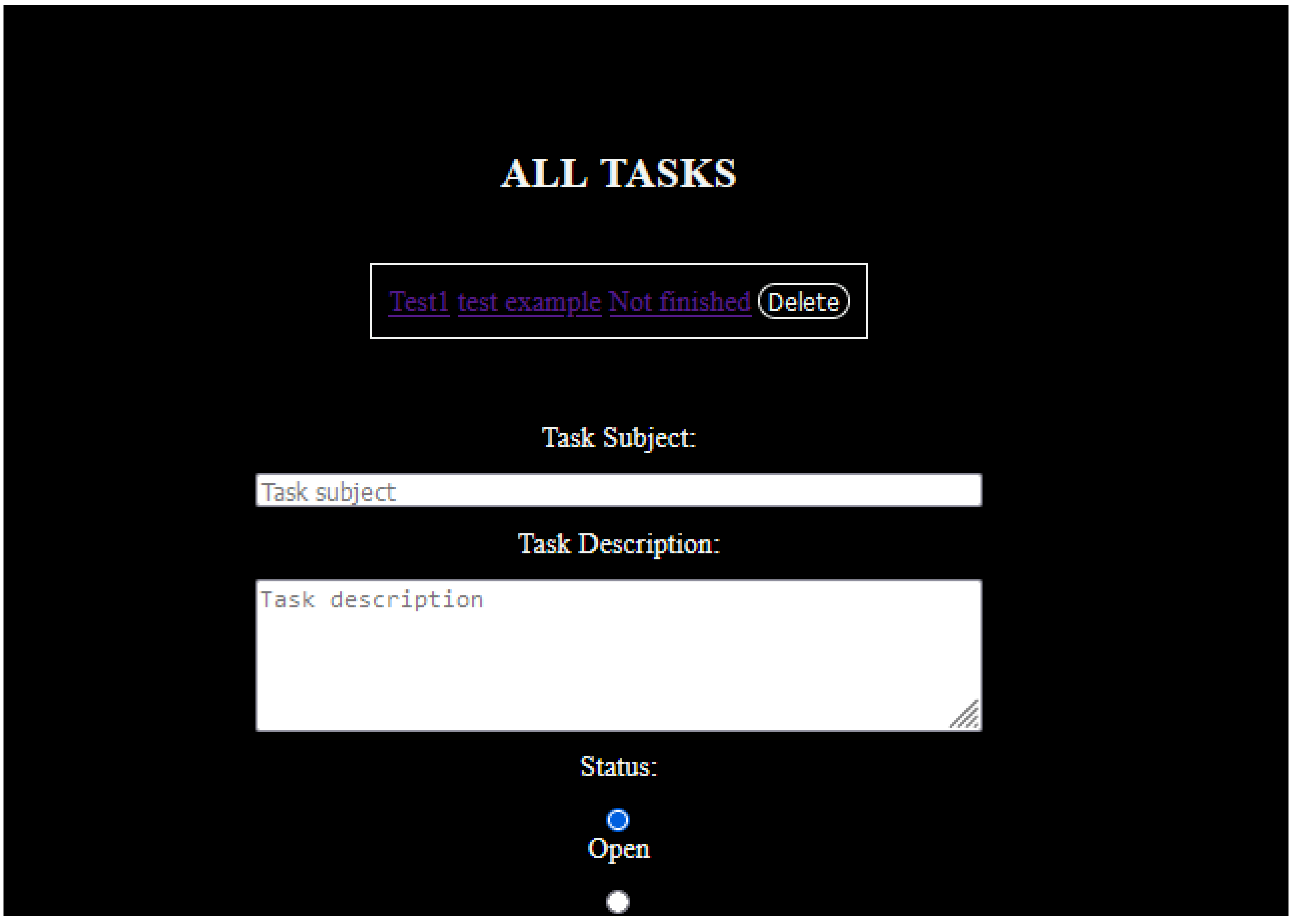
Task description

Status:

Open

Closed

Create



ALL TASKS

Test1 test example Not finished Delete

Task Subject:

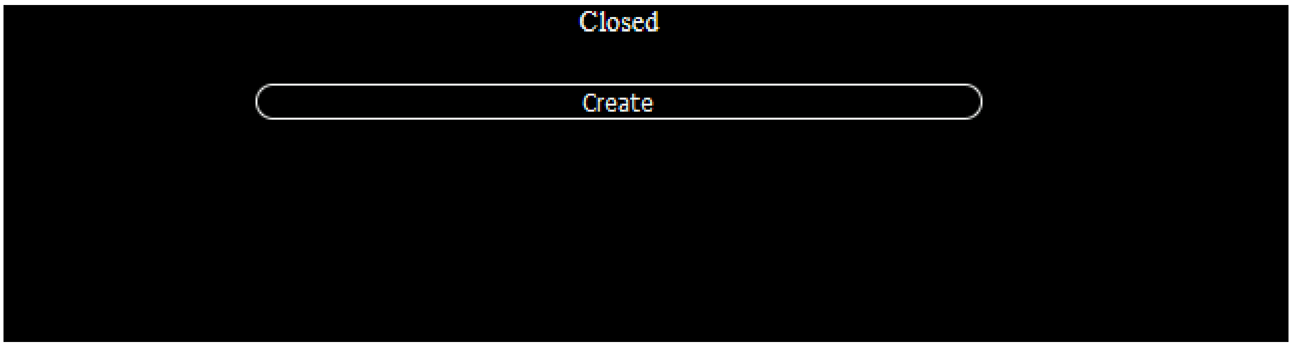
Task subject

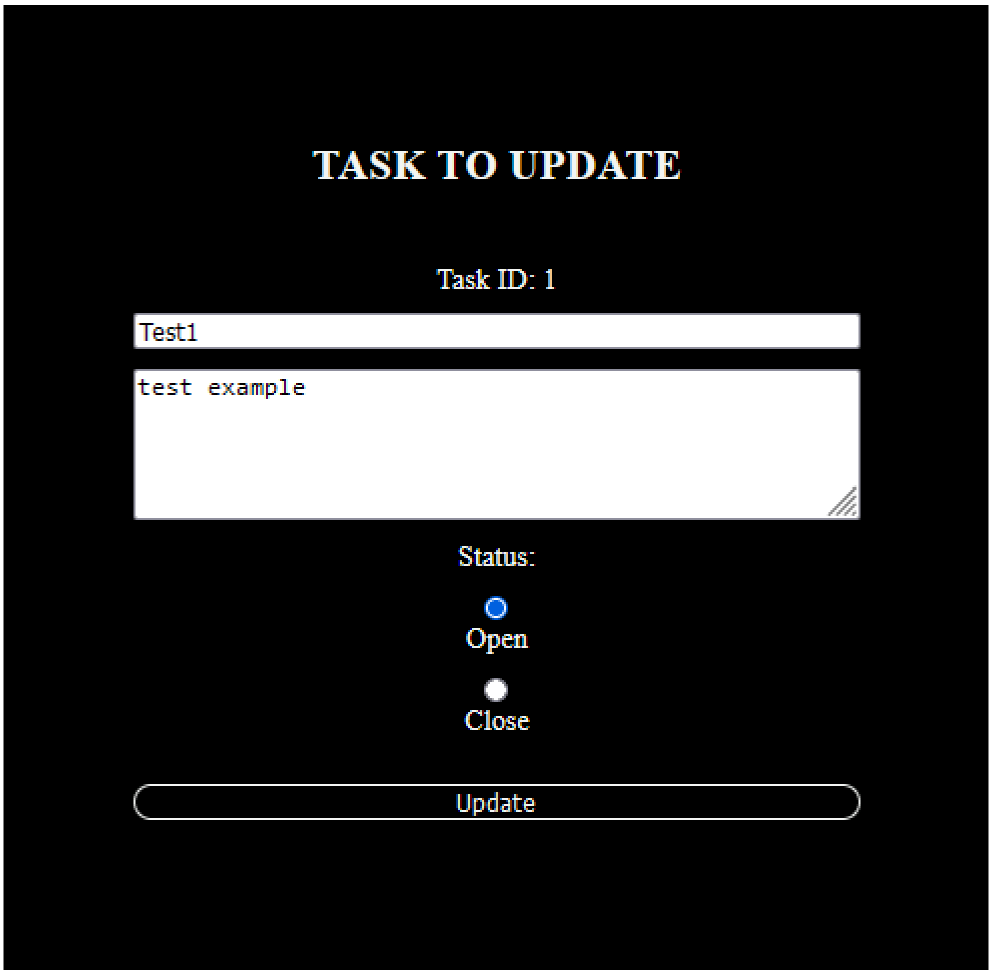
Task Description:

