Министерство образования и науки Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение науки и высшего образования**

«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

**Институт информационных технологий и радиотехники (ИИТР)**

**Лабораторная работа № 1**

**по дисциплине**

**«Распределённые программные системы»**

Тема «Разработка веб-приложений на базе Spring MVC»

Выполнила:

ст. гр. ИСТ-117

Шумейко Д.С.

Принял:

Макаров С.А.

Владимир, 2020

**Цель работы:**

Познакомиться с принципами построения веб-приложений с использованием шаблона MVC и на базе принципов REST, получить практические навыки реализации MVCприложения на платформе Spring MVC.

**Backend часть:**

Модель:

package com.lab02.myapp.model;  
  
import javax.persistence.\*;  
import com.fasterxml.jackson.annotation.JsonFormat;  
import com.fasterxml.jackson.annotation.JsonProperty;  
  
import java.math.BigDecimal;  
import java.time.LocalDate;  
  
  
@Entity  
@Table(name = "person")  
public class Person {  
 @Id  
 @GeneratedValue(strategy = GenerationType.AUTO)  
 private long id;  
 private String name;  
 private String email;  
 @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-MM-dd")  
 @JsonProperty(value = "birthday")  
 private LocalDate birthDay;  
 private BigDecimal cash;  
  
 public Person() {  
 }  
  
 public Person(long id, String name, String email, LocalDate birthDay, BigDecimal cash) {  
 this.id = id;  
 this.name = name;  
 this.email = email;  
 this.birthDay = birthDay;  
 this.cash = cash;  
 }  
  
 public long getId() {  
 return id;  
 }  
  
 public void setId(long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public LocalDate getBirthDay() {  
 return birthDay;  
 }  
  
 public void setBirthDay(LocalDate birthDay) {  
 this.birthDay = birthDay;  
 }  
  
 public BigDecimal getCash() {  
 return cash;  
 }  
  
 public void setCash(BigDecimal cash) {  
 this.cash = cash;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 Person person = (Person) o;  
  
 if (id != person.id) return false;  
 if (!name.equals(person.name)) return false;  
 return birthDay.equals(person.birthDay);  
 }  
  
 @Override  
 public int hashCode() {  
 int result = (int) (id ^ (id >>> 32));  
 result = 31 \* result + name.hashCode();  
 return result;  
 }  
}

DAO класс:

package com.lab02.myapp.dao;  
  
import com.lab02.myapp.model.Person;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface PersonRepository extends JpaRepository<Person, Long> {  
  
}

Сервис:

package com.lab02.myapp.services;  
  
import com.lab02.myapp.dao.PersonRepository;  
import com.lab02.myapp.model.Person;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.\*;  
import java.util.concurrent.atomic.AtomicLong;  
  
@Service  
public class PersonServiceImpl implements PersonService {  
  
 private final PersonRepository personRepository;  
  
 private static final Map<Long, Person> *CLIENT\_REPOSITORY\_MAP* = new HashMap<>();  
  
 private static final AtomicLong *CLIENT\_ID\_HOLDER* = new AtomicLong();  
  
 @Autowired  
 public PersonServiceImpl(PersonRepository personRepository) {  
 this.personRepository = personRepository;  
 }  
  
 @Override  
 public Person create(Person person) {  
 return personRepository.save(person);  
 }  
  
 @Override  
 public Person update(Long id, Person person) {  
 Person oldPerson = getById(id);  
 if (oldPerson != null) {  
 if (person.getBirthDay() != null) oldPerson.setBirthDay(person.getBirthDay());  
 if (person.getCash() != null) oldPerson.setCash(person.getCash());  
 if (person.getEmail() != null) oldPerson.setEmail(person.getEmail());  
 if (person.getName() != null) oldPerson.setName(person.getName());  
 return personRepository.save(oldPerson);  
 }  
 return null;  
 }  
  
 @Override  
 public boolean delete(Long id) {  
 personRepository.deleteById(id);  
 return getById(id) == null;   
 }  
  
 @Override  
 public Person getById(Long id) {  
 return personRepository.findById(id).orElse(null);  
 }  
  
 @Override  
 public List<Person> getAll() {  
 return personRepository.findAll();  
 }  
}

