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 помошь 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).

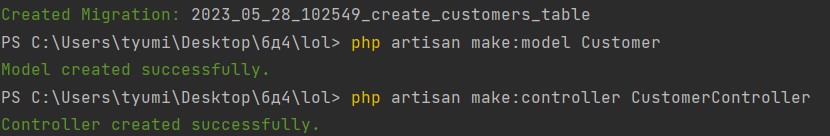


Рисунок 2 - создание миграции, модели и контроллера.

**Миграция:**

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

Класс миграции:

p

?ph

<

use

luminate

Il

\

Database

\

Migrations

\

Migration

;

use

Il

luminate

\

Database

\

Schema

\

Blueprint

;

use

Il

luminate

\

Support

\

Facades

\

Schema

;

class

CreateCustomersTable

extends

Migratio

n

{

*/\**

*\**

*.*

*\**

*Run the migrations*

*\**

*\**

***@return***

*d*

*voi*

*\**

*/*

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***

*d*

*voi*

*\**

*/*

public function

down

(

)

{

Schema::

dropIfExists

(

'customers'

)

;

}

}

**Модель:**

В классе модели описываются ее поля, первичный ключ.

Класс модели:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | ?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:

p

?ph

<

namespace

App

\

Http

\

Controllers

;

use

App

\

Models

\

Customer

;

use

Illuminate

\

Http

\

Request

;

use

Illuminate

\

Support

\

Facades

\

Validator

;

class

CustomerController

extends

Controlle

r

{

public function

index

(

)

{

$customers

Customer

::

=

*all*

()

;

return

view(

'customers.index'