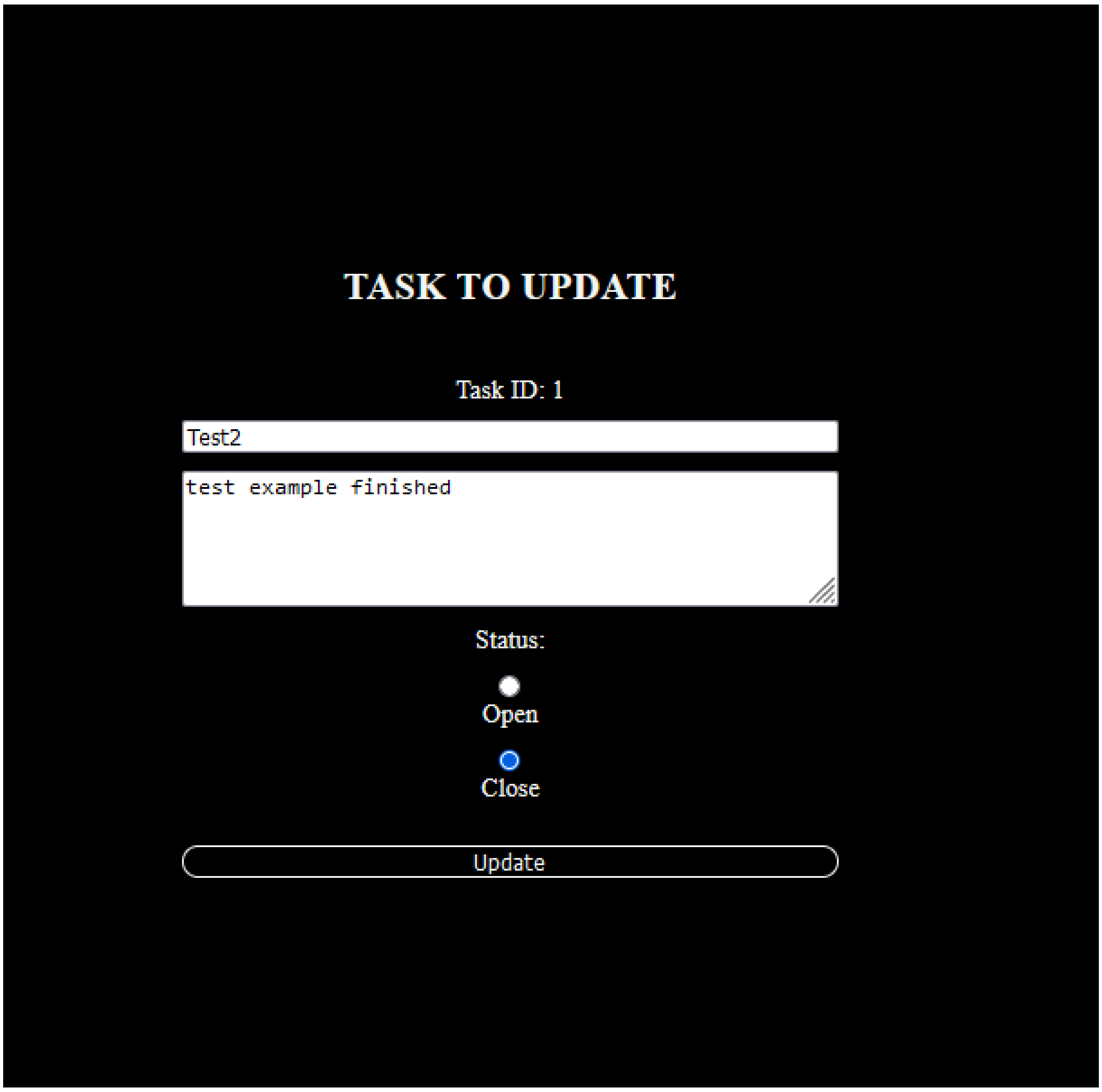
Task description

Status:

Open


Closed
Create


TASK TO UPDATE
Task ID: 1
Test1
test example
Status:
Open
Close
Update


TASK TO UPDATE

Task ID: 1

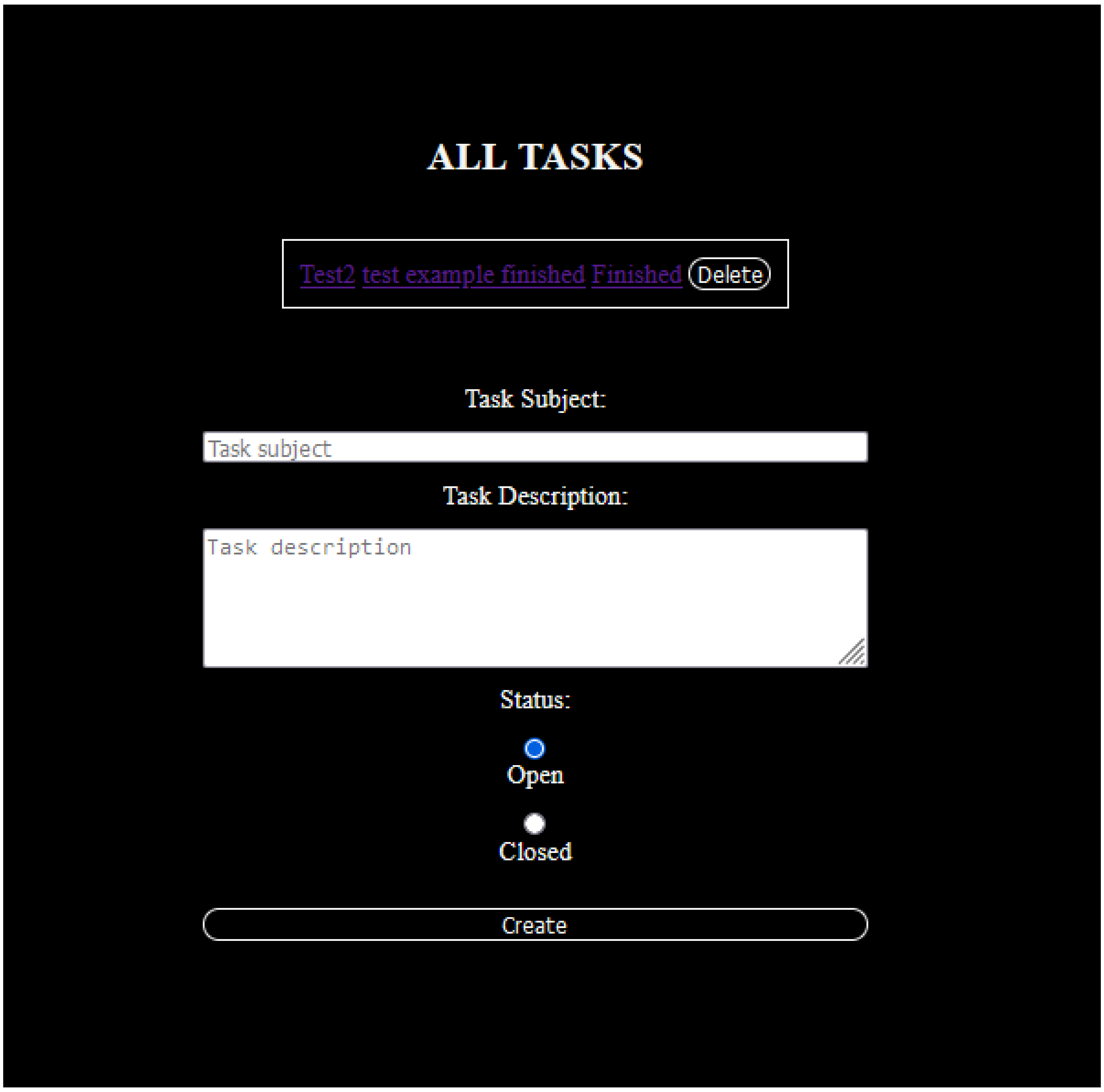
Test2

test example finished

Status:

Open

Close [✓]



To do list - Files

- Routes:

```
Route::get('/', [FormController::class, 'getAllTasks']);
Route::get('/task', [FormController::class, 'getTask']);
Route::post('/task/create', [FormController::class, 'createTask']);
Route::post('/task/delete', [FormController::class, 'deleteTask']);
Route::post('/task/update', [FormController::class, 'updateForm']);
```

Esta es la única clase del modelo.

- Task:

```
  /**
     * @var int
     */
```

```
    public $id;

    /**
     * @var string
     */
    public $subject;

    /**
     * @var string
     */
    public $description;

    /**
     * @var bool
     */
    public $finished;

    /**
     * Task constructor.
     *
     * @param string $subject
     * @param string $description
     *
     *
     * */

     public function __construct(string $subject = "", int $id = 0, string
$description = "", bool $finished = false){
        $this->subject = $subject;
        $this->id = $id;
        $this->description = $description;
        $this->finished = $finished;
    }

    // getters and setters
```