Контроллер:

package com.lab02.myapp.controller;  
  
import com.lab02.myapp.model.Person;  
import com.lab02.myapp.services.PersonService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/api/lab02/")  
public class PersonController {  
  
 private final PersonService personService;  
  
 @Autowired  
 public PersonController(PersonService personService) {  
 this.personService = personService;  
 }  
  
 @GetMapping  
 public ResponseEntity<List<Person>> getAll() {  
 return new ResponseEntity<>(personService.getAll(), HttpStatus.*OK*);  
 }  
  
 @GetMapping(value = "{id}")  
 public ResponseEntity<Person> getById(@PathVariable(name = "id") Long id) {  
 final Person person = personService.getById(id);  
 return person != null  
 ? new ResponseEntity<>(person, HttpStatus.*OK*)  
 : new ResponseEntity<>(HttpStatus.*NOT\_FOUND*);  
 }  
  
 @DeleteMapping(value = "delete/{id}")  
 public ResponseEntity delete(@PathVariable(name = "id") Long id) {  
 return personService.delete(id)  
 ? new ResponseEntity(HttpStatus.*OK*)  
 : new ResponseEntity(HttpStatus.*NOT\_MODIFIED*);  
 }  
  
 @PutMapping(value = "update/{id}")  
 public ResponseEntity<Person> update(@PathVariable(name = "id") Long id, @RequestBody Person person) {  
 Person updated = personService.update(id, person);  
 return updated != null  
 ? new ResponseEntity<>(updated, HttpStatus.*OK*)  
 : new ResponseEntity<>(HttpStatus.*NOT\_MODIFIED*);  
 }  
  
 @PostMapping(value = "create")  
 public ResponseEntity<Person> create(@RequestBody Person person) {  
 return new ResponseEntity<>(personService.create(person), HttpStatus.*CREATED*);  
 }  
}

**Frontend часть:**

Модель пользователя:

export interface User {

  id?: number;

  name?: string;

  email?: string;

  cash?: number;

  birthday?: Date;

}

Сервис для взаимодействия с api:

import { Injectable } from '@angular/core';

import { Observable } from 'rxjs';

import { HttpClient } from '@angular/common/http';

import { User } from './user';

const apiUrl = 'http://localhost:8080/api/lab02/';

@Injectable({

  providedIn: 'root'

})

export class ApiService {

  header = {

    'Content-Type': 'application/json'

  };

  constructor(private http: HttpClient) { }

  getall(): Observable<User[]> {

    return this.http.get<User[]>(apiUrl);

  }

  getById(id: number): Observable<User> {

    return this.http.get<User>(apiUrl + id);

  }

  create(user: User): Observable<any> {

    return this.http.post<any>(`${apiUrl}create`, user);

  }

  update(id: number, user: User): Observable<any> {

    console.log(user);

    return this.http.put<any>(`${apiUrl}update/${id}`, user);

  }

  delete(id: number): Observable<any> {

    return this.http.delete(`${apiUrl}delete/${id}`);

  }

}

Компонент:

import { Component } from '@angular/core';

import CustomStore from 'devextreme/data/custom\_store';

import { User } from './user';

import { ApiService } from './api.service';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.less'],

  providers: [ApiService]

})

export class AppComponent {

  dataSource: any;

  user: User;

  rowKey: number;

  constructor(private apiService: ApiService) {

    this.dataSource = new CustomStore({

      key: 'id',

      load: () => this.apiService.getall().toPromise(),

      insert: (values) => this.apiService.create(values).toPromise(),

      update: (key, values) => this.apiService.update(key, values).toPromise(),

      remove: (key) => this.apiService.delete(key).toPromise()

    });

  }

  onFocusedRowChanged(event: any) {

    this.rowKey = event.row.data.id;

    this.apiService.getById(event.row.data.id).subscribe(user => this.user = user);

  }

}

Разметка компонента:

<dx-data-grid id="grid"

  [dataSource]="dataSource"

  [showBorders]="true"

  [focusedRowEnabled]="true"

  (onFocusedRowChanged)="onFocusedRowChanged($event)">

  <dxo-scrolling mode="virtual"></dxo-scrolling>

  <dxo-editing mode="popup"

    [allowAdding]="true"

    [allowUpdating]="true"

    [allowDeleting]="true">

    <dxo-popup

      [showTitle]="true"

