Факультет программной инженерии и компьютерной техники

Веб-программирование

Лабораторная работа №4

Вариант 22

Преподаватель: Письмак Алексей Евгеньевич

Выполнил: Кульбако Артемий Юрьевич

Р3212

# Задание.

Переписать приложение из [предыдущей лабораторной работы](https://se.ifmo.ru/courses/web#lab3) с использованием следующих технологий:

* Уровень back-end должен быть основан на Spring.
* Уровень front-end должен быть построен на Vue.js 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 1211 пикселей.
* "Планшетный" - для устройств, ширина экрана которых равна или превышает 656, но меньше 1211 пикселей.
* "Мобильный"- для устройств, ширина экрана которых меньше 656 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: Checkbox {'-4','-3','-2','-1','0','1','2','3','4'} для координаты по оси X, Text (-5 ... 3) для координаты по оси Y, и Checkbox {'-4','-3','-2','-1','0','1','2','3','4'} 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.
* Ссылку, по которой аутентифицированный пользователь может закрыть свою сессию и вернуться на стартовую страницу приложения.

**Дополнительные требования к приложению:**

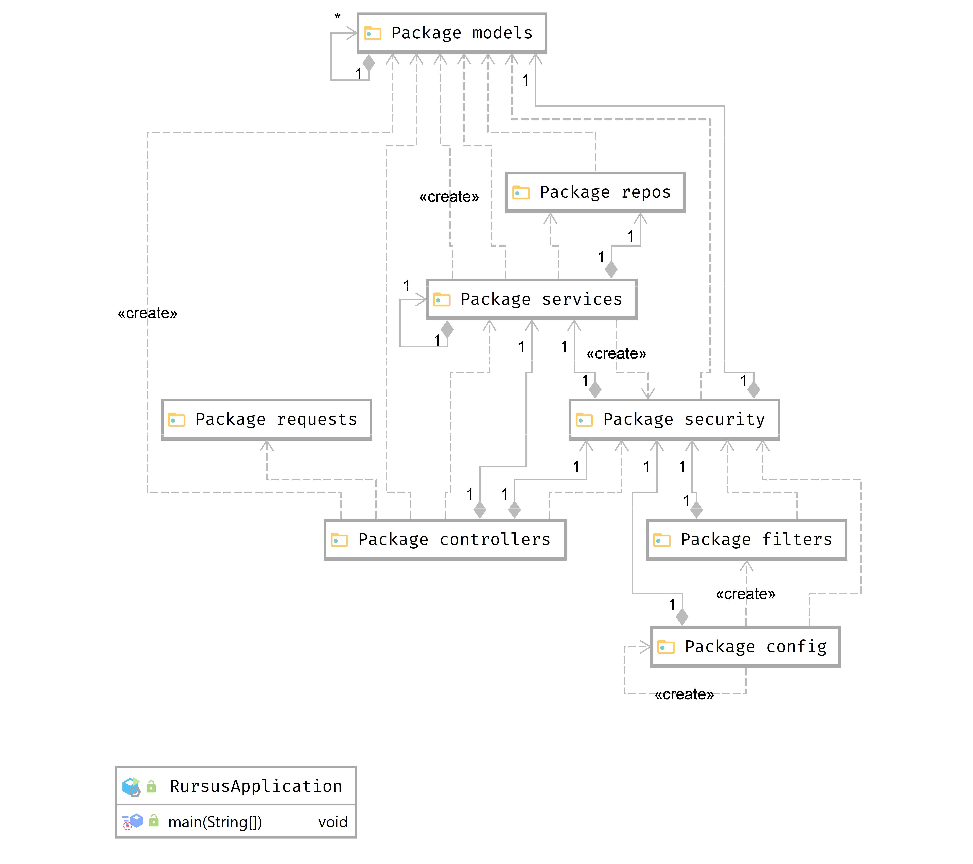
* Все результаты проверки должны сохраняться в базе данных под управлением СУБД PostgreSQL.
* Для доступа к БД необходимо использовать Spring Data.

# 

# Выводы по работе.

В процессе работы я научился создавать SPA на Vue компонентным подходом и RESTful-сервис на Spring MVC с использованием Spring Security и Spring Data, а также получил прикладные навыки: работа с системами сборки Maven и Webpack, менеджером пакетов npm. Узнал о принципах IoC и DI, механизме CORS.

# UML.



# Код.

## Accessor.vue

<template>  
 <div *id*="content">  
 <Notification *id*="title" *:message*="`(${time}) Вход в систему`" *:is-error*="*false*" *:is-hidden*="*false*"/>  
 <hr/>  
 <div *id*="controlContainer">  
 <div *id*="login">  
 <label *for*="loginInput">Введите почту:</label>  
 <input *id*="loginInput" *required class*="illuminated animated bordered rounded colored" *type*="text" *placeholder*="email" *v-model.trim*="email"/>  
 </div>  
 <div *id*="password">  
 <label *for*="passwordInput">Введите пароль:</label>  
 <input *id*="passwordInput" *required class*="illuminated animated bordered rounded colored" *type*="password" *placeholder*="secret\_word" *v-model.trim*="password"/>  
 </div>  
 <div>  
 <CheckButton *color*="blue" *label*="Зарегистрироваться" *@click.native*="register"/>  
 <CheckButton *color*="red" *label*="Войти" *@click.native*="login"/>  
 </div>  
 </div>  
 <hr/>  
 <Notification *v-bind*="notificationParams"/>  
 </div>  
</template>  
  
<script>  
 *import* CheckButton *from* "@/components/CheckButton";  
 *import* Notification *from* "@/components/Notification";  
  
