問題２

Chapter7問題1のindex.blade.phpで表示されるテーブルに対して、

ペジネーションをしていきましょう。

大量のテストデータについて

テキストには載っていませんが、Laravelには大量のテストデータを作成する為の

Factoryクラスがあります。



公式マニュアルを参考に、下記コマンドを実行すること。

php artisan make:factory PersonFactory

「database」内の「factories」フォルダの中に「PersonFactory.php」が作成された事を確認すること。

完成プログラム　ファイル名：PersonFactory.php

<?php

use App\Person;

use Faker\Generator as Faker;

$factory->define(Person::class, function (Faker $faker) {

    return [

        'name' => $faker->name,*//ランダムで名前を生成*

        'mail' => $faker->unique()->safeEmail,*//ランダムでユニークな使われていないメールを生成*

        'age' => $faker->numberBetween($min = 1, $max = 100),*//ランダムで1~100の値を生成*

    ];

});

「database」内の「seeds」フォルダの中の「PeopleTableSeeder.php」に対して、

先ほど作った『factory』と、生成したいダミーデータの数を追記します。

完成プログラム　ファイル名：PeopleTableSeeder.php

<?php

use Illuminate\Database\Seeder;

use Illuminate\Support\Facades\DB;

use App\Person;

class PeopleTableSeeder extends Seeder

{

    public function run()

    {

        $param = [

            'name' => 'taro',

            'mail' => 'taro@yamada.jp',

            'age' => 12,

        ];

        DB::table('people')->insert($param);

        $param = [

            'name' => 'hanako',

            'mail' => 'hanako@flower.jp',

            'age' => 34,

        ];

        DB::table('people')->insert($param);

        $param = [

            'name' => 'sachiko',

            'mail' => 'sachiko@happy.jp',

            'age' => 56,

        ];

        DB::table('people')->insert($param);

        factory(Person::class, 50)->create();*//今回は50個のテストデータを作成*

    }

}

シーダーファイルの登録をするために、

「database」内の「seeds」フォルダの中の「DaTableSeeder.php」を開き、追記をすること。

テキスト通りすすめていれば、こちらは追記されているはずです。

完成プログラム　ファイル名：DatabaseSeeder.php

<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

   public function run()

   {

       $this->call(RestdataTableSeeder::class);

       $this->call(PeopleTableSeeder::class); //無かった場合は追記すること。

       $this->call(SubjectTableSeeder::class);

