Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное
учреждение высшего образования
“Национальный исследовательский университет ИТМО”

Факультет инфокоммуникационных технологий

ЛАБОРАТОРНАЯ РАБОТА №4

Разработка интерфейсов для выполнения CRUD-операций над базой данных средствами PHP по дисциплине: «Проектирование и реализация баз данных»

Выполнил студент:
Зайцев Кирилл Дмитриевич
Группа №K33402
Преподаватель:
Говорова Марина Михайловна

Санкт-Петербург
2023

Цель работы:

Овладеть практическими навыками разработки форм для вставки, выборки и редактирования данных. **Программное обеспечение:**

Среда разработки XAMPP 8 (PHP 8) или аналогичная, СУБД PostgreSQL 1X, pgadmin 4.

Практическое задание:

1. Изучить функции открытия соединения к базе данных средствами PHP.
2. Изучить основные функции для создания php-скрипта (на базе видеоуроков 1-7).
3. Создать сайт с использованием базовых возможностей PHP (в соответствии с содержанием видео-уроков 1-7)

Ход работы:

В работе реализованы CRUD-операции для таблицы customers. CRUD-операции были реализованы при помощь фреймворка PHP – Laravel. В Laravel есть возможность развертывания сервера для разработки, который базируется на XAMPP, он как раз и использовался в работе.

Рисунок 1 – запуск сервера приложения

В качестве БД использовалась PostgreSQL, работающая на localhost.

Laravel является MVC-фреймворком, поэтому для реализации требуется создать классы модели (Model), контроллера (Controller) и миграции для создания таблицы в бд, а так же представления (View).

Рисунок 2 - создание миграции, модели и контроллера.

Миграция:

В классе миграции описывается структура таблицы, поля, тип значений и ограничения.

Класс миграции:

```php
<?php

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

class CreateCustomersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('customers', function (Blueprint $table) {
            $table->bigInteger('passport')->primary();

            $table->string('full_name', 255);
            $table->string('address', 255);

            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('customers');
    }
}
```

Модель:

В классе модели описываются ее поля, первичный ключ.

Класс модели:

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Customer extends Model

    protected $primaryKey = 'passport';

    protected $fillable = ['passport', 'full_name', 'address'];

<
```

```
{

}
```

Контроллер:

В классе контроллера описываются методы, которые соответствуют определенным URI приложения.

Класс контроллера:

```php
<?php

namespace App\Http\Controllers;

use App\Models\Customer;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class CustomerController extends Controller
{
    public function index()
    {
        $customers = Customer::all();

        return view('customers.index', [
            'customers' => $customers
```

```
        ]);
    }

    public function find(Customer $customer)
    {
        return view('customers.find', [
            'customer' => $customer
        ]);
    }

    public function create()
    {
        return view('customers.create');
    }

    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'passport' => 'required|integer|digits:12|unique:customers',
            'full_name' => 'required|string',
            'address' =>  'required|string'
        ]);

        if ($validator->fails()) {
            return redirect()
                ->back()
                ->withErrors($validator)
                ->withInput();
        }

        Customer::create($validator->validated());

        return redirect('/customers');
    }

    public function edit(Customer $customer)
    {
        return view('customers.edit', [
                'customer' => $customer
            ]);
    }

    public function update(Request $request, Customer $customer)
    {
        $validator = Validator::make($request->all(), [
            'passport' => 'required|integer|digits:12|unique:customers',
            'full_name' => 'required|string',
            'address' =>  'required|string'
        ]);

        if ($validator->fails()) {
            return redirect()
                ->back()
                ->withErrors($validator)
                ->withInput();
        }

        $customer->update($validator->validated());

        return redirect('/customers');
    }

    public function destroy(Customer $customer)
    {
        $customer->delete();

        return redirect()->back();
```

```
    }
}
```

Метод index выводит все экземпляры клиентов.
Метод find находит конкретный экземпляр модели клента и выводит только его.
Метод create выводит форму для создания клиента.
Метод store получает данные из формы и создает нового клиента. Метод edit находит конкретный экземпляр клиента и выводит форму для редактирования его данных.
Метод update сохраняет изменения.
Метод destroy удаляет конкретного клиента по значению уникального ключа.

Файл routes.php:

Файл, в котором описываются все URI приложения:

```
<?php use
App\Http\Controllers\CustomerController; use
Illuminate\Support\Facades\Route;