 *export default* {  
 name: "Accessor",  
 components: {Notification, CheckButton},  
 data: *function* () {  
 *return* {  
 time: *null*,  
 email: "",  
 password: "",  
 notificationParams: {  
 isVisible: *false*,  
 isError: *true*,  
 message: undefined  
 }  
 }  
 },  
 methods: {  
 clock: *function* () {  
 *let* date = *new* Date(),  
 hours = (date.getHours() < 10) ? "0" + date.getHours() : date.getHours(),  
 minutes = (date.getMinutes() < 10) ? "0" + date.getMinutes() : date.getMinutes(),  
 seconds = (date.getSeconds() < 10) ? "0" + date.getSeconds() : date.getSeconds();  
 *this*.time = `${hours}:${minutes}:${seconds}`;  
 },  
 login: *function* () {  
 *if* (*this*.validateForm()) {  
 *this*.$axios.post("user", {  
 email: *this*.email,  
 password: *this*.password  
 }).then(response => {  
 *if* (response.status === 202) {  
 localStorage.setItem("jwt", response.data);  
 *this*.$router.push({path: "/app"});  
 }  
 }).catch(error => *this*.axiosErrorsHandler(error));  
 }  
 },  
 register: *function* () {  
 *if* (*this*.validateForm()) {  
 *this*.$axios.put("user", {  
 email: *this*.email,  
 password: *this*.password  
 }).then((response) => {  
 *if* (response.status === 201) {  
 *this*.notificationParams.message = response.data;  
 *this*.notificationParams.isError = *false*;  
 *this*.notificationParams.isVisible = *true*;  
 setTimeout(*this*.login, 1000);  
 }  
 }).catch(error => *this*.axiosErrorsHandler(error));  
 }  
 },  
 axiosErrorsHandler: *function* (error) {  
 *this*.notificationParams.message = error.response.data;  
 *this*.notificationParams.isVisible = *true*;  
 *this*.notificationParams.isError = *true*;  
 },  
 validateForm: *function* () {  
 *let* regExp = *new* RegExp(/\S+@\S+\.\S+/);  
 *if* (*this*.email.length === 0 || *this*.password.length === 0) {  
 *this*.notificationParams.isVisible = *true*;  
 *this*.notificationParams.isError = *true*;  
 *this*.notificationParams.message = "Данные не введены";  
 *return false*;  
 } *else if* (!regExp.test(String(*this*.email).toLowerCase())) {  
 *this*.notificationParams.isError = *true*;  
 *this*.notificationParams.isVisible = *true*;  
 *this*.notificationParams.message = "email введён некорректно";  
 *return false*;  
 } *else return true*;  
 }  
 },  
 created: *function* () {  
 setInterval(*this*.clock, 1000);  
 *this*.clock();  
 }  
 }  
</script>  
  
<style *scoped*>  
 #content {  
 margin-left: 10%;  
 margin-right: 10%;  
 }  
  
 #controlContainer \* {margin: 1.5%}  
  
 #controlContainer {  
 display: flex;  
 flex-direction: column;  
 }  
</style>

## CheckButton.vue

<template>  
 <button *:class*="color">{{ label }}</button>  
</template>  
  
<script>  
 *export default* {  
 name: "CheckButton",  
 props: {  
 label: ***String***,  
 color: ***String*** }  
 }  
</script>  
  
<style *scoped*>  
 */\* Взято и переработано с https://codepen.io/rauldronca/pen/mARYEm\*/* button {  
 text-align: center;  
 text-transform: uppercase;  
 cursor: pointer;  
 letter-spacing: 2px;  
 position: relative;  
 border: none;  
 color: white;  
 padding: 15px;  
 min-width: 150px;  
 transition-duration: 0.4s;  
 overflow: hidden;  
 box-shadow: 0 5px 15px #193047;  
 border-radius: 4px;  
 }  
  
 button:hover {  
 background: white;  
 color: black;  
 }  
  
 button::after {  
 content: "";  
 display: block;  
 position: absolute;  
 padding-top: 300%;  
 padding-left: 350%;  
 margin-left: -20px *!important*;  
 margin-top: -120%;  
 opacity: 0;  
 transition: all 0.8s  
 }  
  
 button:active::after {  
 padding: 0;  
 margin: 0;  
 opacity: 1;  
 transition: 0s  
 }  
  
 button:focus {outline: 0}  
  
 .red {background-color: #eb2a5a}  
  
 .red:hover {box-shadow: 0 2px 10px 5px #eb2a5a}  
  
 .red::after {background: #eb2a5a}  
  
 .yellow {background-color: #ead541}  
  
 .yellow:hover {box-shadow: 0 2px 10px 5px #ead541}  
  
 .yellow::after {background: #ead541}  
  
 .blue {background-color: #0a72b0}  
  
 .blue:hover {box-shadow: 0 2px 10px 5px #0a72b0}  
  
 .blue::after {background: #0a72b0}  
</style>

## Footer.vue

<template>  
 <div *id*="footer" *class*="shaded animated">  
 <figure>  
 <a *class*="illuminated" *v-bind:href*="pulpitUrl">  
 <img *v-bind:src*="pulpitLogoPath" *alt*="Лого сайта кафедры" *title*="Перейти на сайт кафедры" *class*="illuminated">  
 </a>  
 <figcaption>{{ devYear.getFullYear() }}</figcaption>  
 </figure>  
 </div>  
</template>  
  
<script>  
 *export default* {  
 name: "Footer",  
 props: {  
 pulpitLogoPath: ***String***,  
 pulpitUrl: ***String***,  
 devYear: ***Date*** }  
 }  
</script>  
  
<style *scoped*>  
 #footer {  
 bottom: 0;  
 z-index: 10; */\*чтобы footer всегда был сверху\*/* width: 100%;  
 position: fixed;  
 color: white;  
 background-color: #000720;  
 left: 0;  
 }  
  
 figure img {  
 width: 47px;  
 height: 55px;  
 }  
  
 figcaption {  
 font-size: 0.8em;  
 display: block;  
 }  
  
 figure {padding: 0.5%}  
</style>

## Header.vue

<template>  
 <div *id*="header" *class*="shaded animated">  
 <h1>{{ title }}</h1>  
 <table *id*="credit">  
 <tbody><tr>  
 <td><img *v-bind:src*="authorLogoPath" *alt*="Аватар разработчика"></td>  
 <td><a *class*="illuminated" *v-bind:href*="authorUrl" *title*="Перейти на сайт разработчика">{{ authorName }}</a></td>  
 </tr></tbody>  
 </table>  
 </div>  
</template>  
  
<script>  
 *export default* {  
 name: "Header",  
 props: {  
 title: ***String***,  
 authorName: ***String***,  
 authorUrl: ***String***,  
 authorLogoPath: ***String*** }  
 }  
</script>  
  
<style *scoped*>  
 #header {  
 font-size: 0.6em;  
 top: 0;  
 left: 0;  
 z-index: 10; */\*чтобы header всегда был сверху\*/* width: 100%;  
 position: fixed;  
 color: white;  
 background-color: #000720;  
 }  
  
 #credit {margin: auto}  
  
 #credit img {  
 width: 40px;  
 height: 40px;  
 }  
  
 #credit a {  
 font-size: 15px;  
 text-decoration: none;  
 color: white;  
 }  
</style>

## Error.vue

<template>  
 <div *id*="content">  
 <h1>{{ errorCode }}</h1>  
 <h2>{{ errorMessage }}</h2>  
 <router-link *to*="/login" *class*="illuminated">Вернуться на домашную страницу</router-link>  
 </div>  
</template>  
  