   }

}

このままでは名前が英語表記になってしまうため、テストデータの日本語化をしていきます。

「config」フォルダ内の「app.php」を開き、追記をすること。

*/\**

*|----------------------------------------------------------------------*

*| Faker Setting*

*|----------------------------------------------------------------------*

*\*/*

    'faker\_locale' => env('DEV\_FAKER\_LOCALE', 'en\_US'),

「.env」ファイルを開き、追記をすること。

DEV\_FAKER\_LOCALE=ja\_JP

これにて完成です。

下記コマンドを実行していきましょう。

php artisan migrate:refresh

↓

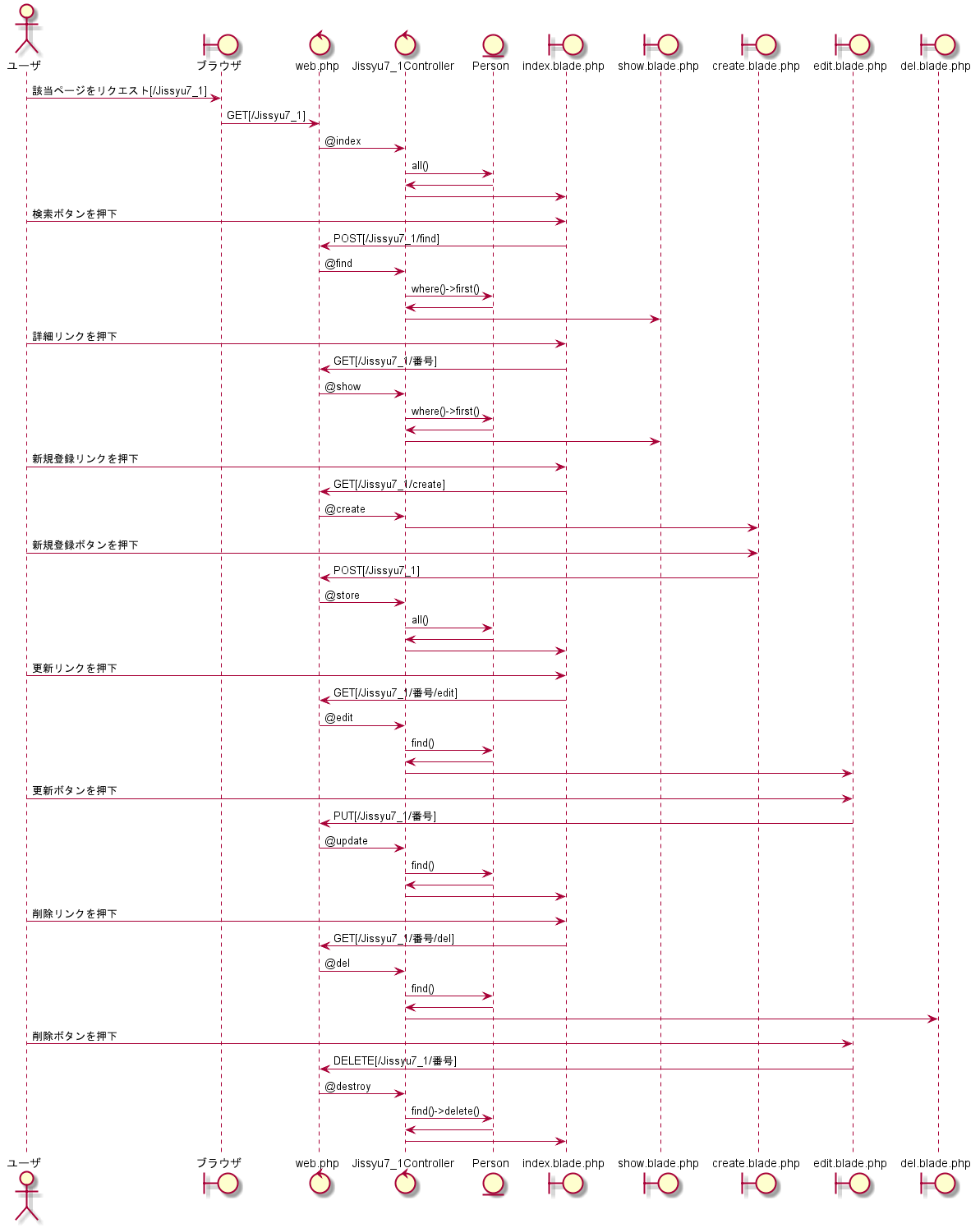
php artisan db:seed

成功すれば、データベース内にテストデータが生成されるはずです。



Chapter7問題1で作成したシーケンス図を利用すること。

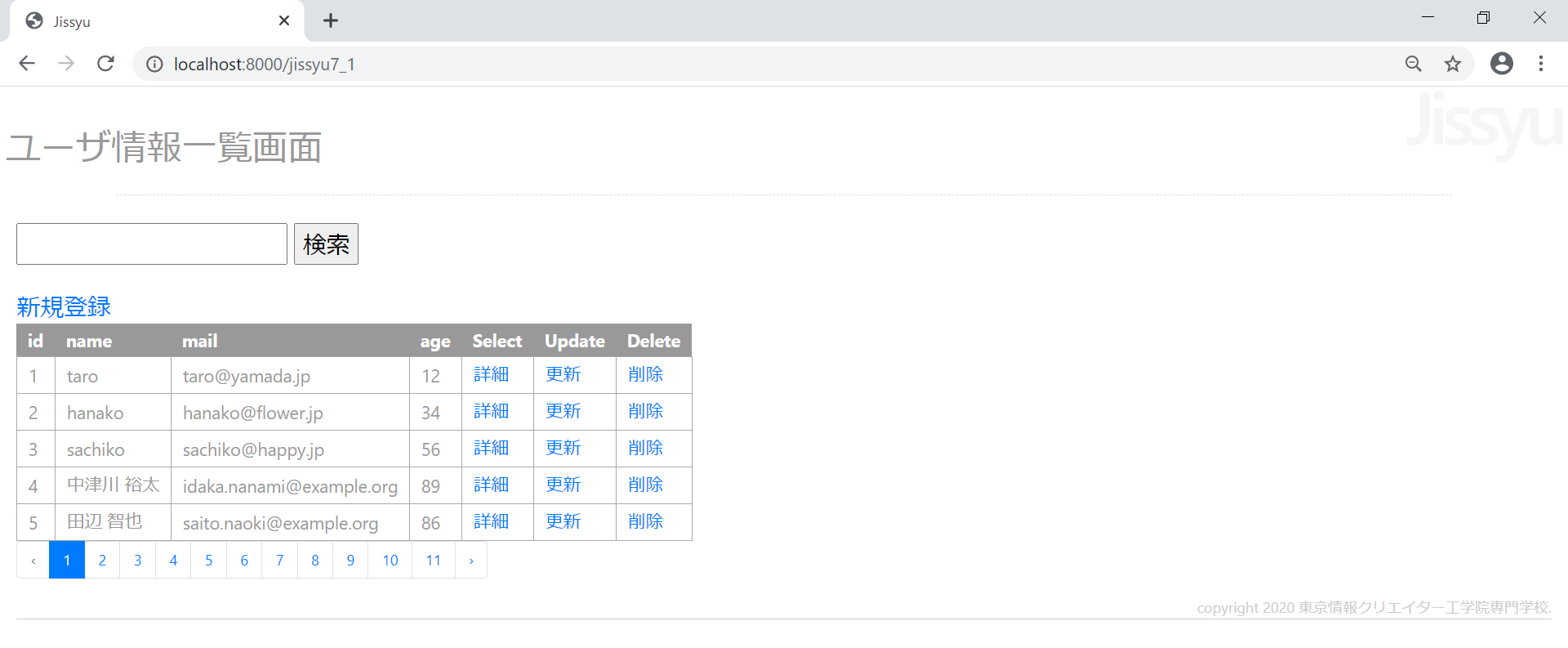
フォルダ名：「jissyu7\_1.wsd」



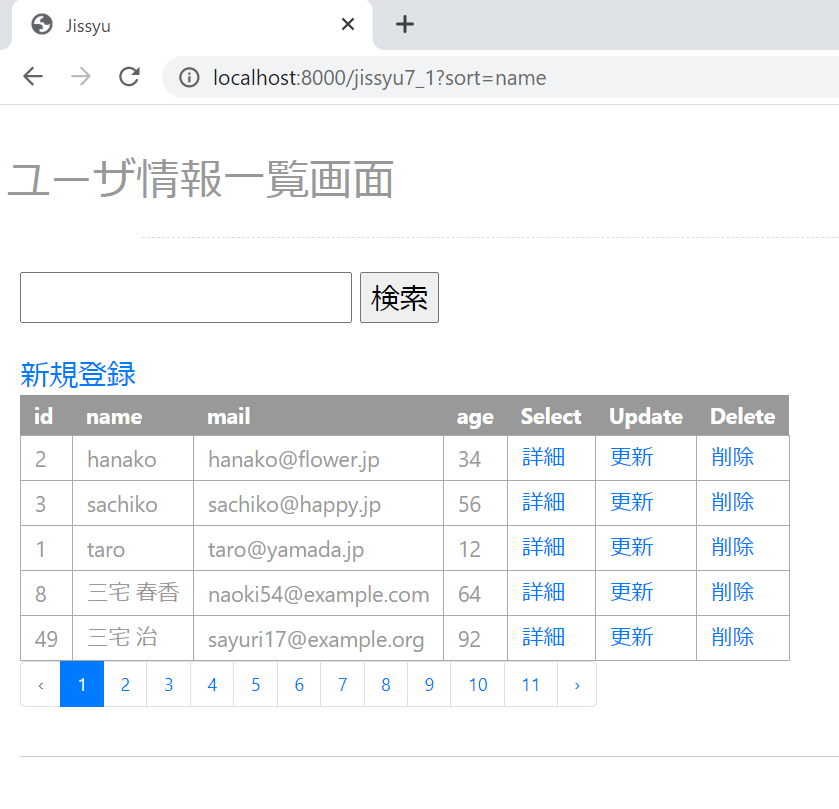
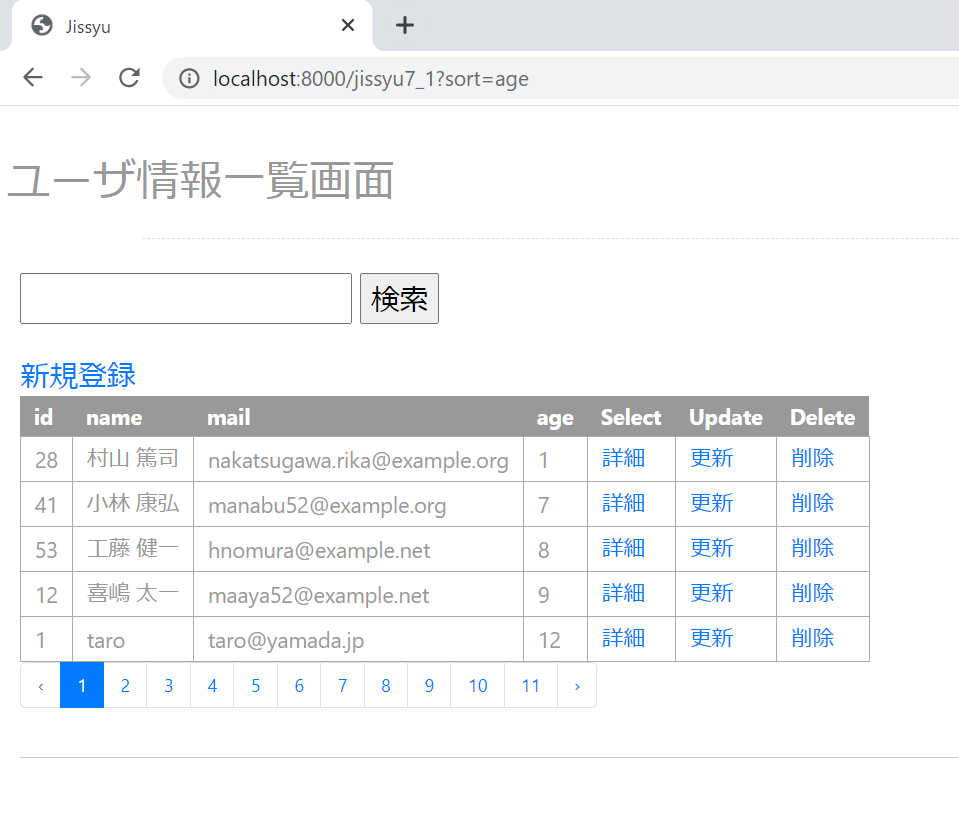
下記の結果になるようプログラムを作成すること。