- Controller

```
  /**
     * Read file
     */
    public function getAllTasks(){
        $filePath = storage_path('app/tasks.csv');

        if(!file_exists($filePath)){
            return redirect('/);
        }

        $tasks = [];


        if(($open = fopen($filePath, 'r') )!== false){
            while (($data = fgetcsv($open, 1000, ','))!== false) {
                if(count($data) == 4){
                    $tasks[] = new Task($data[0],
                                        $data[1],
                                        $data[2],
                                        $data[3] === 'Closed'? true : false);

                }
            }
            fclose($open);
            return view('startpage', compact('tasks'));
        }
    }

    /**
     * Get specific file
     */

    public function getTask(Request $request){

        $id = $request->input('id');

        $filePath = storage_path('app/tasks.csv');

        if(!file_exists($filePath)){
            return redirect('/');
        }

        $auxTask = null;

        if(($open = fopen($filePath, 'r') )!== false){
            while (($data = fgetcsv($open, 1000, ','))!== false) {
                if($data[1] == $id){
                    $auxTask =  new Task(
                                $data[0],
                                $data[1],
                                $data[2],
                                $data[3] === 'Closed'? true : false);
                    break;
                }

            }

            fclose($open);
            return view('tasks', compact('auxTask'));
        }
    }

    /**
     * Create a new Task
     */

    public function createTask(Request $request){
            $filePath = storage_path('app/tasks.csv');
            $subject = $request->input('subject')??null;
            $id = $this->getIdFromCsv($filePath);
            $description=$request->input('description')??null;
            $finished = $request->input('finished') === 'Closed' ? true : false;

            $newTask = new Task($subject, $id, $description, $finished);

            $open = fopen($filePath, 'a');
            if($open){
                fputcsv($open,[
                            $newTask->getSubject(),
                            $newTask->getId(),
                            $newTask->getDescription(),
                            $newTask->getFinished() ? 'Closed' : 'Open'
                        ]);
                fclose($open);
            }

        return redirect('/');
    }

    public function getIdFromCsv($filePath){
        if(file_exists($filePath)){
            $open = fopen($filePath, 'r');
            $id = 1;
            while (($data = fgetcsv($open, 1000, ','))!== false) {
                if(isset($data[1])){
                    $actualId = (int)$data[1];
                }
                $id = max($id, $actualId);
            }
            fclose($open);
            return $id+1;
        }

        return 1;
    }

    public function updateForm(Request $request){

        $filePath = storage_path('app/tasks.csv');

        if(!file_exists($filePath)){
            return redirect('/');
        }

        $subject = $request->input('subject');
        $id = $request->input('id');
        $description=$request->input('description');
        $finished = $request->input('finished') === 'Closed' ? true : false;

        $tasks = [];
        if(($open = fopen($filePath, 'r'))!== false){
            while (($data = fgetcsv($open, 1000, ','))!== false) {
                if($data[1] == $id){
                    $data[0] = $subject;
                    $data[2] = $description;
                    $data[3] = $finished ? 'Closed' : 'Open';
                }
                $tasks[] = $data;
            }
            fclose($open);
        }

        //var_dump($tasks);
        //die();

        if(($open = fopen($filePath, 'w'))!== false){
            foreach($tasks as $task){
                fputcsv($open, $task);
            }
            fclose($open);
        }

        return redirect('/');
    }


    /**
     * @param Request $request for get the id of the item to delete
     */
    public function deleteTask(Request $request){

        $filePath = storage_path('app/tasks.csv');

        if(!file_exists($filePath)){
            return redirect('/');
        }


        $id = $request->input('id');
        $tasks = [];

        if(($open = fopen($filePath, 'r'))!== false){
            while (($data = fgetcsv($open, 1000, ','))!== false) {
                if($data[1] != $id && count($data) >= 4){
                    $task =  new Task(
                                $data[0],
                                (int)$data[1],
                                $data[2],
                                $data[3] === 'Closed'? true : false
                            );
                    $tasks[] = $task;
                }

            }
            fclose($open);
        }


        if(($open = fopen($filePath, 'w'))!== false){
            foreach($tasks as $task){
                fputcsv($open,[
                    $task->getSubject(),
                    $task->getId(),
                    $task->getDescription(),
                    $task->getFinished() ? 'Closed' : 'Open'
                ]);
            }
            fclose($open);
        }

        return redirect('/');

    }
```

- View (página principal):

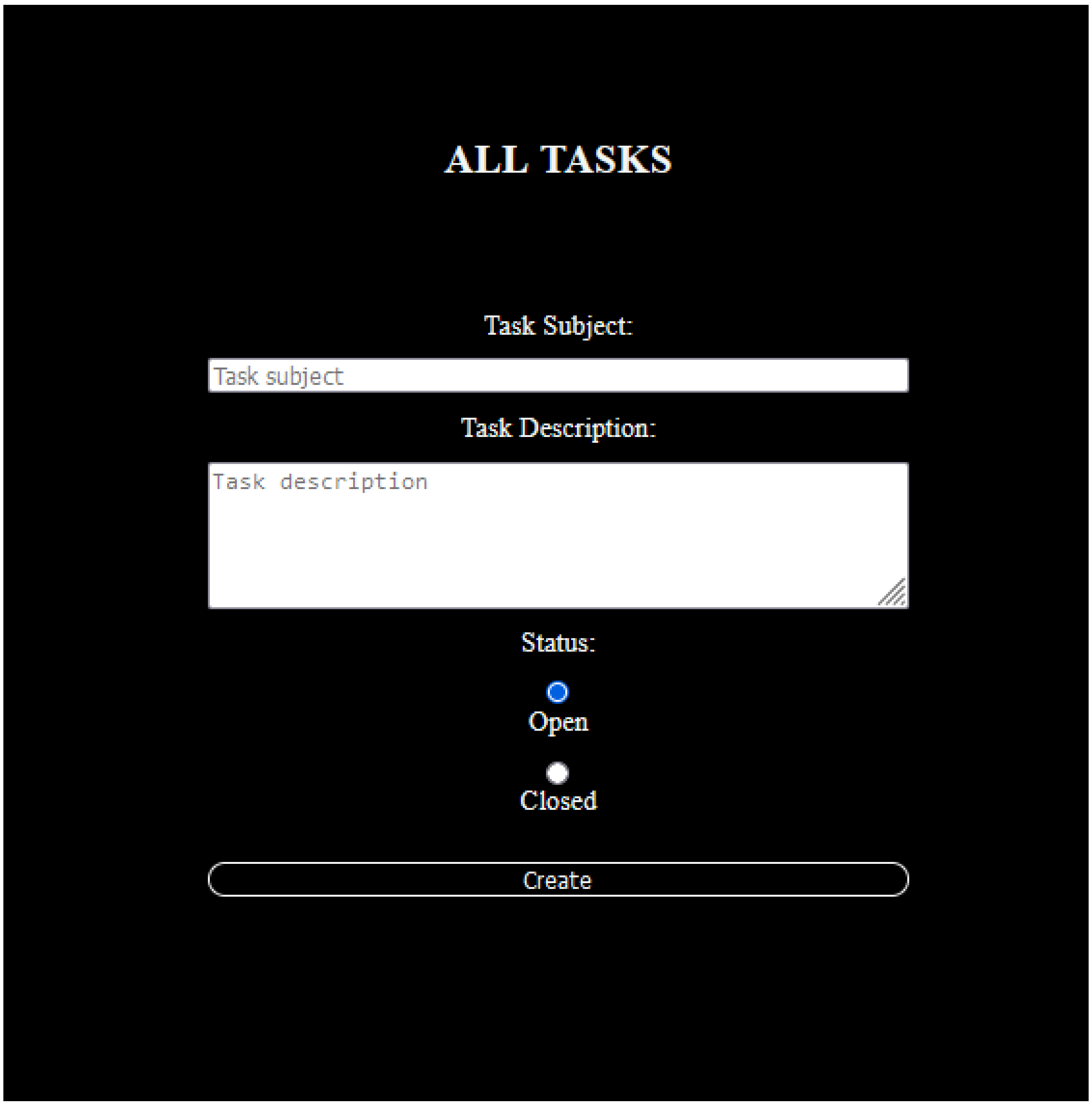
```
<div class="main-container">

    <h2>ALL TASKS</h2>
    <ul>
        @foreach ($todolist as $task)
            <li>
                <a href="./task?id={{$task->id}}">{{ $task->subject }}</a>
                <a href="./task?id={{$task->id}}">{{ $task->description }}</a>
                <a href="./task?id={{$task->id}}">{{ $task->finished ? 'Finished'
: 'Not finished' }}</a>

                <form action="{{ url('task/delete') }}" method="POST"
style="display:inline;">
                    @csrf
                    <input type="hidden" name="id" value="{{ $task->id }}">
                    <input type="submit" name="delete" id="delete" value="Delete">
                </form>
            </li>
        @endforeach
    </ul>

    <form action="{{ url('task/create') }}" method="POST">
        @csrf
        <label for="subject">Task Subject:</label>
        <input type="text" name="subject" id="subject" placeholder="Task subject"
/>
        <br>
        <label for="description">Task Description:</label>
        <textarea cols="50" rows="5" name="description" id="description"
placeholder="Task description"></textarea>
```

```
        <br>
        <label for="finished">Status:</label><br>
        <div class="status-container">
            <input type="radio" value="Open" name="finished" id="finishedOpen"
checked>
            <label for="finishedOpen">Open</label><br>
            <input type="radio" value="Closed" name="finished"
id="finishedClosed">
            <label for="finishedClosed">Closed</label><br>
        </div>
        <br>
        <input type="submit" name="create" value="Create">
    </form>
</div>
```

- View (Task seleccionada):

```
<div class="main-container">
    <h2>TASK TO UPDATE</h2>
    <form action="{{ url('task/update')}}" method="POST" id="formTasks">
        @csrf
        <label for="subject">Task ID: {{$auxTask->id}}</label>
        <br>
        <input type="hidden" name="id" value="{{ $auxTask->id }}">
        <input type="text" name="subject" id="subject" placeholder="Task subject"
value="{{$auxTask->subject}}" />
        <textarea cols="200" rows="5" name="description" placeholder="Task
description" id="description">{{$auxTask->description}}</textarea>

        <label for="finished">Status:</label><br>
        <div class="status-container">
            @if ($auxTask->finished)
                <input type="radio" value="Open" name="finished"
id="finishedOpen">
                <label for="finishedOpen">Open</label>
                <input type="radio" value="Closed" name="finished"
id="finishedClosed" checked>
                <label for="finishedClosed">Close</label>
            @else
                <input type="radio" value="Open" name="finished" id="finishedOpen"
checked>
                <label for="finishedOpen">Open</label>
                <input type="radio" value="Closed" name="finished"
id="finishedClosed">
                <label for="finishedClosed">Close</label>
            @endif
        </div>

        <input type="submit" name="submit" id="submit" value="Update">
    </form>
</div>
```