<script>  
 *export default* {  
 name: "NotFoundError",  
 props: {  
 errorCode: ***String***,  
 errorMessage: ***String*** }  
 }  
</script>  
  
<style *scoped*>  
 h1 {  
 font-size: 8em;  
 margin: 0;  
 background: linear-gradient(130deg, #2a48eb, #ff0018);  
 color: transparent;  
 background-clip: text;  
 text-transform: uppercase;  
 }  
  
 h2 {  
 font-size: 33px;  
 font-weight: 200;  
 text-transform: uppercase;  
 margin: 0;  
 letter-spacing: 3px;  
 }  
</style>

## Notification.vue

<template>  
 <h3><span *class*="notification stub" *v-if*="isVisible" *:class*="{ errorStub: isError }">{{ message }}</span></h3>  
</template>  
  
<script>  
 *export default* {  
 name: "Notification",  
 props: {  
 isError: ***Boolean***,  
 isVisible: ***Boolean***,  
 message: ***String*** }  
 }  
</script>  
  
<style *scoped*>  
 .notification {  
 color: white;  
 padding: 10px;  
 }  
  
 .stub {background-color: #000720}  
  
 .errorStub {background-color: #dc143b}  
</style>

## Validator.vue

<template>  
 <div *id*="content">  
 <Notification *message*="Валидация введёных значений" *:is-error*="*false*" *:is-hidden*="*false*"/>  
 <hr/>  
 <div *id*="wrapper">  
 <div *id*="controlContainer">  
 <table *id*="controlTable">  
 <tbody>  
 <tr>  
 <td *colspan*="4">  
 <label> Выберите X:  
 <select *required class*="animated illuminated bordered" *v-model*="point.x">  
 <option>-4</option>  
 <option>-3</option>  
 <option>-2</option>  
 <option>-1</option>  
 <option>0</option>  
 <option>1</option>  
 <option>2</option>  
 <option>3</option>  
 <option>4</option>  
 </select>  
 </label>  
 </td>  
 </tr>  
 <tr>  
 <td *colspan*="4">  
 <label>  
 Введите Y:  
 <input *id*="Y-input" *required class*="illuminated animated bordered rounded colored" *type*="text" *placeholder*="(-5 до 3)" *maxlength*="6" *v-model*="point.y">  
 </label>  
 </td>  
 </tr>  
 <tr><td *colspan*="4">Выберите R:</td></tr>  
 <tr>  
 <td><label>1<input *type*="radio" *class*="illuminated animated" *value*="1" *v-model*="point.r" *@change*="redrawGraph"></label></td>  
 <td><label>2<input *type*="radio" *class*="illuminated animated" *value*="2" *v-model*="point.r" *@change*="redrawGraph"></label></td>  
 <td><label>3<input *type*="radio" *class*="illuminated animated" *value*="3" *v-model*="point.r" *@change*="redrawGraph"></label></td>  
 <td><label>4<input *type*="radio" *class*="illuminated animated" *value*="4" *v-model*="point.r" *@change*="redrawGraph"></label></td>  
 </tr>  
 <tr><td *colspan*="3"><CheckButton *color*="red" *label*="Проверить" *@click.native*="validateFromButton"/></td></tr>  
 <tr>  
 <td><button *@click*="deletePoints" *title*="Удалить все точки из базы данных и с графика" *class*="animated illuminated system-button">Удалить все точки</button></td>  
 <td><button *@click*="logout" *title*="Завершить сеанс и вернуться на домашную страницу" *class*="animated illuminated system-button">Выйти</button></td>  
 </tr>  
 </tbody>  
 </table>  
 </div>  
 <div *id*="svgContainer">  
 <svg *id*="graph" *xmlns*="http://www.w3.org/2000/svg" *class*="bordered rounded" *@click*="validateFromGraph">  
 <line *x1*="0" *y1*="150" *x2*="300" *y2*="150" *stroke*="#000720"/>  
 <line *x1*="150" *y1*="0" *x2*="150" *y2*="300" *stroke*="#000720"/>  
 <polygon *points*="300,150 295,155 295, 145" *fill*="#000720" *stroke*="#000720"/>  
 <polygon *points*="150,0 145,5 155,5" *fill*="#000720" *stroke*="#000720"/>  
 <rect *id*="rect" *x*="30" *y*="90" *width*="120" *height*="60" *fill-opacity*="0.4" *stroke*="navy" *fill*="blue"/>  
 <polygon *id*="triangle" *points*="150,150 150,270 30,150" *fill-opacity*="0.4" *stroke*="navy" *fill*="blue"/>  
 <path *id*="path" *d*="M 150 150 L 270 150 C 270 210 210 270 150 270 Z" *fill-opacity*="0.4" *stroke*="navy" *fill*="blue"/>  
 </svg>  
 </div>  
 </div>  
 <hr/>  
 <div *id*="outputContainer">  
 <Notification *message*="Точки отсутствуют" *:is-error*="*false*" *:is-visible*="points.length === 0"/>  
 <Notification *v-bind*="errorTableNotification"/>  
 <table *v-if*="points.length !== 0" *id*="outputTable">  
 <thead>  
 <tr>  
 <th>x</th>  
 <th>y</th>  
 <th>r</th>  
 <th>Точка входит в ОДЗ</th>  
 <th>Дата рождения</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr *v-for*="***point*** *in* points" *:key*="***point***">  
 <td>{{ ***point***.x }}</td>  
 <td>{{ ***point***.y }}</td>  
 <td>{{ ***point***.r }}</td>  
 <td>{{ (***point***.status) ? "\u2611" : "\u2610" }}</td>  
 <td>{{ new ***Date***(***Date***.parse(***point***.bornDate)).toLocaleString() }}</td>  
 </tr>  
 </tbody>  
 </table>  
 </div>  
 </div>  
</template>  
  
<script>  
 *import* CheckButton *from* "@/components/CheckButton";  
 *import* Notification *from* "@/components/Notification";  
  
 *export default* {  
 name: "Validator",  
 components: {Notification, CheckButton},  
 data: *function* () {  
 *return* {  
 errorTableNotification: {  
 message: *undefined*,  
 isError: *true*,  
 isVisible: *false*,  
 },  
 point: {  
 x: *null*,  
 y: *null*,  
 r: 4 *//Изначально задано макс. значение для отрисовки графика в макс. масштабе* },  
 points: *new* ***Array***(0)  
 }  
 },  
 methods: {  
 validateFromButton: *function* () {  
 *if* ((*this*.point.x >= -4 && *this*.point.x <= 4) && *this*.point.y > -5 && *this*.point.y < 3) *this*.addPoint();  
 *else* {  
 *this*.errorTableNotification.message = "Данные введены некорректно";  
 *this*.errorTableNotification.isVisible = *true*;  
 }  
 },  
 validateFromGraph: *function* () {  
 *let* position = getMousePosition(***document***.getElementById("graph"), ***event***);  
 *this*.point.x = ((position.x - 150) / 125 \* *this*.point.r).toFixed(6);  
 *this*.point.y = ((150 - position.y) / 125 \* *this*.point.r).toFixed(6);  
 *this*.addPoint();  
  