実行結果

ユーザ情報一覧画面[GET送信/jissyu7\_1] idでソートされる



ユーザ情報一覧画面[nameリンク押下] [ageリンク押下]



layouts\jissyu.blade.php

下記のソースコードを<head>内に追記しておくこと。

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">

未完成プログラム　ファイル名：Jissyu7\_1Controller.php

<?php

namespace App\Http\Controllers;

use App\Person;

use Illuminate\Http\Request;

use Validator;

class Jissyu7\_1Controller extends Controller

{

    public function index(Request $request)

    {

*//$items = Person::all();*

*//$param = ['input' => '','items' => $items];*

        if ($request->has('sort')){

            $sort = \_\_\_(1)\_\_\_;

        }else{

            $sort = 'id';

        }

        $items = \_\_\_(2)\_\_\_;

        $param = ['input' => '','items' => $items,\_\_\_(3)\_\_\_];

        return view('jissyu7\_1.index', $param);

    }

    public function find(Request $request)

    {

        $rules = [

            'input' => 'required',

        ];

        $messages = [

            'input.required' => '文字を入力してください。',

        ];

        $validator = Validator::make($request->all(),$rules,$messages);

        if($validator->fails()){

            return redirect('/jissyu7\_1')

            ->withErrors($validator)

            ->withInput();

        }

        $item = Person::where('name',$request->input)->first();

        return view('jissyu7\_1.show', ['item' => $item]);

    }

    public function create()

    {

        return view('jissyu7\_1.create');

    }

    public function store(Request $request)