- Captura:


ALL TASKS

Task Subject:

Task subject

Task Description:

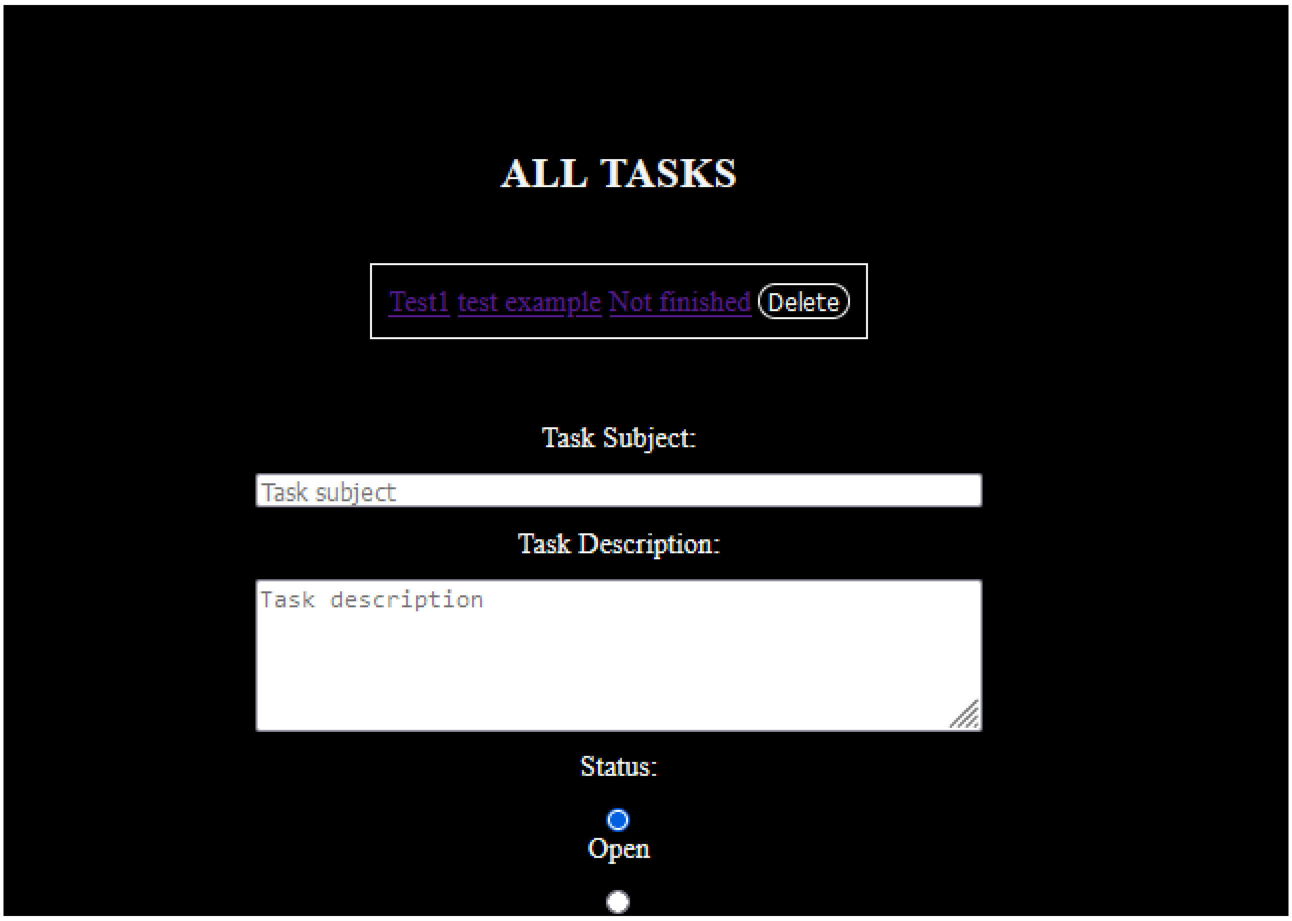
Task description

Status:

Open

Closed

Create


ALL TASKS

Test1 test example Not finished Delete

Task Subject:

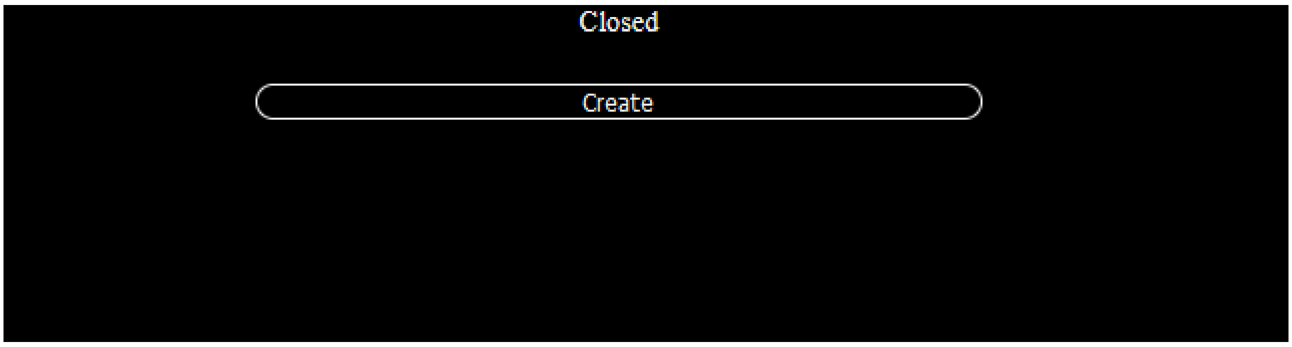
Task subject

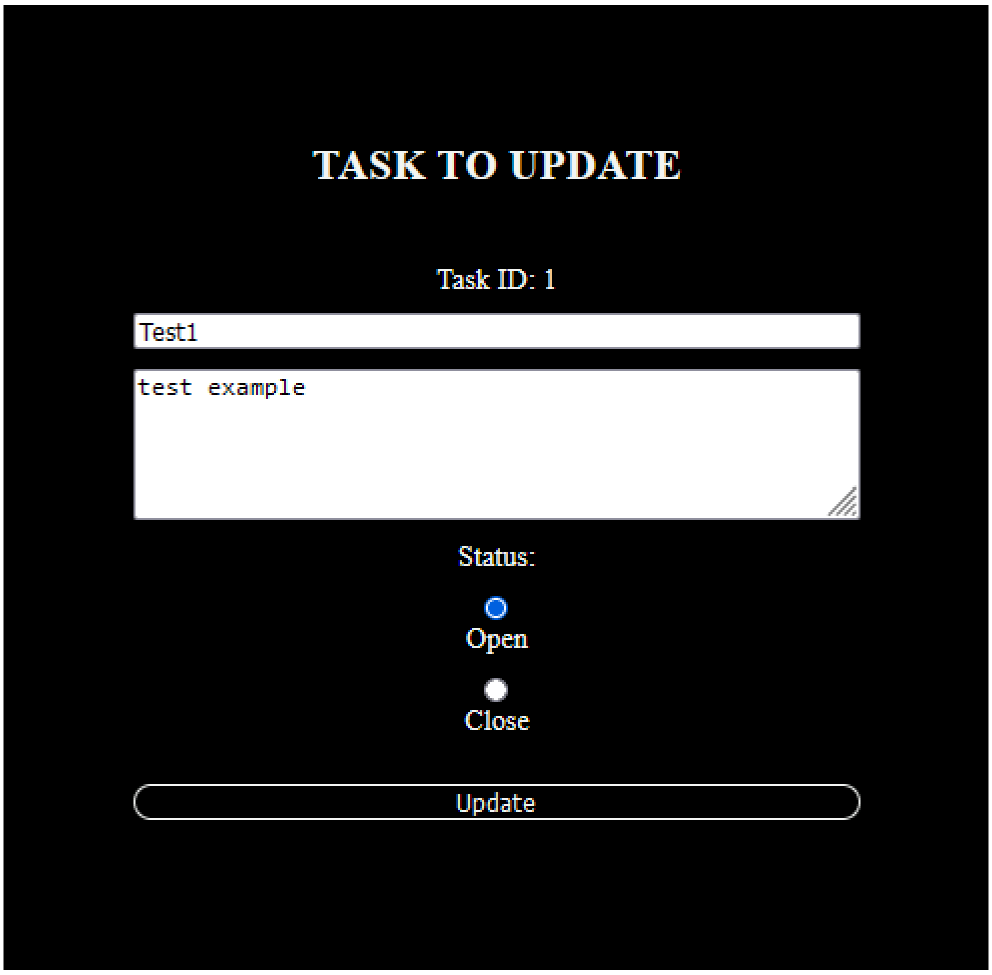
Task Description:

Task description

Status:

Open

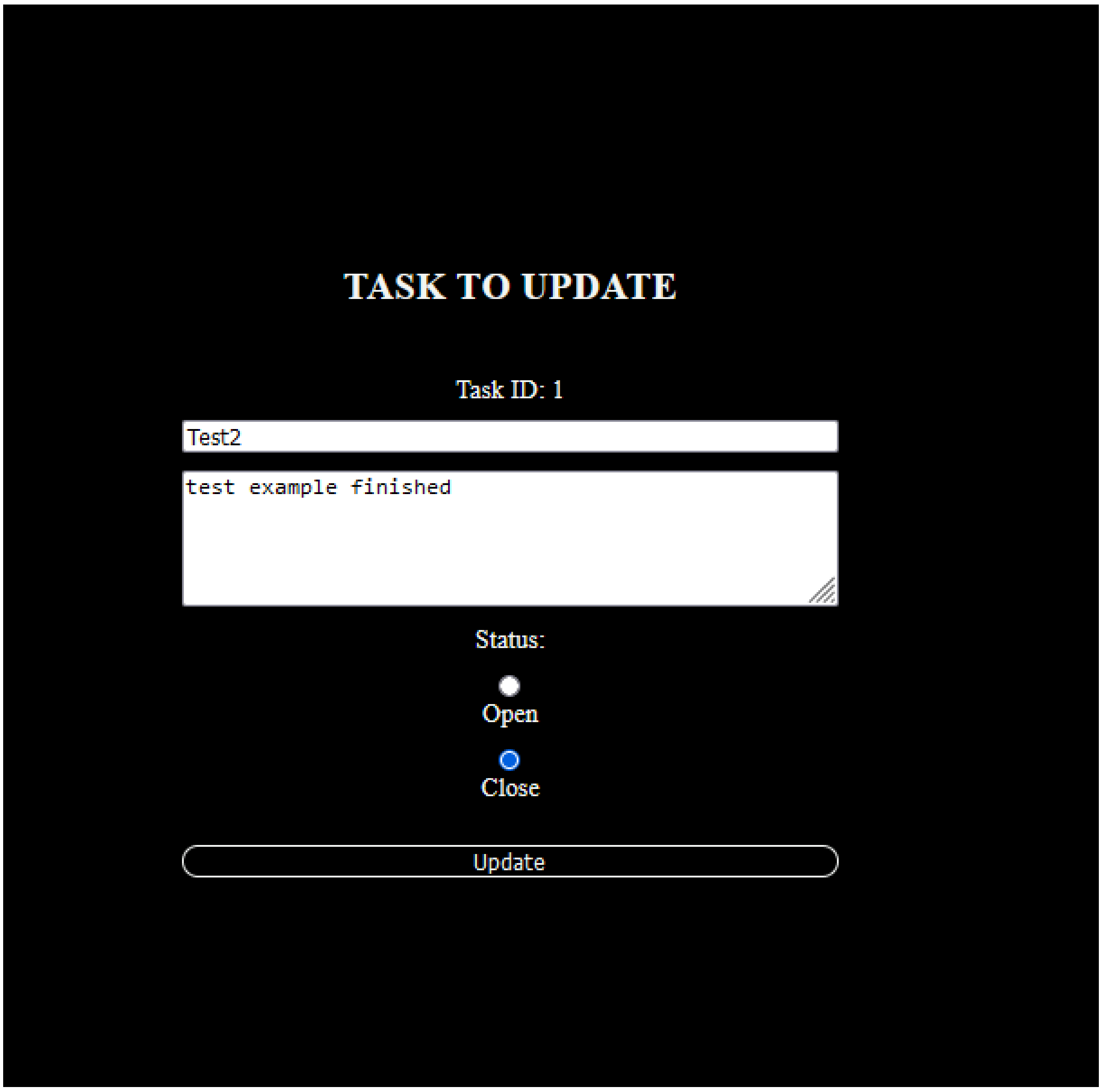




Test1

test example

Status: [✓ Open] Close



TASK TO UPDATE

Task ID: 1

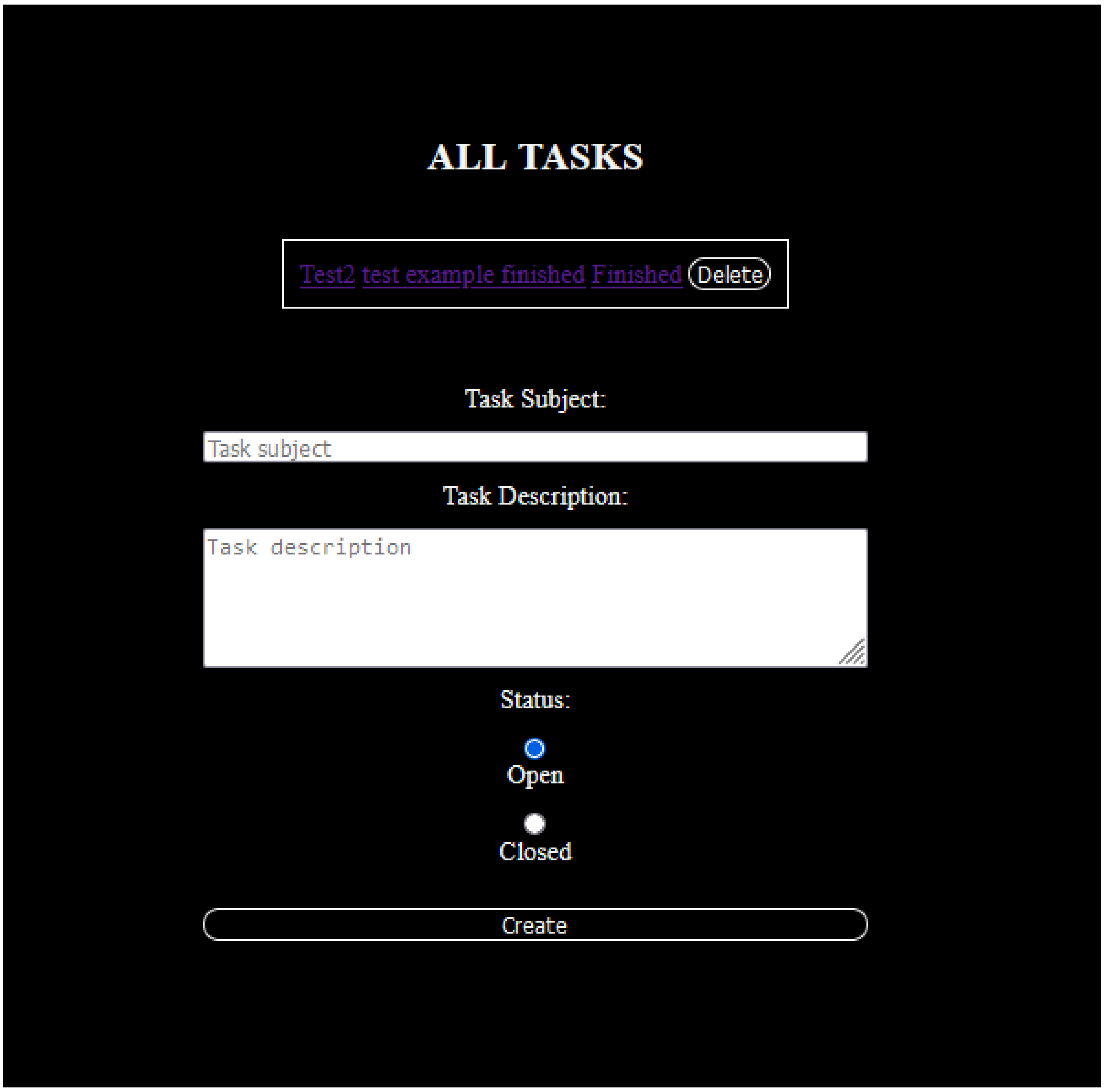
Test2

test example finished

Status:

Open

[✓ Close]



BlackJack

- Routes:

```
/**
 * Login
 */
Route::get('/', function (){
    return view('login');
});

Route::post('/login', [LoginController::class, 'createUser']);


/**
 * Game
 */

Route::get('/blackjack', function () {
    return view('blackjack');
});

Route::post('/start-game', [GameController::class, 'startGame']);

Route::post('/player-action', [GameController::class, 'getActions']);
```

Las clases del modelo son las siguientes:

- Card:

```


    /**
    * @var string type of card (hearts, )
    */
    public $suit;

    /**
    * @var string
    */
    public $rank;

    /**
    * @var int
    */
    public $value;


    /**
    * Constructor of the class
    */
    public function __construct(string $suit, string $rank, int $value){
        $this->suit = $suit;
        $this->rank = $rank;
        $this->value = $value;
    }

    // getters and setters
```

- DeckCards:

```


 /**
     * @var array
     */
    public $deckCards = [];

    public $currentIndexDeck = 0;
    public function __construct(){
        $this->deckCards = $this->initializeDeck();
        shuffle($this->deckCards);
    }

    /**
     * Function to create the deck cards
     */
    public function initializeDeck() {
        $suits = ['Hearts', 'Diamonds', 'Clubs', 'Spades'];
        $ranks = [
            '2' => 2, '3' => 3, '4' => 4, '5' => 5, '6' => 6, '7' => 7, '8' => 8,
'9' => 9, '10' => 10,
            'J' => 10, 'Q' => 10, 'K' => 10, 'A' => 11
        ];

        foreach ($suits as $suit) {
            foreach ($ranks as $rank => $value) {
                $this->deckCards[] = new Card($suit, $rank, $value);
            }
        }
        return $this->deckCards;
    }

    /**
     * Function to draw a card from the deck and remove it from it
     */
    public function drawCard(){
        if ($this->currentIndexDeck >= count($this->deckCards)){
            return null;
        }

        $cardSelected = $this->deckCards[$this->currentIndexDeck];
        $this->currentIndexDeck++;
        return $cardSelected;
    }

    // getters and setters
```