 *function* getMousePosition(element, event) {  
 *let* rect = element.getBoundingClientRect();  
 *return* {  
 x: event.clientX - rect.left,  
 y: event.clientY - rect.top  
 };  
 }  
 },  
 addPoint: *function*() {  
 *this*.$axios.put("point", {  
 x: *this*.point.x,  
 y: *this*.point.y,  
 r: *this*.point.r  
 }, {  
 headers: {Authorization: "Bearer " + ***localStorage***.getItem("jwt")}  
 }).then(() => {  
 *this*.loadPoints();  
 }).catch(error => {  
 *let* answer = error.response.data.errors[0];  
 *this*.errorTableNotification.message = `${answer.field} ${answer.defaultMessage}`;  
 *this*.errorTableNotification.isVisible = *true*;  
 });  
 },  
 loadPoints: *function* () {  
 *this*.$axios.get("point", {  
 headers: {Authorization: "Bearer " + ***localStorage***.getItem("jwt")}  
 }).then(response => {  
 *this*.points = response.data;  
 *this*.redrawGraph();  
 }).catch(error => {  
 *this*.errorTableNotification.message = error.response.statusText;  
 *this*.errorTableNotification.isVisible = *true*;  
 });  
 },  
 redrawGraph: *function* () {  
 *let* r = *this*.point.r;  
 *const* svg = ***document***.getElementById("graph");  
 *const* rect = svg.getElementById("rect");  
 *const* triangle = svg.getElementById("triangle");  
 *const* path = svg.getElementById("path");  
 rect.setAttribute("width", `${r \* 30}`);  
 rect.setAttribute("height", `${r \* 15}`);  
 rect.setAttribute("x", `${150 - r \* 30}`);  
 rect.setAttribute("y", `${150 - r \* 15}`);  
 triangle.setAttribute("points", `150,150 150,${150 + 30 \* r} ${150 - 30 \* r},150`);  
 *let* k = (r !== 4) ? ((4 - r) \* 14) : 0;  
 path.setAttribute("d", `M 150 150 L ${150 + r \* 30} 150 C ${150 + r \* 30} ${210 - k} ${210 - k} ${150 + r \* 30} 150 ${150 + r \* 30} Z`);  
 *let* oldPoints = ***document***.querySelectorAll("circle");  
 oldPoints.forEach(oldPoint => oldPoint.parentNode.removeChild(oldPoint));  
 *if* (*this*.points.length !== 0) {  
 *this*.points.forEach(item => {  
 *let* newPoint = ***document***.createElementNS("http://www.w3.org/2000/svg", "circle");  
 newPoint.setAttribute("r", "5");  
 newPoint.setAttribute("cx", `${item.x / r \* 125 + 150}`);  
 newPoint.setAttribute("cy", `${150 - item.y / r \* 125}`);  
 newPoint.setAttribute("fill", item.status === *true* ? "green" : "red");  
 svg.appendChild(newPoint);  
 });  
 }  
 },  
 deletePoints: *function* () {  
 *this*.$axios.delete("point", {  
 headers: {Authorization: "Bearer " + ***localStorage***.getItem("jwt")}  
 }).then(() => {  
 *this*.loadPoints();  
 }).catch(error => {  
 *this*.errorTableNotification.message = error.response.statusText;  
 *this*.errorTableNotification.isVisible = *true*;  
 });  
 },  
 logout: *function* () {  
 *this*.$router.push({path: "/login"}, () => ***localStorage***.clear());  
 }  
 },  
 mounted: *function* () {  
 *this*.loadPoints();  
 }  
 }  
</script>  
  
<style *scoped*>  
 #content {  
 margin-left: 5%;  
 margin-right: 5%;  
 }  
  
 #wrapper {  
 display: flex;  
 flex-wrap: wrap;  
 justify-content: center;  
 }  
  
 #wrapper \* {  
 min-width: 50%;  
 }  
  
 #outputContainer {  
 margin-bottom: 130px;  
 }  
  
 input, svg {  
 background-color: white;  
 }  
  
 svg {  
 width: 300px;  
 height: 300px;  
 box-shadow: inset 0 0 7px 1px gray;  
 }  
  
 #svgContainer \* {margin: 2px}  
  
 .system-button {background-color: aliceblue}  
  
 #controlTable \* {margin: 4%}  
  
 #outputTable {  
 border: 1px solid #000720;  
 border-collapse: collapse;  
 margin: auto;  
 width: 90%;  
 }  
  
 #outputTable th {  
 background-color: #000720;  
 color: white;  
 }  
  
 #outputTable \* {  
 padding: 25px;  
 }  
</style>

## App.vue

<template>  
 <div *id*="app">  
 <Header *v-bind*="headerParams"/>  
 <div *id*="contentContainer" *class*="shaded">  
 <router-view/>  
 </div>  
 <Footer *v-bind*="footerParams"/>  
 </div>  
</template>  
  
<script>  
*import* Header *from* "@/components/Header";  
*import* Footer *from* "@/components/Footer";  
  
*export default* {  
 name: 'app',  
 components: {Header, Footer},  
 data: *function* () {  
 *return* {  
 headerParams: {  
 title: "Веб-программирование, Лаб. 4, Вариант 22",  
 authorName: "Кульбако Артемий Юрьевич, P3212",  
 authorUrl: "https://github.com/testpassword",  
 authorLogoPath: "/assets/img/avatar.png"  
 },  
 footerParams: {  
 pulpitLogoPath: "/assets/img/pulpit.png",  
 pulpitUrl: "https://se.ifmo.ru",  
 devYear: *new* ***Date***()  
 }  
 }  
 },  
 mounted: *function* () {  
 *const* trap = ***document***.createElement("script");  
 trap.setAttribute("src", "/assets/js/easter\_egg.js");  
 ***document***.body.appendChild(trap);  
 }  
}  
</script>  
  
<style>  
 *@font-face* {  
 font-family: "Avenir Next Cyr";  
 src: url("/assets/fonts/AvenirNextCyr-Regular.ttf");  
 }  
  