Route::get('/customers', [CustomerController::class, 'index']);
Route::get('/customers/create', [CustomerController::class, 'create']);
Route::get('/customers/{customer}', [CustomerController::class, 'find']);
Route::post('/customers', [CustomerController::class, 'store']);
Route::get('/customers/{customer}/update', [CustomerController::class,
'edit']);
Route::put('/customers/{customer}', [CustomerController::class, 'update']);
Route::delete('customers/{customer}', [CustomerController::class, 'destroy']);
```

Файлы представлений:

Файл описывающий общую разметку приложения:

```
<ul>
        <li><a href="/customers">index</a></li>
    <li><a href="/customers/create">create</a></li> </ul>

<div class="container">
    @yield('content'
</div>
```

Представления отдельных страниц, расширяющих вышеуказанную разметку:
customers/index.blade.php:

```
        'layout'        )
@extends(

@section( 'content'
        <h1>index</h1>
        <ul>
```

```
            @foreach(            $customers as $customer
                  <li>
                              <p>{{ $customer->passport .            ': ' . $customer-
                              <p>{{ 'Address: ' .                    >full_name }}</p>
                      <form action="/customers/ >                   tomer-
                                                                     >addr
                                                                     ess
method="POST"                                                        }}</p
                                                                     > {{
                                                              type="h $cust
                                  @csrf                       idden"  omer-
                                  <input                      type="s >pass
                                  <input                      ubmit"  port
                                    > href= href=                     }}"
                                                                   {{
                                                                   {{
                 </form                                               name="_method"
                 <a                                         /customer value="DELETE" />
                 <a                                         /         value="Delete">
             </li>                                          /customer
            @endforeach                                     /          $customer->passpor
                                                                       $customer->passpor

                                                                      /update">Edit</a>
            </ul>

@endsection
```

customers/find.blade.php:

```
                 'layout'                             )
@extends(

                                                       { $customer-
@section(        'content'                             ssport }}</h2> $
                 <h1>show</h1> <h2>Customer #          stomer->full_name
                 <p>Name:    <p>Address:               </p>
                           {{                           $customer->address }}</p>
                              {{

@endsection
```

customers/create.blade.php:

```
        'layout'                   )
@extends(

@section('content'                 customers" method="POST">
        <h1>create</h
                                   assport:</p>
        <form action=
```

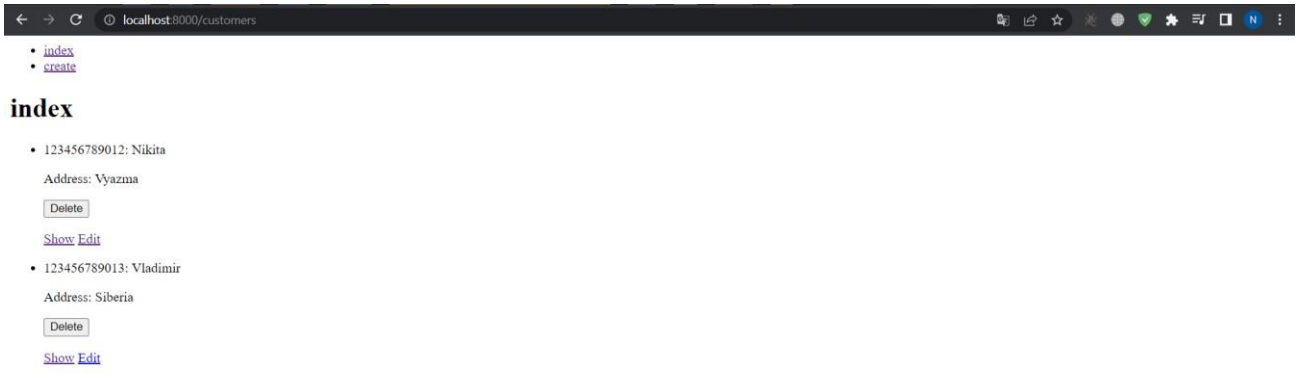
```
                                ="text"
                                   name="passport"/>
                                   </p>
                                ="text" name="full_name"/>
        @csrf                  ddress:</p>
        <p>            type     ="text" name="address"/>
        <input         Enter n  ="hidden" name="_method" value="POST"/>
        <p>                    pe="submit">Create new customer</button>
        <input <p>   Enter typ
        <input              typ
        <input
        <hr>
        <button orm>
    </f
       @ $errors->any()
       i
       f
       (
        {!!      implode('', $errors->all('<div>:!!}
    @endif
@endsection
```

customers/edit.blade.php:

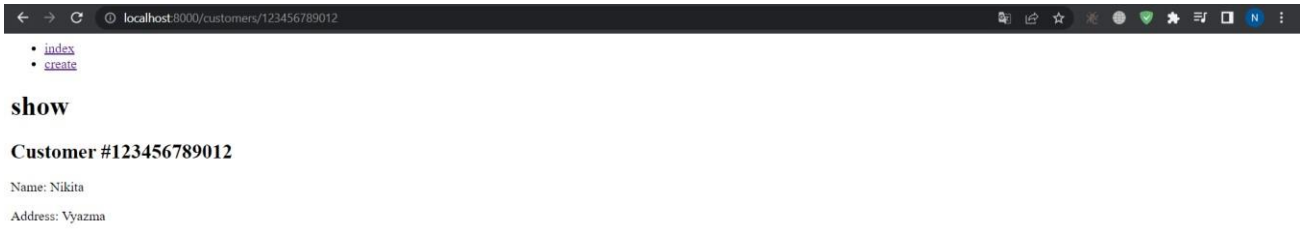
```
@extends('layout')
@section('content')
    <h1>edit</h1>