      [width]="300"

      [height]="350"

      [position]="{ my: 'top', at: 'top', of: 'window' }">

    </dxo-popup>

    <dxo-form>

      <dxi-item itemType="group"

        [colCount]="1"

        [colSpan]="2">

        <dxi-item dataField="name"></dxi-item>

        <dxi-item dataField="email"></dxi-item>

        <dxi-item dataField="cash"></dxi-item>

        <dxi-item dataField="birthday"></dxi-item>

      </dxi-item>

    </dxo-form>

  </dxo-editing>

  <dxi-column dataField="name"

    dataType="string">

  </dxi-column>

  <dxi-column dataField="email"

    dataType="string">

  </dxi-column>

  <dxi-column dataField="cash"

    dataType="number">

  </dxi-column>

  <dxi-column dataField="birthday"

    dataType="date">

  </dxi-column>

</dx-data-grid>

<div class="task-info" \*ngIf="!!rowKey">

    <p>{{user.name}}</p>

    <p>{{user.cash | currency}}</p>

</div>

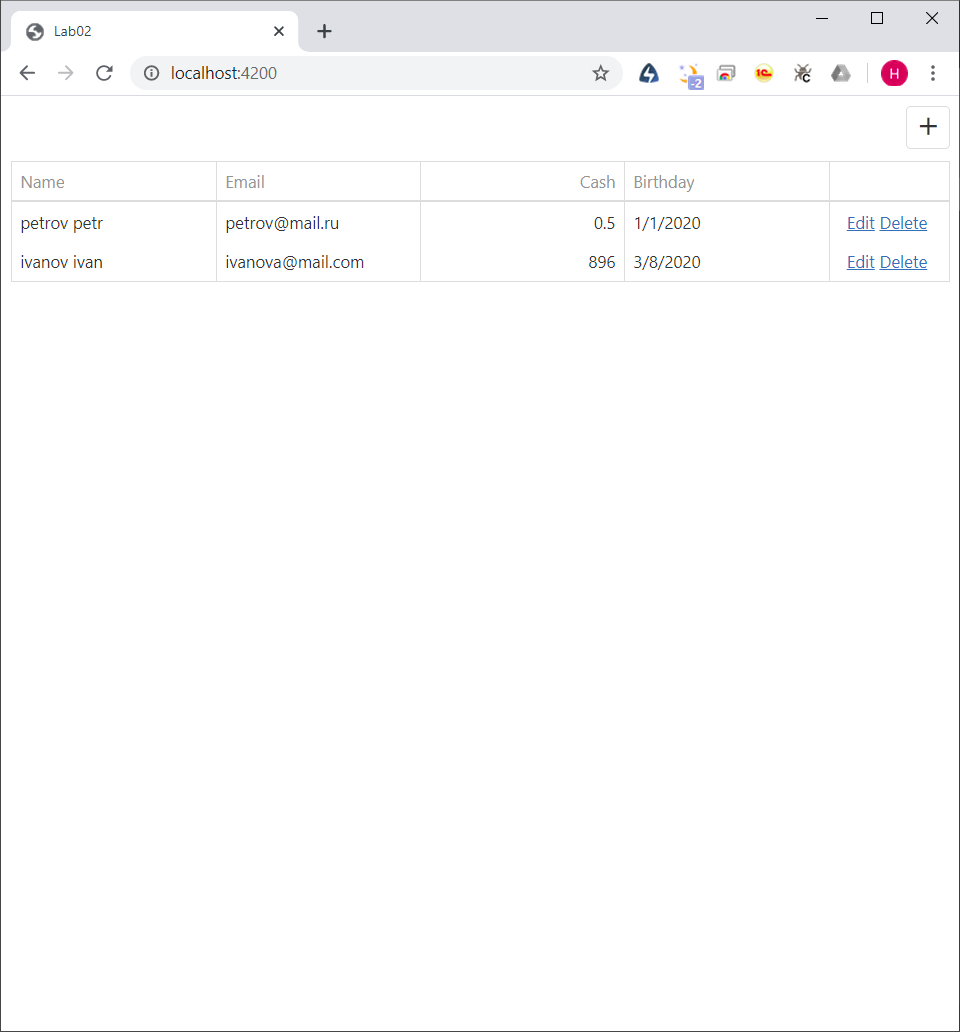


Рисунок 1 – Список всех пользователей (метод getAll)