 \* {  
 font-family: "Avenir Next Cyr", Arial, sans-serif;  
 text-align: center;  
 font-weight: normal;  
 }  
  
 body {  
 background-color: whitesmoke;  
 }  
  
 #contentContainer {  
 margin-top: 90px;  
 position: absolute;  
 padding: 10px;  
 left: 50%;  
 width: 75%;  
 transform: translate(-50%, 0);  
 background-color: ghostwhite;  
 min-height: 1152px; */\*минимальное разрешение макета\*/* min-width: 648px;  
 }  
  
 a {  
 color: #eb2a5a;  
 font-weight: 200;  
 border-radius: 2px;  
 }  
  
 hr {  
 color: #f41c52;  
 }  
  
 .shaded {box-shadow: 0 0 10px 1px black} */\* постоянная тень\*/* .shaded:hover, .animated:hover {transition: 0.5s} */\*время эффекта\*/* .shaded:hover {box-shadow: 0 0 20px 2px black} */\*увеличение тени при наведении\*/* .animated:hover {transform: scale(1.05)} */\*увеличение элемента при наведении\*/* .illuminated:hover, .illuminated:focus { */\*подстветка элемента при наведении\*/* box-shadow: 0 0 40px 5px #f41c52;  
 outline: none; */\*убирает рамку фокуса в chrome\*/* }  
  
 .colored:focus { */\*меняет цвет элемента в фокусе\*/* background-color: #f41c52;  
 color: white;  
 }  
  
 .bordered {border: 2px solid #f41c52} */\*красные края\*/* .rounded { */\*скруглённые края\*/* border-radius: 20px;  
 background-color: white;  
 }  
  
 .rotated {  
 animation: rotation 5s infinite linear;  
 background: rgb(2,0,36);  
 background: radial-gradient(circle, rgba(2,0,36,1) 11%, rgba(115,2,9,1) 74%, rgba(138,3,3,1) 83%, rgba(138,3,3,1) 96%);  
 }  
  
 *@-webkit-keyframes* rotation {  
 from {  
 -webkit-transform: rotate(0deg);  
 }  
 to {  
 -webkit-transform: rotate(360deg);  
 }  
 }  
</style>

## main.js

*import* Vue *from* 'vue'  
*import* App *from* './App.vue'  
*import* router *from* "@/router";  
*import* ***Axios*** *from* "axios";  
  
Vue.config.productionTip = *false*;  
Vue.prototype.$axios = ***Axios***.create({baseURL: "/api"});  
  
*new* Vue({router, render: h => h(App)}).$mount('#app');

## router.js

*import* Vue *from* 'vue'  
*import* VueRouter *from* "vue-router";  
*import* Accessor *from* "@/components/Accessor";  
*import* Validator *from* "@/components/Validator";  
*import* NotFoundError *from* "@/components/Error";  
  
Vue.use(VueRouter);  
  
*export default new* VueRouter({  
 mode: "history",  
 routes: [  
 {  
 path: "/login",  
 name: "auth-page",  
 component: Accessor  
 },  
 {  
 path: "/app",  
 name: "app-page",  
 component: Validator,  
 beforeEnter: (to, from, next) => {  
 *if* (***localStorage***.getItem("jwt")) next();  
 *else* next({  
 name: "error-page",  
 params: {  
 errorCode: "401",  
 errorMessage: "Доступ неавторизованным пользователям запрещён"  
 }  
 });  
 }  
 },  
 {  
 path: "/",  
 name: "default-page",  
 beforeEnter: (to, from, next) => {  
 (***localStorage***.getItem("jwt") !== *null*) ? next({name: "app-page"}) : next({path: "auth-page"});  
 }  
 },  
 {  
 path: "/\*",  
 name: "error-page",  
 component: NotFoundError,  
 props: {  
 *default*: *true*,  
 errorCode: "404",  
 errorMessage: "Страница, которую вы запрашиваете, отсутствует"  
 }  
 }  
 ]  
});

## vue.config.js

*module*.exports = {  
 devServer: {  
 proxy: {  
 "/api": {  
 target: "http://localhost:8088",  
 changeOrigin: *true* }  
 }  
 }  
};

## JWTConfigurer

*package* logic.config;  
  
*import* logic.filters.JWTFilter;  
*import* logic.security.JWTUtil;  
*import* org.springframework.security.config.annotation.SecurityConfigurerAdapter;  
*import* org.springframework.security.config.annotation.web.builders.HttpSecurity;  
*import* org.springframework.security.web.DefaultSecurityFilterChain;  
*import* org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;  
  
*/\*\*  
 \* Конфигурирует Spring Security для работы с jwt.  
 \** ***@author*** *Артемий Кульбако  
 \** ***@version*** *1.0  
 \*/  
public class* JWTConfigurer *extends* SecurityConfigurerAdapter<DefaultSecurityFilterChain, HttpSecurity> {  
  
 *private* JWTUtil jwtUtil;  
  
 *public* JWTConfigurer(JWTUtil jwtUtil) {  
 *this*.jwtUtil = jwtUtil;  
 }  
  
 @Override  
 *public void* configure(HttpSecurity builder) *throws* Exception {  
 builder.addFilterBefore(*new* JWTFilter(jwtUtil), UsernamePasswordAuthenticationFilter.*class*);  
 }  
}

## SecurityConfigurer

*package* logic.config;  
  
*import* logic.security.JWTUtil;  
*import* org.springframework.beans.factory.annotation.Autowired;  
*import* org.springframework.context.annotation.Bean;  
*import* org.springframework.context.annotation.Configuration;  
*import* org.springframework.security.authentication.*AuthenticationManager*;  
*import* org.springframework.security.config.annotation.web.builders.HttpSecurity;  
*import* org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
*import* org.springframework.security.config.http.SessionCreationPolicy;  
  
*/\*\*  
 \* Отвечает за доступ к различным REST-контроллерам.  
 \** ***@author*** *Артемий Кульбако  
 \** ***@version*** *1.0  
 \*/*@Configuration  
*public class* SecurityConfigurer *extends* WebSecurityConfigurerAdapter {  
  
 @Autowired *private* JWTUtil jwtUtil;  
  
 @Bean @Override  
 *public AuthenticationManager* authenticationManagerBean() *throws* Exception {  
 *return super*.authenticationManagerBean();  
 }  
  
 @Override  
 *protected void* configure(HttpSecurity http) *throws* Exception {  
 http.httpBasic().disable().csrf().disable().sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)  
 .and().authorizeRequests()  
 .antMatchers("/api/user").permitAll()  
 .anyRequest().authenticated()  
 .and().apply(*new* JWTConfigurer(jwtUtil));  
 }  
}

