Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Кафедра информатики и прикладной математики

Дисциплина «Программирование интернет-приложений»
Лабораторная работа №4

Вариант 3814

Выполнили:
Съестов Дмитрий Вячеславович
Давлетшин Рушан Рамилевич
Группа P3217

Преподаватель:
Николаев Владимир Вячеславович

Санкт-Петербург
2018

Задание

Переписать приложение из предыдущей лабораторной работы с использованием следующих технологий:

- Уровень back-end должен быть основан на Java EE (необходимо использовать EJB).
- Уровень front-end должен быть построен на Angular 2 (Angular 3) с использованием обычных полей ввода HTML
- Взаимодействие между уровнями back-end и front-end должно быть организовано посредством REST API.

Приложение по-прежнему должно включать в себя 2 страницы - стартовую и основную страницу приложения. Обе страницы приложения должны быть адаптированы для отображения в 3 режимах:

- "Десктопный" - для устройств, ширина экрана которых равна или превышает 1223 пикселей.
- "Планшетный" - для устройств, ширина экрана которых равна или превышает 644, но меньше 1223 пикселей.
- "Мобильный"- для устройств, ширина экрана которых меньше 644 пикселей.

Стартовая страница должна содержать следующие элементы:

- "Шапку", содержащую ФИО студента, номер группы и номер варианта.
- Форму для ввода логина и пароля. Информация о зарегистрированных в системе пользователях должна храниться в отдельной таблице БД (пароль должен храниться в виде хэш-суммы). Доступ неавторизованных пользователей к основной странице приложения должен быть запрещён.

Основная страница приложения должна содержать следующие элементы:

- Набор полей ввода для задания координат точки и радиуса области в соответствии с вариантом задания: Select {'-5','-4','-3','-2','-1','0','1','2','3'} для координаты по оси X, Text (-5 ... 3) для координаты по оси Y, и Select {'-5','-4','-3','-2','-1','0','1','2','3'} для задания радиуса области. Если поле ввода допускает ввод заведомо некорректных данных (таких, например, как буквы в координатах точки или отрицательный радиус), то приложение должно осуществлять их валидацию.
- Динамически обновляемую картинку, изображающую область на координатной плоскости в соответствии с номером варианта и точки, координаты которых были заданы пользователем. Клик по картинке должен инициировать сценарий, осуществляющий определение координат новой точки и отправку их на сервер для проверки её попадания в область. Цвет точек должен зависить от факта попадания / непопадания в область. Смена радиуса также должна инициировать перерисовку картинки.
- Таблицу со списком результатов предыдущих проверок.
- Кнопку, по которой аутентифицированный пользователь может закрыть свою сессию и вернуться на стартовую страницу приложения.

Дополнительные требования к приложению:

- Все результаты проверки должны сохраняться в базе данных под управлением СУБД PostgreSQL.
- Для доступа к БД необходимо использовать JPA.

Point.java

```
@Entity
@Table(name = "points", schema = "s225116")
public class Point
{
    private int id;
    private double x;
    private double y;
    private double r;
    private boolean hit;

    public Point() { }
    public Point(double x, double y, double r)
    {
        this.x = x;
        this.y = y;
        this.r = r;
    }

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    @Basic
    @Column(name = "x", nullable = false)
    public double getX() { return x; }
    public void setX(double x) { this.x = x; }
```

```
    @Basic
    @Column(name = "y", nullable = false)
    public double getY() { return y; }
    public void setY(double y) { this.y = y;}

    @Basic
    @Column(name = "r", nullable = false)
    public double getR() { return r; }
    public void setR(double r) { this.r = r; }

    @Basic
    @Column(name = "hit", nullable = false)
    public boolean isHit() { return hit; }
    public void setHit(boolean hit) { this.hit = hit; }
}
```