    {

        $this->validate($request, Person::$rules);

        $person = new Person;

        $form = $request->all();

        unset($form['\_token']);

        $person->fill($form)->save();

        return redirect('/jissyu7\_1');

    }

    public function show($id)

    {

        $item = Person::find($id);

        return view('jissyu7\_1.show', ['item' => $item]);

    }

    public function edit($id)

    {

        $item = Person::find($id);

        return view('jissyu7\_1.edit', ['item' => $item]);

    }

    public function update(Request $request, $id)

    {

        $this->validate($request, Person::$rules);

        $person = Person::find($id);

        $form = $request->all();

        unset($form['\_token']);

        $person->fill($form)->save();

        return redirect('/jissyu7\_1');

    }

    public function del($id)

    {

        $item = Person::find($id);

        return view('jissyu7\_1.del', ['item' => $item]);

    }

    public function destroy($id)

    {

        Person::find($id)->delete();

        return redirect('/jissyu7\_1');

    }

}

未完成プログラム　ファイル名：index.blade.php

※問題1はコメントにしてあります。

@extends('layouts.jissyu')

<style>

    .pagination { font-size:10pt; }

    .pagination li { display:inline-block }

    tr th a:link { color: white; }

    tr th a:visited { color: white; }

    tr th a:hover { color: white; }

    tr th a:active { color: white; }

 </style>

@section('title', 'Jissyu')

@section('menu\_title')

ユーザ情報一覧画面

@endsection

*{{--問題1*

*@section('content')*

*<form action="/jissyu7\_1/find" method="post">*

*@csrf*

*<input type="text" name="input" value="{{$input}}">*

*<input type="submit" value="検索">*

*</form>*

*@if($errors->has('input'))*

*<font color="red">{{$errors->first('input')}}</font><br>*

*@endif*

*<a href="/jissyu7\_1/create">新規登録</a>*

*<table>*

*<tr><th>Name(Age)</th><th>Mail</th><th>Select</th><th>Update</th><th>Delete</th></tr>*

*@foreach ($items as $item)*

*<tr>*

*<td>{{$item->getData()}}</td>*

*<td>{{$item->mail}}</td>*

*<td><a href="/jissyu7\_1/{{$item->id}}">詳細</a></td>*

*<td><a href="/jissyu7\_1/{{$item->id}}/edit">更新</a></td>*

*<td><a href="/jissyu7\_1/{{$item->id}}/del">削除</a></td>*

*<!--GET送信に注意。テキスト299Pを参考のうえ、作成すること。 -->*

*</tr>*

*@endforeach*

*</table>*

*@endsection*

*--}}*

*{{--問題2--}}*

@section('content')

    <form action="/jissyu7\_1/find" method="post">

    @csrf

    <input type="text" name="input" value="{{$input}}">

    <input type="submit" value="検索">

    </form>

    @if($errors->has('input'))

    <font color="red">{{$errors->first('input')}}</font><br>

    @endif

   <a href="/jissyu7\_1/create">新規登録</a>

   <table>

   <tr>

       <th><a href="\_\_\_(4)\_\_\_">id</a></th>

       <th><a href="\_\_\_(5)\_\_\_">name</a></th>

       <th><a href="\_\_\_(6)\_\_\_">mail</a></th>

       <th><a href="/jissyu7\_1?sort=age">age</a></th>

       <th>Select</th>

       <th>Update</th>

       <th>Delete</th>

    </tr>

   @foreach ($items as $item)

       <tr>

           <td>\_\_\_(7)\_\_\_</td>

           <td>{{$item->name}}</td>

           <td>{{$item->mail}}</td>

           <td>{{$item->age}}</td>

           <td><a href="/jissyu7\_1/{{$item->id}}">詳細</a></td>

           <td><a href="/jissyu7\_1/{{$item->id}}/edit">更新</a></td>

           <td><a href="/jissyu7\_1/{{$item->id}}/del">削除</a></td>

*<!--GET送信に注意。テキスト299Pを参考のうえ、作成すること。 -->*

       </tr>

   @endforeach

   </table>

   {{\_\_\_(8)\_\_\_}}

@endsection

@section('footer')

copyright 2020 東京情報クリエイター工学院専門学校.

@endsection