## PointController

*package* logic.controllers;  
  
*import* logic.models.\*;  
*import* logic.requests.PointDTO;  
*import* logic.security.JWTUtil;  
*import* logic.services.UserService;  
*import* org.springframework.beans.factory.annotation.Autowired;  
*import* org.springframework.http.\*;  
*import* org.springframework.web.bind.annotation.\*;  
*import* javax.servlet.http.*HttpServletRequest*;  
*import* javax.validation.Valid;  
*import* java.util.*List*;  
  
*/\*\*  
 \* Обрабатывает запросы к url-у /api/point.  
 \** ***@author*** *Артемий Кульбако  
 \** ***@version*** *1.2  
 \*/*@RestController @RequestMapping(path = "/api/point")  
*public class* PointController {  
  
 *//TODO валидация  
 private final* UserService userService;  
 *private final* JWTUtil jwtUtil;  
  
 @Autowired *public* PointController(JWTUtil jwtUtil, UserService userService) {  
 *this*.jwtUtil = jwtUtil;  
 *this*.userService = userService;  
 }  
  
 */\* В приведённых ниже методах для идентификации пользователя используется jwt. Он имеет вид 'Bearer hash'. Для корректной работы с ним, необходимо после получения его из заголовка, отрезать часть 'Bearer '. Можно делать это вручную, получив RequestHeader("Authorization") String token в сигнатуре метода, и обрезав часть в теле метода, либо передать чистый запрос экземпляру класса JWTUtil, обрабатывающему jwt. Второй вариант будет более правильным, так как в теории, мы можем не знать какую часть токена надо обрезать. \*/* @PutMapping  
 *private* ResponseEntity<String> addPoint(@Valid @RequestBody PointDTO data, *HttpServletRequest* req) {  
 User modifiedUser = userService.findByEmail(jwtUtil.getUsername(jwtUtil.resolveToken(req)));  
 modifiedUser.getPoints().add(*new* Point(data.getX(), data.getY(), data.getR()));  
 userService.updateUser(modifiedUser);  
 *return new* ResponseEntity<>(HttpStatus.CREATED);  
 }  
  
 @GetMapping  
 *private* ResponseEntity<*List*<Point>> loadPoints(*HttpServletRequest* req) {  
 *return new* ResponseEntity<>(userService.findByEmail(jwtUtil.getUsername(jwtUtil.resolveToken(req))).getPoints(), HttpStatus.OK);  
 }  
  
 @DeleteMapping  
 *private* ResponseEntity<String> clearPoints(*HttpServletRequest* req) {  
 User modifiedUser = userService.findByEmail(jwtUtil.getUsername(jwtUtil.resolveToken(req)));  
 modifiedUser.getPoints().clear();  
 userService.updateUser(modifiedUser);  
 *return new* ResponseEntity<>("Все ваши точки удалены", HttpStatus.OK);  
 }  
}

## UserController

*package* logic.controllers;  
  
*import* logic.models.User;  
*import* logic.requests.UserDTO;  
*import* logic.security.JWTUtil;  
*import* logic.services.UserService;  
*import* org.springframework.beans.factory.annotation.Autowired;  
*import* org.springframework.http.\*;  
*import* org.springframework.security.authentication.\*;  
*import* org.springframework.web.bind.annotation.\*;  
*import* javax.validation.Valid;  
*import* java.util.\*;  
  
*/\*\*  
 \* Обрабатывает запросы к url-у /api/user.  
 \** ***@author*** *Артемий Кульбако  
 \** ***@version*** *1.2  
 \*/*@RestController @RequestMapping(path = "/api/user")  
*public class* UserController {  
  
 *private final AuthenticationManager* authManager;  
 *private final* JWTUtil jwtUtil;  
 *private final* UserService userService;  
  
 @Autowired *public* UserController(*AuthenticationManager* authManager, JWTUtil jwtUtil, UserService userService) {  
 *this*.authManager = authManager;  
 *this*.userService = userService;  
 *this*.jwtUtil = jwtUtil;  
 }  
  
 @PostMapping  
 *private* ResponseEntity<String> login(@Valid @RequestBody UserDTO req) {  
 String email = req.getEmail();  
 String password = req.getPassword();  
 User user = userService.findByEmail(email);  
 *if* (user != *null*) {  
 authManager.authenticate(*new* UsernamePasswordAuthenticationToken(email, password));  
 String token = jwtUtil.generateToken(email, *new* ArrayList<String>(){{add("USER");}});  
 *return new* ResponseEntity<>(token, HttpStatus.ACCEPTED);  
 } *else return new* ResponseEntity<>("Сочетания почты и пароля не существует", HttpStatus.NOT\_FOUND);  
 }  
  
 @PutMapping  
 *private* ResponseEntity<String> register(@Valid @RequestBody UserDTO req) {  
 String email = req.getEmail();  
 String password = req.getPassword();  
 User user = userService.findByEmail(email);  
 *if* (user != *null*) *return new* ResponseEntity<>("Пользователь уже существует", HttpStatus.BAD\_REQUEST);  
 *else* {  
 userService.register(email, password);  
 *return new* ResponseEntity<>("Успешная регистрация", HttpStatus.CREATED);  
 }  
 }  
  
 @DeleteMapping  
 *private* ResponseEntity<String> remove(@Valid @RequestBody UserDTO req) {  
 String email = req.getEmail();  
 String password = req.getPassword();  
 User user = userService.findByEmail(email);  
 *if* (user != *null*) {  
 authManager.authenticate(*new* UsernamePasswordAuthenticationToken(email, password));  
 userService.remove(user);  
 *return new* ResponseEntity<>("Вы удалили свой аккаунт", HttpStatus.OK);  
 } *else return new* ResponseEntity<>("Указанного сочетания почты и пароля не существует", HttpStatus.NOT\_FOUND);  
 }  
}

## CORSFilter

*package* logic.filters;  
  
*import* org.springframework.stereotype.Component;  
  
*import* javax.servlet.\*;  
*import* javax.servlet.http.*HttpServletResponse*;  
*import* java.io.IOException;  
  
*/\*\*  
 \* Открывает доступ к контроллерам для запросов с любого url-а.  
 \** ***@author*** *Артемий Кульбако  
 \** ***@version*** *1.0  
 \*/*@Component  
*public class* CORSFilter *implements Filter* {  
  