- Game:

```

    /**
     * @var DeckCards
     */
    public $deck;
    /**
     * @var Player
     */
    public $playerGame;

    /**
     * @var Player
```

```
         */

         public $dealer;

        const BLACKJACK = 21;
        const DEALER_STAND = 17;
        const HIT = 'hit';
        const STAND = 'stand';


        public function __construct($playerGame){
            $this->deck = new DeckCards();
            $this->dealer = new Player("Dealer");
            $this->playerGame = $playerGame;
        }

        public function initialDeal() {
            for ($i = 0; $i < 2; $i++) {
                $this->playerGame->addCard($this->deck->drawCard());
                $this->dealer->addCard($this->deck->drawCard());
            }
        }


        public function getActions($playerAction){
            if($playerAction == self::HIT){
                $card = $this->deck->drawCard();
                $this->playerGame->addCard($card);

                if($this->playerGame->getScore() > self::BLACKJACK){
                    return $this->checkGameOver();
                }

            } elseif($playerAction == self::STAND){
                $this->playerGame->setIsStand(true);
            }

            $dealerAction = $this->dealerActions();

            if($dealerAction == self::HIT){
                $card = $this->deck->drawCard();
                $this->dealer->addCard($card);

            } elseif($dealerAction == self::STAND){
                $this->dealer->setIsStand(true);
                //dd($dealerAction);
            }

            if ($this->playerGame->getIsStand() && $this->dealer->getIsStand()) {
                return $this->checkGameOver();
            }
        }
```

```
    public function dealerActions(){
        $score = $this->dealer->getScore();

        if($score < 11){
            return self::HIT;
        }

        if($score >= 11 && $score <= self::DEALER_STAND){
            $probability = rand(1, 100);
            if($probability >= 70){
                return self::HIT;
            } else {
                return self::STAND;
            }
        }

        if($score == self::BLACKJACK){
            return self::STAND;
        }

        return self::STAND;
    }



    public function checkGameOver() {
        $playerScore = $this->playerGame->getScore();
        $dealerScore = $this->dealer->getScore();
        if ($playerScore > self::BLACKJACK) {
            $this->endGame();
            return false;
        }

        if ($dealerScore > self::BLACKJACK) {
            $this->endGame();
            return true;
        }

        if ($playerScore > $dealerScore) {
            $this->endGame();
            return true;
        } else {
            $this->endGame();
            return false;
        }
    }



    public function endGame(){
        $this->playerGame->setIsStand(false);
        $this->dealer->setIsStand(false);

        $this->deck = new DeckCards();
    }

    // getters and setters
```

- Player:

```

    /**
     * @var string name of the player
     */
    public $playerName;
    /**
     * @var array of cards in the player's hand
     */
    public $hand;
    /**
     * @var int number of points of the player
     */
    public $score;

    /**
     * @var bool
     */
    public $isStand;


    const BLACKJACK = 21;
    const ACE_VALUE = 10;

    /**
     * Constructor of the class
     */
    public function __construct($playerName="") {
        $this->playerName = $playerName;
        $this->hand = [];
        $this->score = 0;
        $this->isStand = false;
    }


    /**
     * Add a card to the player's hand
     * @param  Card  $card  the card to be added
     **/

    public function addCard(Card $card) {
         $this->hand[] = $card;
         $this->score = $this->calculateScore();

    }

    /**
     * Function to calculate the score of the player
     */

    public function calculateScore() {
        $aceCounter = 0;
        $this->score = 0;

        foreach ($this->hand as $card) {
            $this->score += $card->getValue();
            if ($card->getRank() == 'A') {
                $aceCounter++;
            }
        }

        while($aceCounter > 0 && $this->score > self::BLACKJACK){
            $this->score = $this->score - self::ACE_VALUE;
            $aceCounter--;
        }

        return $this->score;
    }

    // getters and setters
```

- UserModel:

```
    /**
     * @var int
     */
    public $id;

    /**
     * @var string
     */
    public $username;


    public function __construct (int $id=0, string $username=""){
        $this->id=$id;
        $this->username=$username;
    }

    // getters and setters
```

- LoginController:

```
       public function createUser(Request $request){

        $filePath = storage_path('app/users.csv');
        $username = $request->input('username');
        $userExists = $this->getUserIfExists($username, $filePath);

        if($userExists !== null){
            session(['user' => $userExists]);

            if (session('player') === null) {
                session(['player' => new Player($username)]);
            }

            return redirect('/blackjack');
        }

        $id = $this->createId($filePath);

        $newUser = new UserModel();
        $newUser->setId($id);
        $newUser->setUsername($username);

        session(['user' => $newUser]);
        session(['player' => new Player($username)]);

        $open = fopen($filePath, 'a');
        if($open){
            fputcsv($open, [
                        $newUser->getId(),
                        $newUser->getUsername()
                    ]);
            fclose($open);
        }

        return redirect('/blackjack');
    }



    public function createId($filePath){
     if(file_exists($filePath)){
        $open = fopen($filePath, 'r');
        $id = 1;
        while (($data = fgetcsv($open, 1000, ','))!== false) {
            if(isset($data[0])){
                $actualId = (int)$data[0];
            }
            $id = max($id, $actualId);
        }
        fclose($open);
        return $id+1;
     }
     return 1;
    }

    public function getUserIfExists($username, $filePath){
        if(!file_exists($filePath)){
            return redirect('/');
        }

        $auxUser = null;

        if (($open = fopen($filePath, 'r')) !== false) {
            while (($data = fgetcsv($open, 1000, ',')) !== false) {
                if (isset($data[1]) && $data[1] == $username) {
                    $auxUser = new UserModel();
                    $auxUser->setId($data[0]);
                    $auxUser->setUsername($data[1]);
                    fclose($open);
                    return $auxUser;
                }
            }
        }

        return null;

    }
```

- GameController:

```


    public  $game;

    // route /player-action
    public function getActions(Request $request){
       $game = session('game');
       $player = new Player();
       if (!$game) {
            $playerName = $request->input('playerName');
            $player->setPlayerName($playerName);
            $game = new Game($player);
            session(['game' => $game]);
        } else {
            $player = $game->getPlayerGame();
        }

        $action = $request->input('action');
        $result = $game->getActions($action);
        $dealer = $game->getDealer();

            if ($result === true){
                $message = $player->getPlayerName() . " wins!";
                session(['firstTry' => true]);
            } elseif ($result === false){
                $message = $dealer->getPlayerName() . " wins!";
                session(['firstTry' => true]);
            } else {
                $message = "";
                session(['firstTry' => false]);
            }

            session(['message' => $message]);


        $dealer = $game->getDealer();

        session(['game' => $game]);
        session(['player' => $player]);
        session(['dealer' => $dealer]);

        return redirect('/blackjack');
    }

    // route /start-game
    public function startGame(Request $request) {
        $playerName = $request->input('playerName');
        $player = new Player();
        $player->setPlayerName($playerName);
        $game = new Game($player);
        $dealer = $game->getDealer();
        $message = "";
        $game->initialDeal();
        $firstTry = false;

        session(['game' => $game]);
        session(['player' => $player]);
        session(['dealer' => $dealer]);
        session(['message' => $message]);
        session(['firstTry' => $firstTry]);


        return redirect('/blackjack');
    }
```

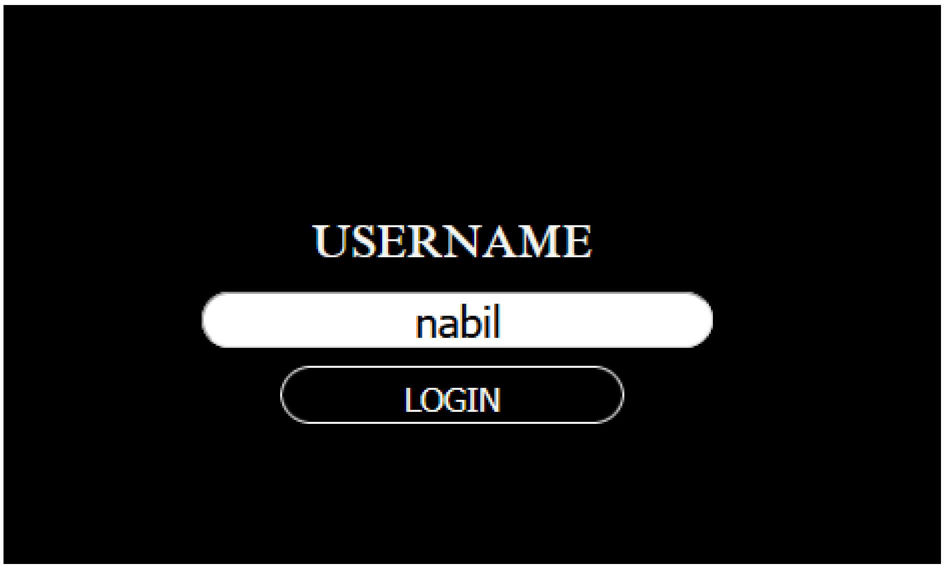
- View (login):