,

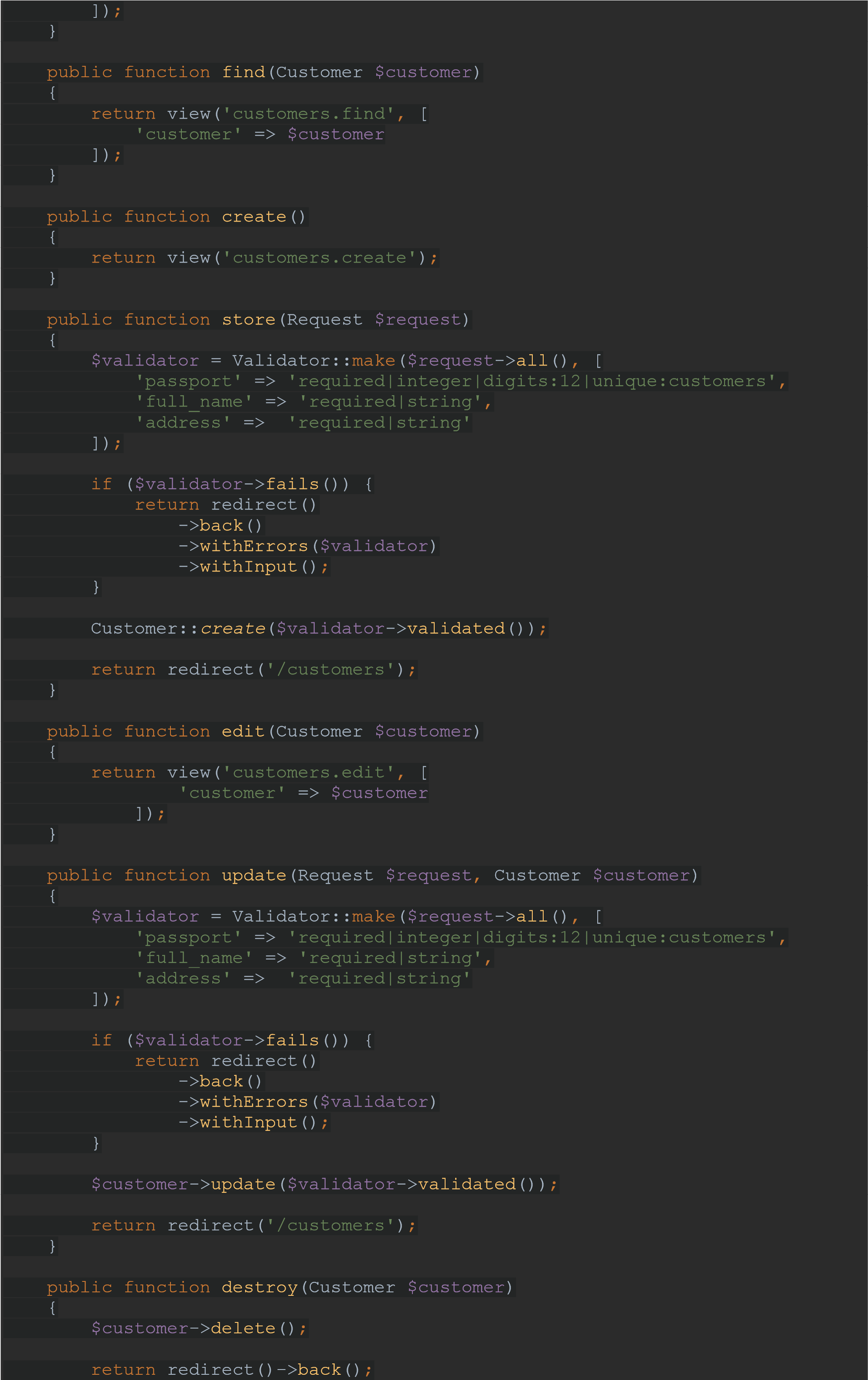
[

'customers'

=>

$customer

s



|  |  |  |
| --- | --- | --- |
| |  | | --- | | } | |  |
| } |

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']);

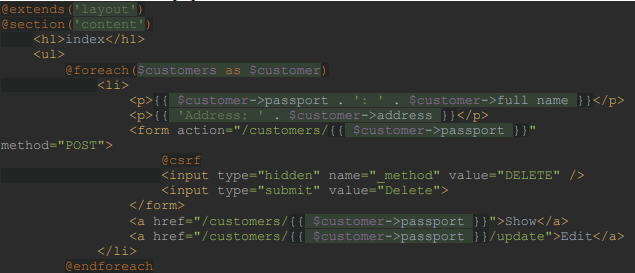
**Файлы представлений:**

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> |

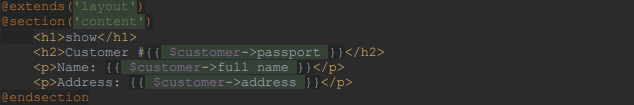
Представления отдельных страниц, расширяющих вышеуказанную разметку:

customers/index.blade.php:

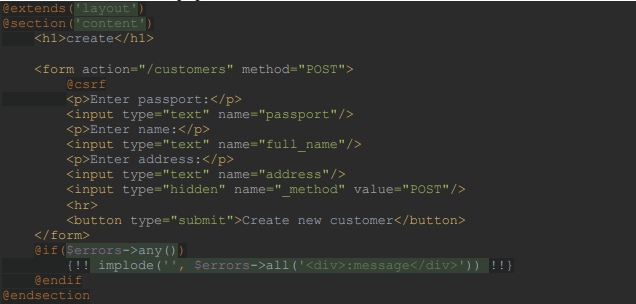




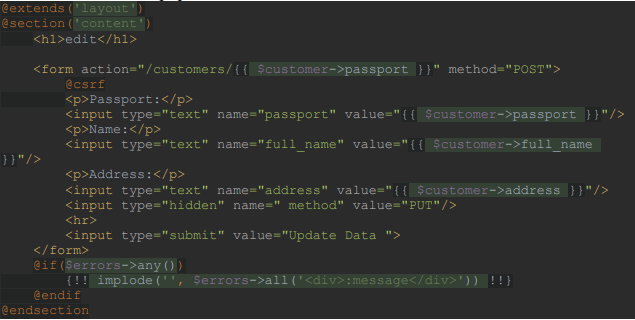
customers/find.blade.php:



customers/create.blade.php:

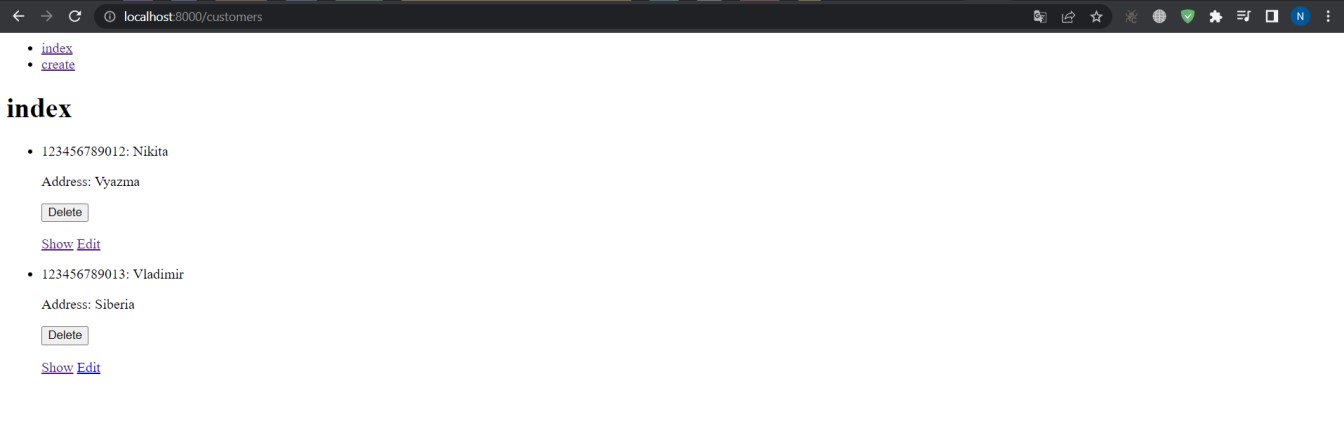


customers/edit.blade.php:

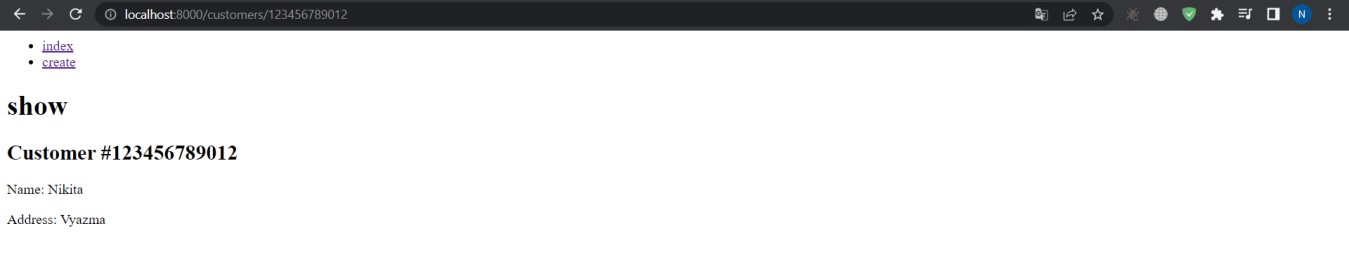


**Функционал приложения:**

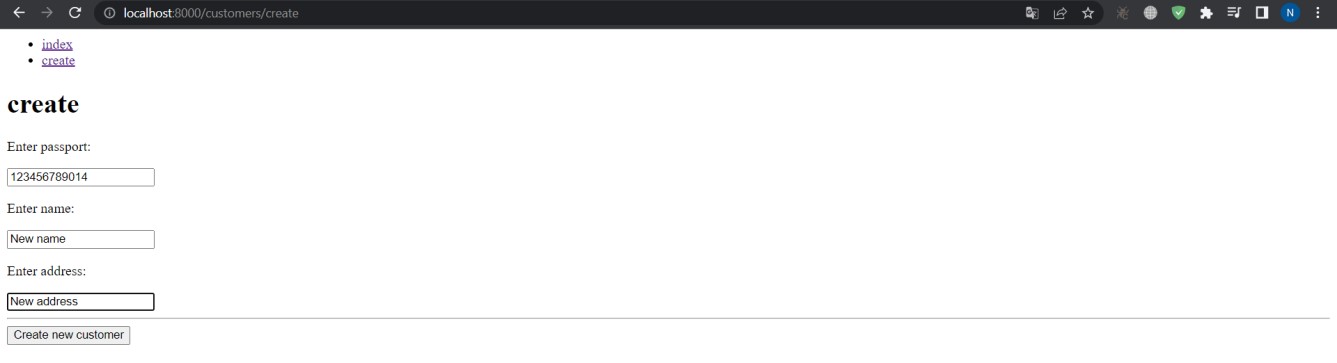
Страница выводящая всех клиентов:



Страница просмотра клиента:



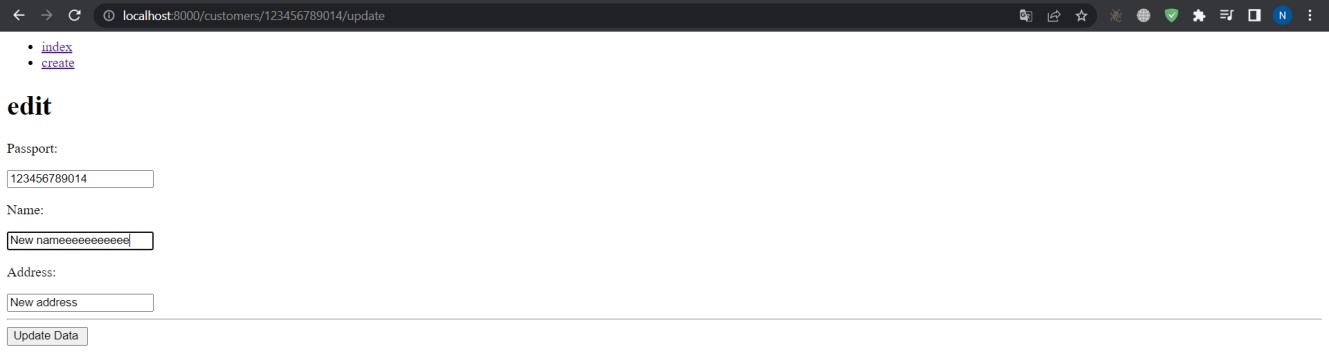
Страница добавления нового клиента:



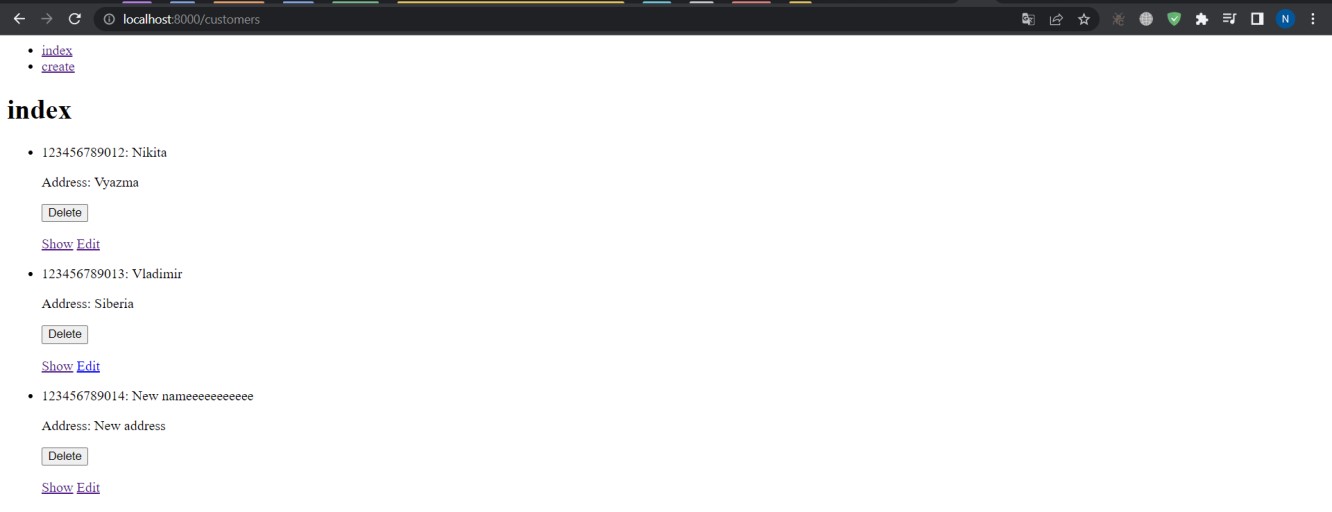
После создания мы попадаем на страницу со всеми клиентами:



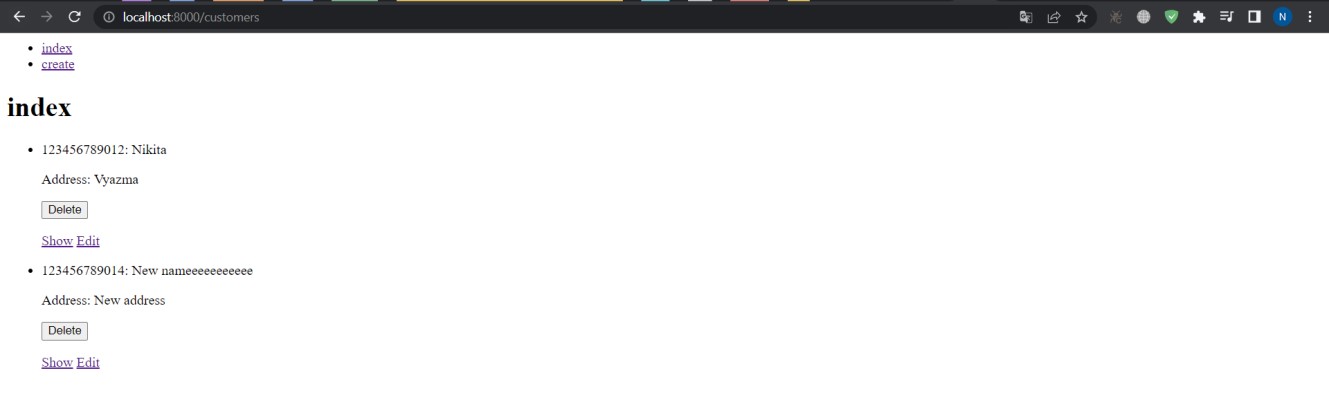
Страница редактирования клиента:



После редактирования мы попадаем на страницу со всеми клиентами:



Удаление клиента:



**Выводы:**

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