PointBean.java

```
@Stateful
@SessionScoped
@Path("/point")
public class PointBean
{
    private PointService service = new PointService();

    private double distance(Point a, Point b)
    {
        return Math.sqrt(Math.pow(a.getX() - b.getX(), 2) + Math.pow(a.getY() -
b.getY(), 2));
    }
```

```
    private double triangleArea(Point a, Point b, Point c)
    {
        double ab = distance(a, b);
        double bc = distance(b, c);
        double ca = distance(c, a);
        double p = (ab + bc + ca) / 2;
        return Math.sqrt(p * (p - ab) * (p - bc) * (p - ca));
    }

    private boolean checkHit(Point p)
    {
        Point center = new Point(0, 0, 0);

        //Квадрат R x R в II четверти
        if (p.getX() >= -p.getR() && p.getX() <= 0 && p.getY() >= 0 && p.getY() <=
p.getR())
            return true;

        //сектор радиуса R/2 в III четверти
        if (p.getX() <= 0 && p.getY() <= 0 && distance(center, p) <= p.getR() / 2)
            return true;

        ///треугольник в IV четверти
        Point a = new Point(p.getR() / 2, 0, 0);
        Point b = new Point(0, -p.getR() / 2, 0);
        double abc = triangleArea(a, b, center);
        double abp = triangleArea(a, b, p);
        double acp = triangleArea(a, p, center);
        double bcp = triangleArea(p, b, center);

        return Math.abs(abp + acp + bcp - abc) < 1E-5;
    }
```

```
@POST
@Path("/check")
public void check(@FormParam("X") double x, @FormParam("Y") double y, FormParam("R")
   double r, @Context HttpServletRequest req, @Context HttpServletResponse resp)
{
    try
    {
        Point point = new Point(x, y, r);
        boolean hit = checkHit(point);
        point.setHit(hit);

        service.savePoint(point);

        List<Point> list = (List<Point>) req.getSession().getAttribute("points");
        list.add(point);

        resp.sendRedirect(UserBean.APP_NAME + "main");
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

@GET
@Path("/getpoints")
public List<Point> getPoints(@Context HttpServletRequest req)
{
    return service.getAllPoints();
}
```

```
    @GET
    @Path("/update")
    public void updateRadius(@QueryParam("r") double r, @Context HttpServletRequest req)
    {
        List<Point> points = service.getAllPoints();

        for(Point p : points)
        {
            p.setR(r);
            boolean hit = checkHit(p);
            service.updatePoint(p.getId(), r, hit);
        }
    }
}
```

Компонент главной страницы

```
<form id="form" role="form" action="rest/point/check"
  onsubmit="return validateInput(this);" (submit)="onFormSubmit()" method="post">
  <div class="form-group row">
    <div class="col-xs-3 col-sm-3 col-lg-3"></div>
    <div class="col-xs-2 col-sm-2 col-lg-2">
      <label id="labelX" for="X">X</label>
      <select id="X" name="X" class = "form-control">
        <option value="-5">-5</option>
        <option value="-4">-4</option>
        <option value="-3">-3</option>
        <option value="-2">-2</option>
        <option value="-1">-1</option>
        <option selected value="0">0</option>
        <option value="1">1</option>
        <option value="2">2</option>
        <option value="3">3</option>
      </select>
    </div>
```

```
    <div class="col-xs-2 col-sm-2 col-lg-2">
      <div class = "form-group">
        <label id="labelY" for="Y">Y</label>
        <input id="Y" type="text" class="form-control" name="Y" placeholder="[-5, 3]"/>
      </div>
    </div>

    <div class = "col-xs-2 col-sm-2 col-lg-2">
      <label id="labelR" for="R">R</label>
      <select id="R" name="R" class="form-control" (change)="onRadiusChange()">
        <option selected value="1">1</option>
        <option value="2">2</option>
        <option value="3">3</option>
        <option value="4">4</option>
        <option value="5">5</option>
      </select>
    </div>
    <div class="col-xs-3 col-sm-3 col-lg-3"></div>
  </div>

  <label id="errorLabel" class="error-text"></label>
  <input id="btn-check" class="btn btn-info" type="submit" value="Отправить">
</form>

<form id="click_form" action="rest/point/check" method="post">
  <input id="click_X" type="hidden" name="X"/>
  <input id="click_Y" type="hidden" name="Y"/>
  <input id="click_R" type="hidden" name="R"/>
</form>

<canvas id="canvas" class="center" width="705" height="705"
(click)="onCanvasClick($event)"></canvas> <br>
```

```
<div *ngIf="points.length > 0">
  <table class = "table table-striped">
    <tr>
      <th>X</th>
      <th>Y</th>
      <th>R</th>
      <th>Попадание</th>
    </tr>
    <tr *ngFor="let point of points">
      <td>{{point.x | trunc}}</td>
      <td>{{point.y | trunc}}</td>
      <td>{{point.r}}</td>
      <td>{{point.hit | yesno}}</td>
    </tr>
  </table>
</div>


<form action="rest/user/logout" method="post">
  <input class="btn btn-danger" type="submit" value="Выход">
</form>
```

Вывод

В ходе выполнения данной работы мы научились работать с фреймворком Angular, создавать и использовать компоненты, организовывать его взаимодействие с бэкэндом и создавать адаптивные страницы.