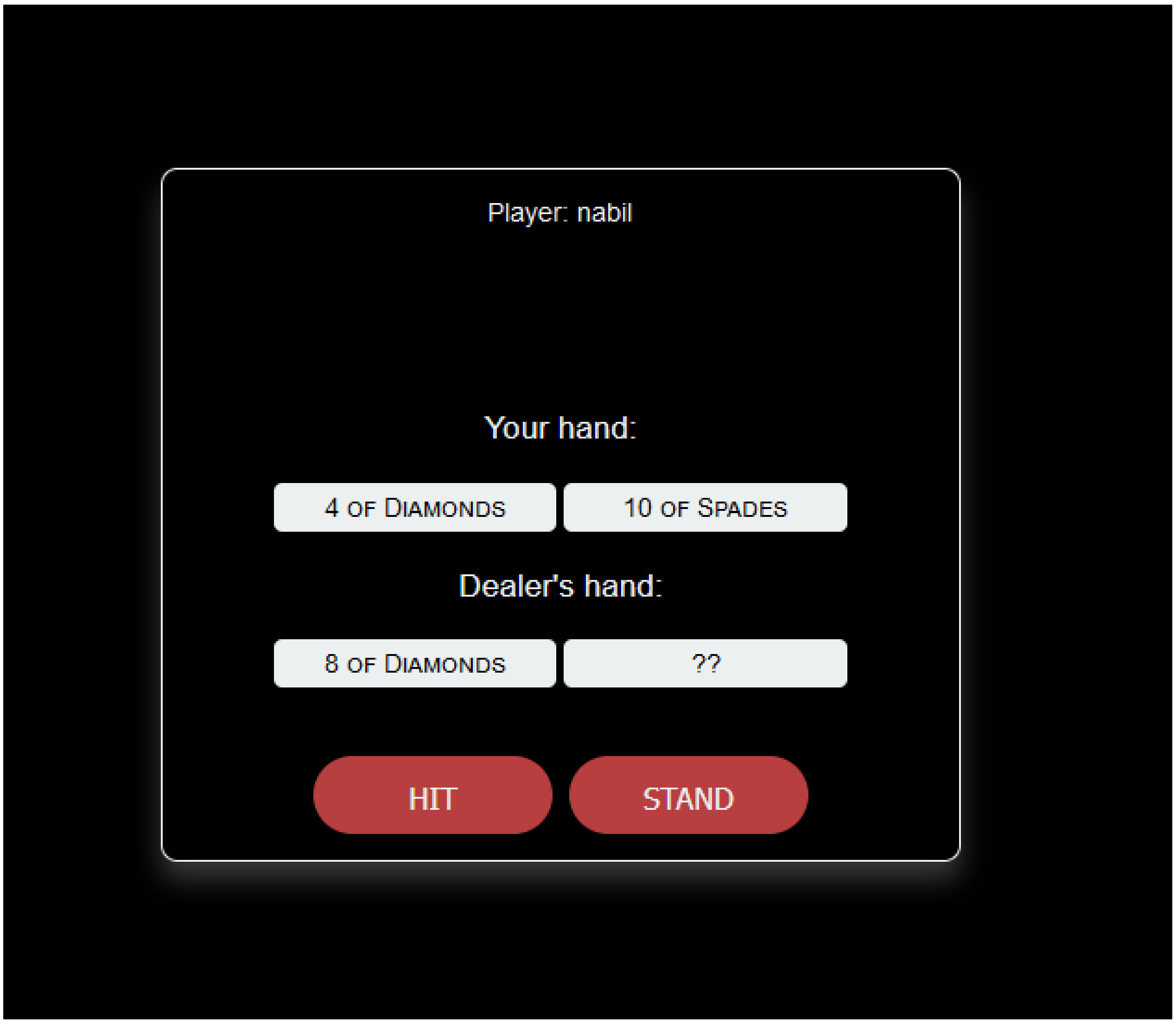
```
<div class="main-container">
    <form action="{{ url('/login')}}" method="POST">
            @csrf
            <label for="username">Username</label>
            <input type="text" name="username" id="username" placeholder="Enter
your username" />
            <br>
            <input type="submit" name="login" value="Login">
    </form>
</div>
```

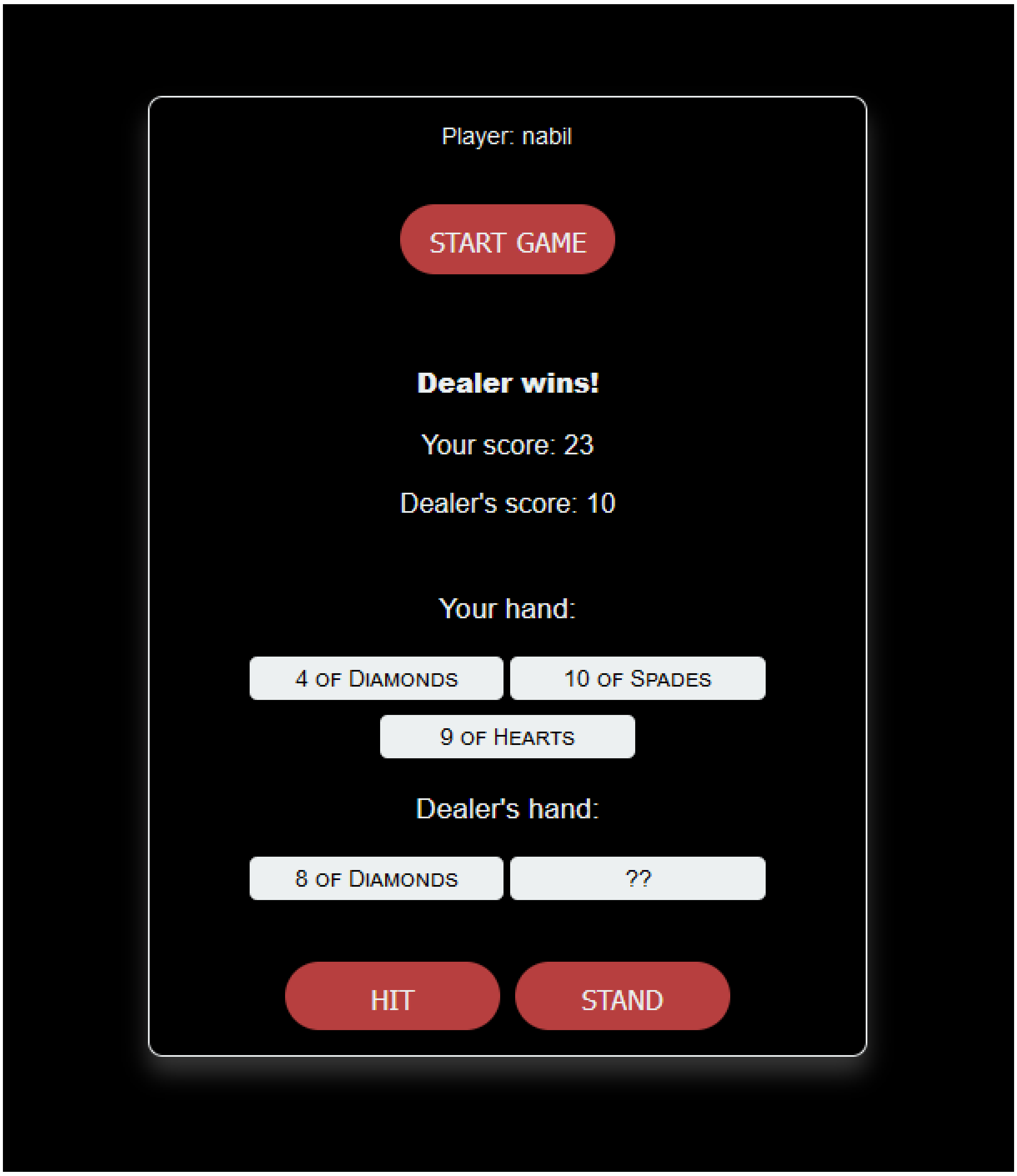
- View (juego):