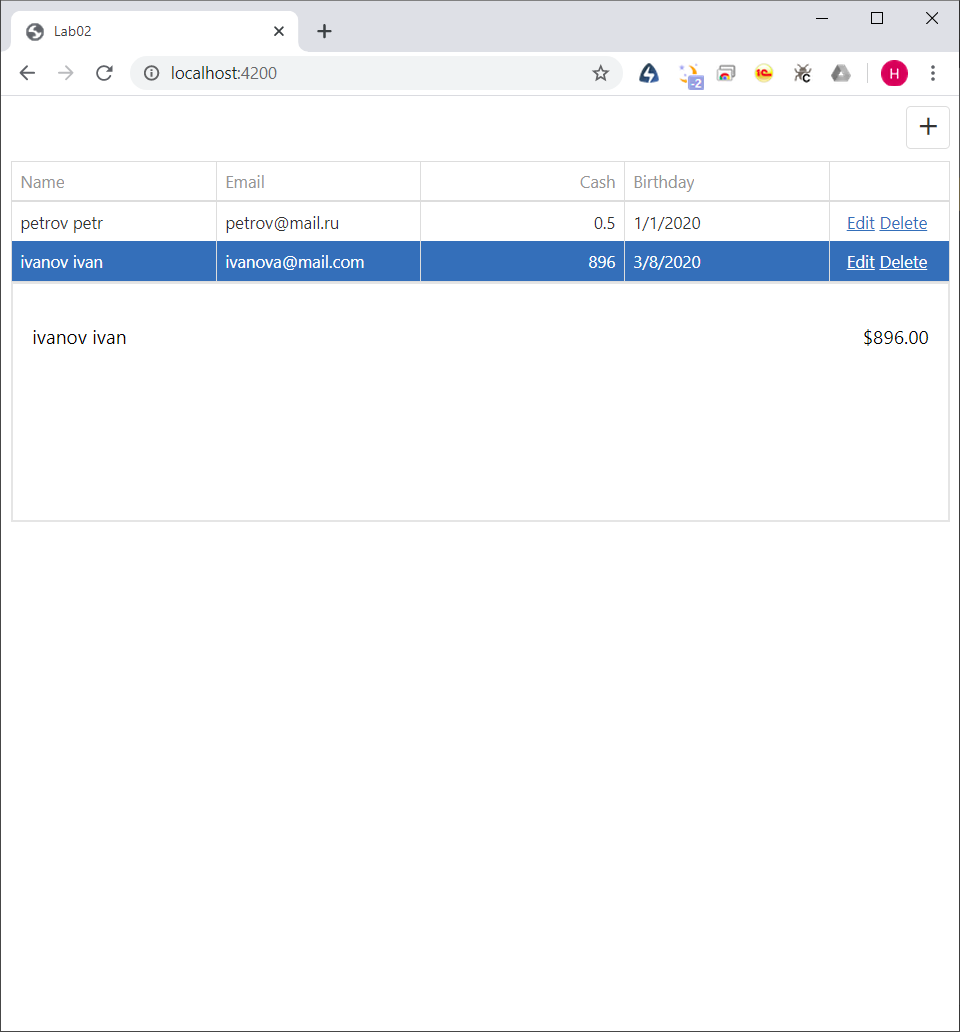


Рисунок 2 – Информация о конкретном пользователе (метод getById)