 @Override  
 *public void* doFilter(*ServletRequest* servletRequest, *ServletResponse* servletResponse, *FilterChain* filterChain) *throws* IOException, ServletException {  
 *HttpServletResponse* response = (*HttpServletResponse*) servletResponse;  
 response.setHeader("Access-Control-Allow-Origin", "\*");  
 response.setHeader("Access-Control-Allow-Methods", "GET, POST, DELETE, PUT");  
 response.setHeader("Access-Control-Allow-Headers", "x-requested-with");  
 filterChain.doFilter(servletRequest, servletResponse);  
 }  
}

## JWTFilter

*package* logic.filters;  
  
*import* logic.security.JWTUtil;  
*import* org.springframework.security.core.*Authentication*;  
*import* org.springframework.security.core.context.SecurityContextHolder;  
*import* org.springframework.web.filter.GenericFilterBean;  
*import* javax.servlet.\*;  
*import* javax.servlet.http.*HttpServletRequest*;  
*import* java.io.IOException;  
  
*/\*\*  
 \* Фильтрует запросы на наличие jwt.  
 \** ***@author*** *Артемий Кульбако  
 \** ***@version*** *1.0  
 \*/  
public class* JWTFilter *extends* GenericFilterBean {  
  
 *private* JWTUtil jwtUtil;  
  
 *public* JWTFilter(JWTUtil jwtUtil) {  
 *this*.jwtUtil = jwtUtil;  
 }  
  
 @Override  
 *public void* doFilter(*ServletRequest* request, *ServletResponse* response, *FilterChain* chain) *throws* IOException, ServletException {  
 String token = jwtUtil.resolveToken((*HttpServletRequest*) request);  
 *if* (token != *null* && jwtUtil.validateToken(token)) {  
 *Authentication* auth = jwtUtil.getAuthentication(token);  
 *if* (auth != *null*) SecurityContextHolder.*getContext*().setAuthentication(auth);  
 }  
 chain.doFilter(request, response);  
 }  
}

## Point

*package* logic.models;  
  
*import* lombok.Data;  
*import* javax.persistence.\*;  
*import* java.io.*Serializable*;  
*import* java.util.Date;  
  
*/\*\*  
 \* Представляет сущность точки для бд.  
 \** ***@author*** *Артемий Кульбако  
 \** ***@version*** *1.2  
 \*/*@Data @Embeddable @Table(name = "points")  
*public class* Point *implements Serializable* {  
  
 @Transient *private static final long* serialVersionUID = 4L;  
 @Column(nullable = *false*) *private double* x, y, r;  
 @Column(nullable = *false*) *private boolean* status;  
 *private* Date bornDate;  
  
 *public* Point(*double* x, *double* y, *double* r) {  
 *this*.x = x;  
 *this*.y = y;  
 *this*.r = r;  
 checkCoordinates();  
 bornDate = *new* Date();  
 }  
  
 *public* Point() {}  
  
 *private void* checkCoordinates() {  
 *this*.status = (x <= 0 && y >= 0 && x >= -r && y <= r/2) || ((y >= -x - r) && x <= 0 && y <= 0) ||  
 (x >= 0 && y <= 0 && x \* x + y \* y <= Math.*pow*(r, 2));  
 }  
  
 @Deprecated  
 *public* String toHtmlMarkup() {  
 *return* "<tr><td>" + x + "</td>" +  
 "<td>" + y + "</td>" +  
 "<td>" + r + "</td>" +  
 "<td style='color: " + ((status) ? "green" : "red") + "'>" + status + "</td>" +  
 "<td>" + bornDate + "</td></tr>";  
 }  
}

## User

*package* logic.models;  
  
*import* lombok.Data;  
*import* javax.persistence.\*;  
*import* java.io.*Serializable*;  
*import* java.util.\*;  
  
*/\*\*  
 \* Представляет сущность пользователя для бд.  
 \** ***@author*** *Артемий Кульбако  
 \** ***@version*** *1.1  
 \*/*@Data @Entity @Table(name = "users")  
*public class* User *implements Serializable* {  
  
 @Transient *private static final long* serialVersionUID = 4L;  
 @Id *private* String email;  
 @Column(nullable = *false*) *private* String password;  
 @ElementCollection(fetch = FetchType.EAGER) @CollectionTable(name = "points") *private List*<Point> points;  
  
 *public* User() {}  
  
 *public* User(String email, String password) {  
 *this*.email = email;  
 *this*.password = password;  
 *this*.points = *new* ArrayList<>();  
 }  
}

## UserRepository

*package* logic.repos;  
  
*import* logic.models.User;  
*import* org.springframework.data.repository.*CrudRepository*;  
*import* org.springframework.stereotype.Repository;  
  
*/\*\*  
 \* CRUD-репозиторий для {****@code*** *User}.  
 \** ***@see*** *User  
 \** ***@author*** *Артемий Кульбако  
 \** ***@version*** *1.0  
 \*/*@Repository *public interface UserRepository extends CrudRepository*<User, String> {  
  
 *boolean* existsByEmail(String email);  
  
 *long* removeByEmailAndPassword(String email, String password);  
  
 User getByEmail(String email);  
}

## PointDTO

*package* logic.requests;  
  
*import* lombok.Data;  
*import* javax.validation.constraints.\*;  
*import* java.io.*Serializable*;  
  
*/\*\*  
 \* Объект обращения к серверу для {****@code*** *PointController}.  
 \** ***@see*** *logic.controllers.PointController  
 \** ***@author*** *Кульбако Артемий  
 \** ***@version*** *1.2  
 \*/*@Data  
*public class* PointDTO *implements Serializable* {  
 *private static final long* serialVersionUID = 4L;  
 @NotNull @Min(value = -4) @Max(value = 4) *private double* x;  
 @NotNull @Min(value = -5) @Max(value = 3) *private double* y;  
 @NotNull @Min(value = 1) @Max(value = 4) *private double* r;  
}

## UserDTO

*package* logic.requests;  
  
*import* lombok.Data;  
*import* javax.validation.constraints.\*;  
*import* java.io.*Serializable*;  
  
*/\*\*  
 \* Объект обращения к серверу для {****@code*** *UserController}.  
 \** ***@see*** *logic.controllers.UserController  
 \** ***@author*** *Кульбако Артемий  
 \** ***@version*** *1.0  
 \*/*@Data  
*public class* UserDTO *implements Serializable* {  
 *private static final long* serialVersionUID = 4L;  
 @Email @NotNull *private* String email;  
 @NotNull *private* String password;  
}

## JWTUtil

*package* logic.security;  
  