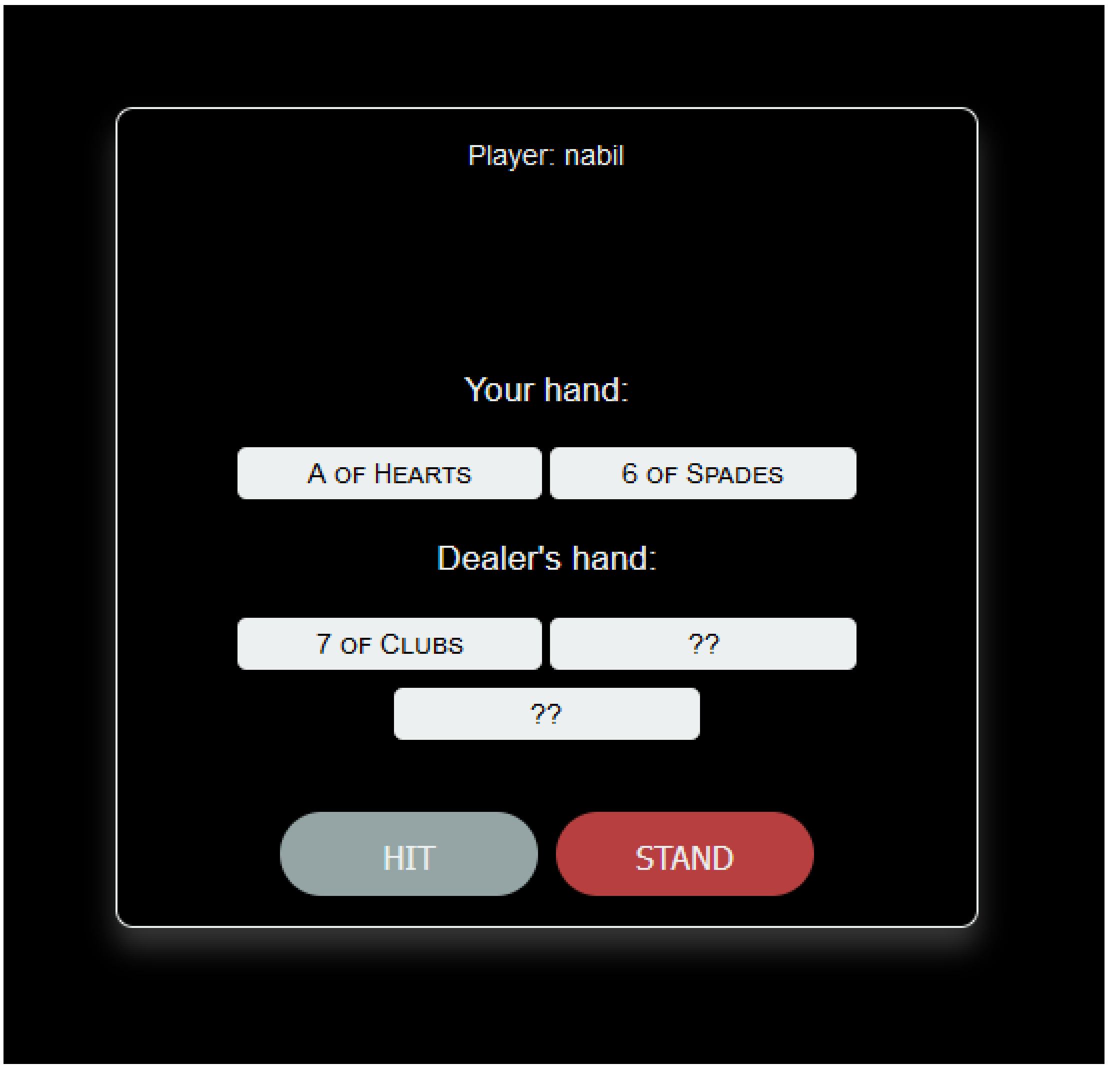
```
    @php
        $user = session('user');
        $username = $user ? $user->getUsername(): 'Anonymous';

        $player = session('player');
        $playerName = $username;
        $hand = $player ? $player->getHand() : [];
        $score = $player ? $player->getScore() : 0;
        $isStand = $player ? $player->getIsStand() : false;

        $dealer = session('dealer');
        $dealerHand = $dealer ? $dealer->getHand() : [];
        $dealerScore = $dealer ? $dealer->getScore() : 0;
        $dealerIsStand = $dealer ? $dealer->getIsStand() : false;

        $firstTry = session('firstTry', false);
    @endphp

    <div class="main-container">
    <div class="player-name">
        <p>Player: {{$playerName}}</p>
    </div>

    @if($firstTry)
        <div class="action-container">
            <form action="{{ url('start-game') }}" method="POST">
                @csrf
                <input type="hidden" id="playerName" name="playerName" value="{{
$playerName }}"></input>
                <input type="submit" id="startBtn" value="Start Game">
            </form>
        </div>
    @endif

    <br></br>
    <div class="result">
        @if(session('message'))
            <div class="message">
                <b>{{ session('message') }}</b>
            </div>
            <p>Your score: {{ $score }}</p>
            <p>Dealer's score: {{ $dealerScore }}</p>
        @endif
    </div>

    <div class="players-container">
        <p>Your hand:</p>
        <ul>
            @foreach ($hand as $index => $card)
                <li>{{ $card->getRank() }} of {{ $card->getSuit() }}</li>
            @endforeach
        </ul>
        @if (session('dealer') !== null)
            <p>Dealer's hand:</p>
            <ul>
                @foreach ($dealerHand as $index => $card)
                    @if($index != 0 && !$dealerIsStand)
                        <li>??</li>
                    @else
                        <li>{{ $card->getRank() }} of {{ $card->getSuit() }}</li>
                    @endif
                @endforeach
            </ul>
        @endif

        <div class="action-container">
            <form action="{{ url('player-action') }}" method="POST">
                @csrf
                <input type="hidden" id="playerName" name="playerName" value="{{
$playerName }}"></input>
                <input type="submit" name="action" value="hit"
                    @if($player->getIsStand())
                        disabled
                    @endif
                ></input>
                <input type="submit" name="action" value="stand"></input>
            </form>
        </div>
    </div>
</div>
```

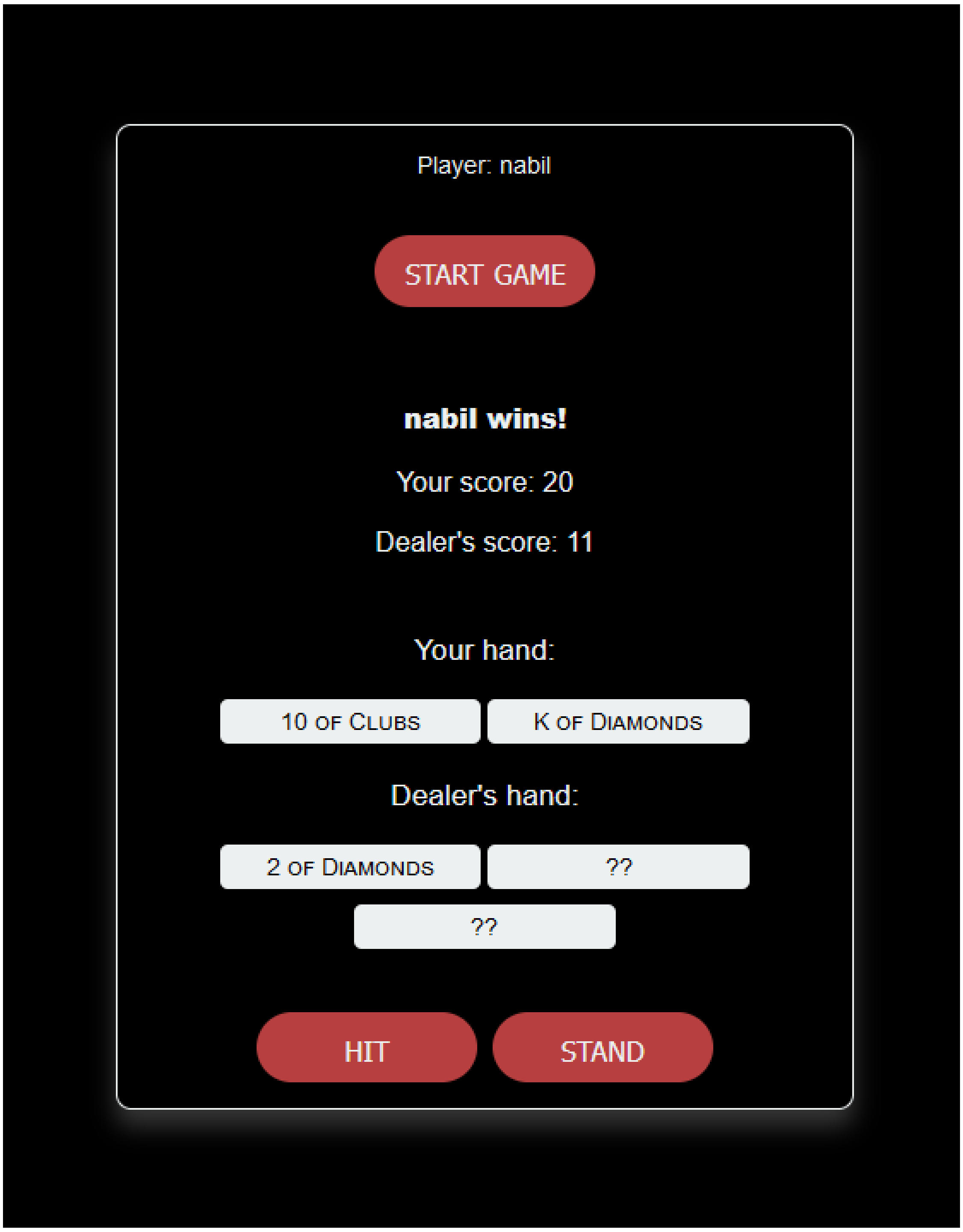
- Capturas:


USERNAME
nabil
LOGIN


Player: nabil
Your hand:
4 of Diamonds
10 of Spades
Dealer's hand:
8 of Diamonds
??
HIT
STAND


Player: nabil
START GAME
Dealer wins!
Your score: 23
Dealer's score: 10
Your hand:
4 of Diamonds
10 of Spades
9 of Hearts
Dealer's hand:
8 of Diamonds
??
HIT
STAND


Player: nabil
Your hand:
A of Hearts
6 of Spades
Dealer's hand:
7 of Clubs
??
??
HIT
STAND


Player: nabil
START GAME
nabil wins!
Your score: 20
Dealer's score: 11
Your hand:
10 of Clubs
K of Diamonds
Dealer's hand:
2 of Diamonds
??
??
HIT
STAND