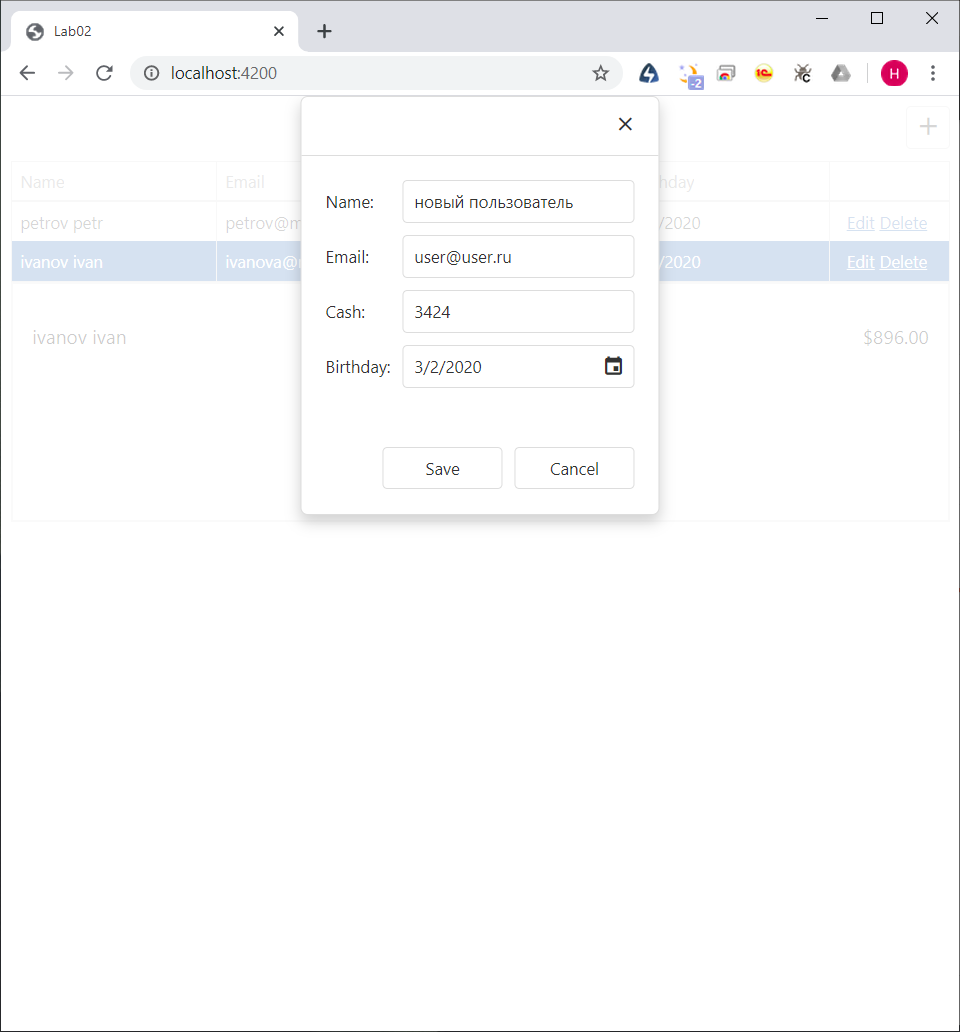


Рисунок 3 – Создание нового пользователя (метод create)