*import* io.jsonwebtoken.\*;  
*import* io.jsonwebtoken.SignatureAlgorithm;  
*import* logic.services.UserDetailsServiceImpl;  
*import* org.springframework.beans.factory.annotation.Autowired;  
*import* org.springframework.context.annotation.Bean;  
*import* org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
*import* org.springframework.security.core.*Authentication*;  
*import* org.springframework.security.core.userdetails.\*;  
*import* org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
*import* org.springframework.stereotype.Component;  
*import* javax.servlet.http.*HttpServletRequest*;  
*import* java.util.\*;  
*import* java.util.function.*Function*;  
  
*/\*\*  
 \* Класс, обсулживающий JsonWebToken-ы.  
 \** ***@author*** *Артемий Кульбако  
 \** ***@version*** *1.0  
 \*/*@Component  
*public class* JWTUtil {  
  
 *private final* String KEY = "liquid";  
 *private static final long* TOKEN\_VALIDITY = 604800000; *//1 неделя* @Autowired UserDetailsServiceImpl userDetails;  
  
 @Bean *public* BCryptPasswordEncoder passwordEncoder() {  
 *return new* BCryptPasswordEncoder();  
 }  
  
 *public* String generateToken(String username, *List*<String> roles) {  
 *Claims* claims = Jwts.*claims*().setSubject(username);  
 claims.put("roles", roles);  
 Date now = *new* Date();  
 *return* Jwts.*builder*().setClaims(claims).setIssuedAt(now).setExpiration(*new* Date(now.getTime() + TOKEN\_VALIDITY))  
 .signWith(SignatureAlgorithm.HS512, KEY).compact();  
 }  
  
 *public boolean* validateToken(String token) {  
 *Jws*<*Claims*> claims = Jwts.*parser*().setSigningKey(KEY).parseClaimsJws(token);  
 *if* (claims.getBody().getExpiration().before(*new* Date())) {  
 System.out.println("Токен невалиден");  
 *return false*;  
 } *else return true*;  
 }  
  
 *private boolean* isTokenExpired(String token) {  
 *final* Date expiration = getExpirationDate(token);  
 *return* expiration.before(*new* Date());  
 }  
  
 *public* String resolveToken(*HttpServletRequest* req) {  
 String bearerToken = req.getHeader("Authorization");  
 *if* (bearerToken != *null* && bearerToken.startsWith("Bearer ")) {  
 *return* bearerToken.substring(7);  
 } *else return null*;  
 }  
  
 *public Authentication* getAuthentication(String token) {  
 *UserDetails* ud = *this*.userDetails.loadUserByUsername(getUsername(token));  
 *return new* UsernamePasswordAuthenticationToken(ud, "", ud.getAuthorities());  
 }  
  
 *public* String getUsername(String token) {  
 *return* getClaim(token, *Claims*::getSubject);  
 }  
  
 *public* Date getExpirationDate(String token) {  
 *return* getClaim(token, *Claims*::getExpiration);  
 }  
  
 *public* <T> T getClaim(String token, *Function*<*Claims*, T> claimsResolver) {  
 *final Claims* claims = Jwts.*parser*().setSigningKey(KEY).parseClaimsJws(token).getBody();  
 *return* claimsResolver.apply(claims);  
 }  
}

## UserPrincipal

*package* logic.security;  
  
*import* logic.models.User;  
*import* org.springframework.security.core.*GrantedAuthority*;  
*import* org.springframework.security.core.authority.SimpleGrantedAuthority;  
*import* org.springframework.security.core.userdetails.*UserDetails*;  
*import* java.util.\*;  
  
*/\*\*  
 \* Предоставляет необходимую информацию для построения объекта {****@code*** *Authentication} из источников данных системы  
 \* безопасности с ролями.  
 \** ***@see*** *GrantedAuthority  
 \** ***@author*** *Артемий Кульбако  
 \** ***@version*** *1.0  
 \*/  
public class* UserPrincipal *implements UserDetails* {  
  
 *private* User user;  
  
 *public* UserPrincipal(User user) {  
 *this*.user = user;  
 }  
  
 @Override  
 *public Collection*<? *extends GrantedAuthority*> getAuthorities() {  
 *return* Collections.*singletonList*(*new* SimpleGrantedAuthority("User"));  
 }  
  
 @Override  
 *public* String getPassword() {  
 *return* user.getPassword();  
 }  
  
 @Override  
 *public* String getUsername() {  
 *return* user.getEmail();  
 }  
  
 @Override  
 *public boolean* isAccountNonExpired() {  
 *return true*;  
 }  
  
 @Override  
 *public boolean* isAccountNonLocked() {  
 *return true*;  
 }  
  
 @Override  
 *public boolean* isCredentialsNonExpired() {  
 *return true*;  
 }  
  
 @Override  
 *public boolean* isEnabled() {  
 *return true*;  
 }  
}

## ApplicationConfigurer

*package* logic.config;  
  
*import* org.slf4j.\*;  
*import* org.springframework.context.annotation.Bean;  
*import* org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
*import* org.springframework.stereotype.Component;  
  
*/\*\*  
 \* Класс, представляющий служебные бины на глобальном уровне.  
 \** ***@author*** *Артемий Кульбако  
 \** ***@version*** *1.0  
 \*/*@Component  
*public class* ApplicationConfigurer {  
  
 @Bean  
 *public* BCryptPasswordEncoder passwordEncoder() {  
 *return new* BCryptPasswordEncoder();  
 }  
  
 @Bean  
 *public Logger* logger() {  
 *return* LoggerFactory.*getLogger*("application");  
 }  
}

## RursusApplication

*package* logic;  
  
*import* org.springframework.boot.SpringApplication;  
*import* org.springframework.boot.autoconfigure.SpringBootApplication;  
  
*/\*\*  
 \* Точка входа в приложение на базе Spring Framework.  
 \** ***@author*** *Артемий Кульбако  
 \** ***@version*** *1.0  
 \*/*@SpringBootApplication  
*public class* RursusApplication {  
 *public static void* main(String[] args) {  
 SpringApplication.*run*(RursusApplication.*class*, args);  
 }  
}

## application.properties

*#Database  
spring.jpa.generate-ddl*=*true  
spring.datasource.url*=jdbc:postgresql://localhost:5432/web4  
*spring.datasource.username*=postgres  
*spring.datasource.password*=root  
  
*#Server  
server.port*=8088  
  
*#Email  
spring.mail.host*=smtp.mail.ru  
*spring.mail.port*=465  
*spring.mail.username*=nope  
*spring.mail.password*=nope  
*spring.mail.properties.mail.smtp.auth*=true  
*spring.mail.properties.mail.smtp.connectiontimeout*=1000  
*spring.mail.properties.mail.smtp.timeout*=1000  
*spring.mail.properties.mail.smtp.writetimeout*=1000  
*spring.mail.properties.mail.smtp.starttls.enable*=true