    <form action="/customers/{{ $customer->passport }}" method="POST">
@csrf
        <p>Passport:</p>
        <input type="text" name="passport" value="{{ $customer->passport }}"/>
<p>Name:</p>
        <input type="text" name="full_name" value="{{ $customer->full_name }}"/>
        <p>Address:</p>
        <input type="text" name="address" value="{{ $customer->address }}"/>
        <input type="hidden" name="_method" value="PUT"/>
        <hr>
        <input type="submit" value="Update Data ">
</form>
    @if($errors->any())
        {!! implode('', $errors->all('<div>:message</div>')) !!}
@endif
@endsection
```

Функционал приложения:

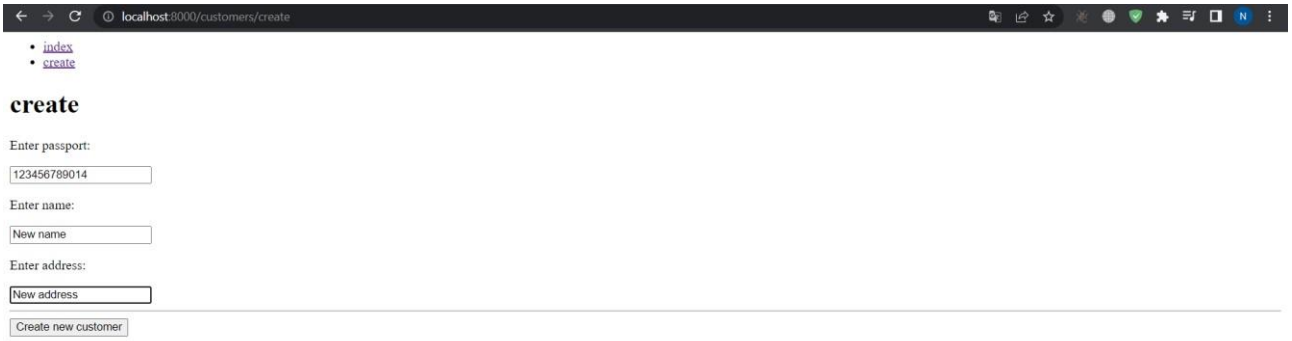
Страница выводящая всех клиентов:



Страница просмотра клиента:

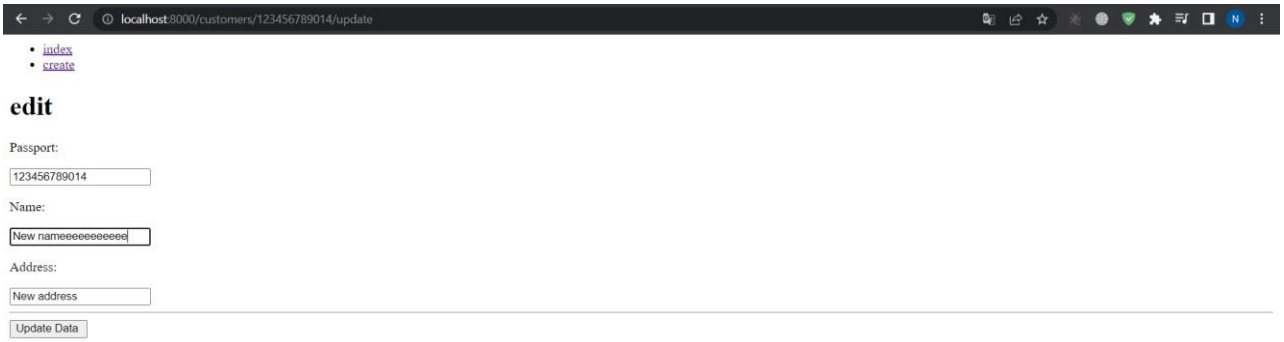


Страница добавления нового клиента:

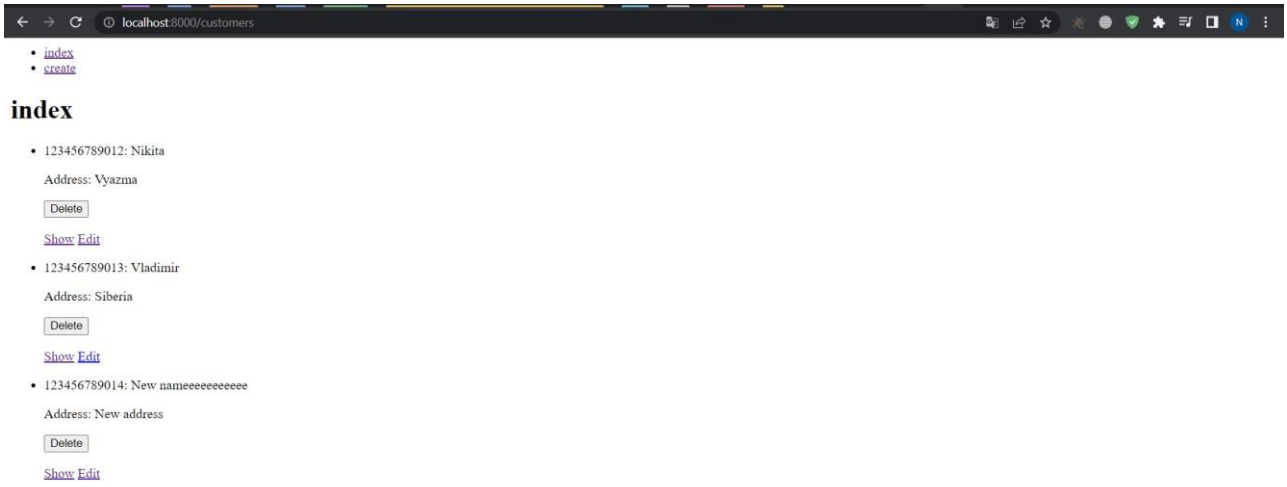


После создания мы попадаем на страницу со всеми клиентами:

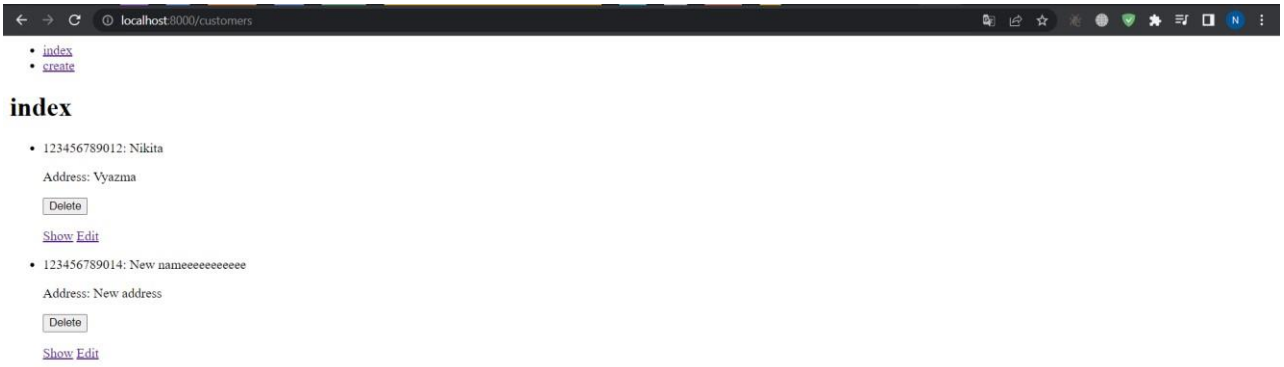
Страница редактирования клиента:



После редактирования мы попадаем на страницу со всеми клиентами:



Удаление клиента:



Выводы:

1. Были созданы CRUD-операции для таблицы customer согласно индивидуальному заданию