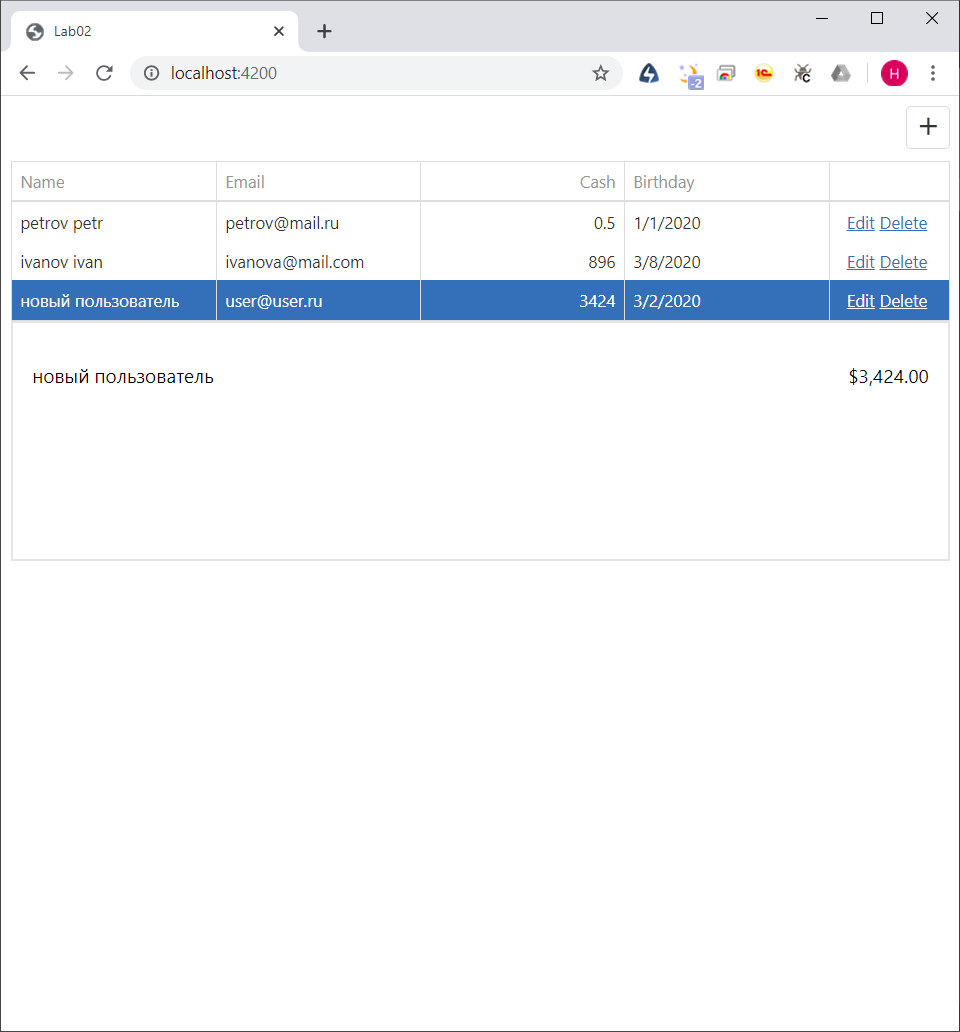


Рисунок 4 – Список всех пользователей

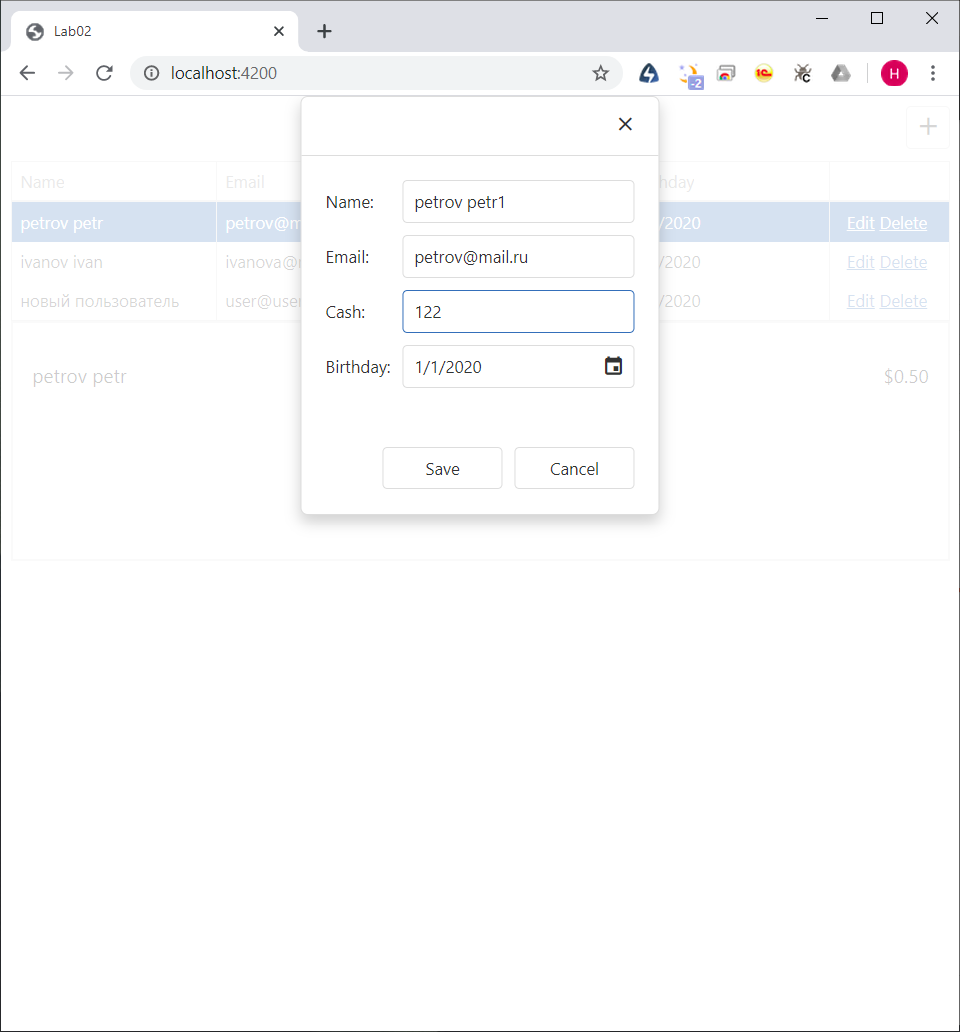


Рисунок 5 – Редактирование пользователя (метод update)

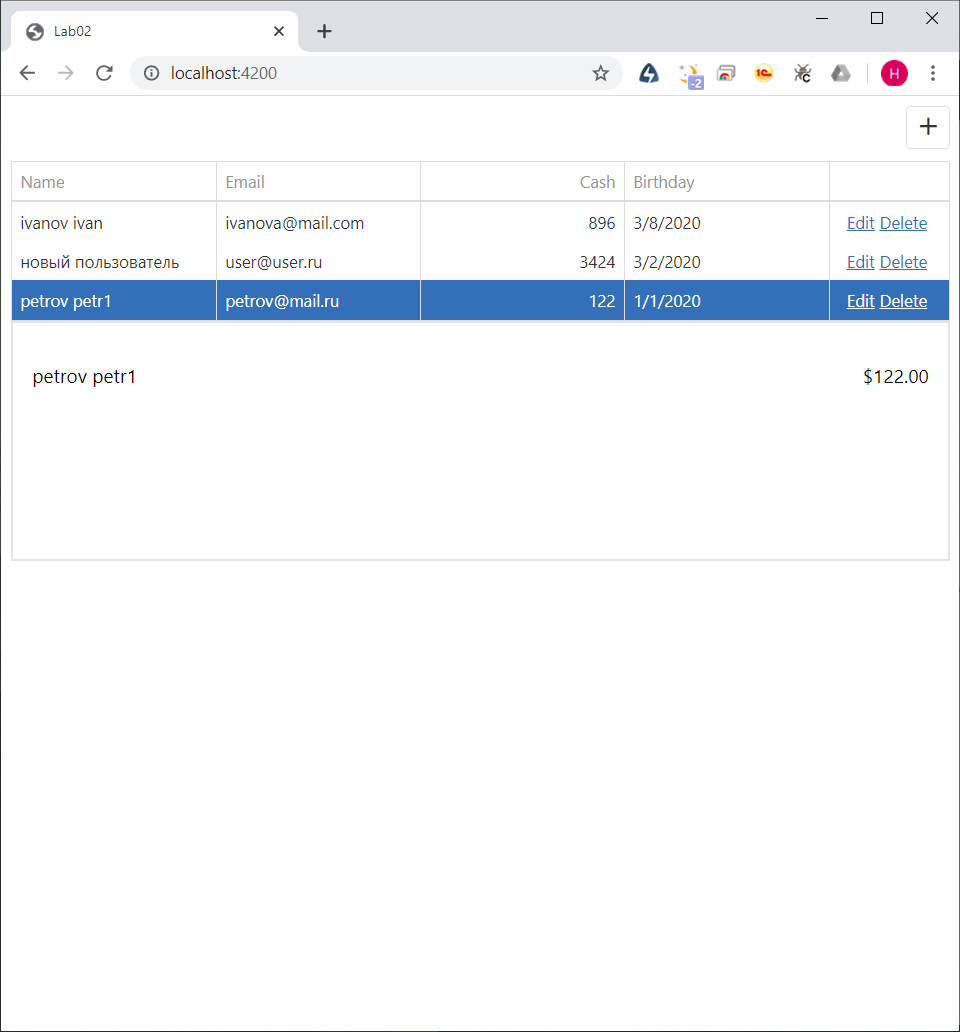


Рисунок 6 – Список всех пользователей

**Выводы:**

В ходе выполнения данной лабораторной работы произошло знакомство с принципами построения веб-приложений с использованием шаблона MVC и на базе принципов REST, получены практические навыки реализации MVC приложения на платформе Spring MVC.