**ДОДАТОК А**

Лістинг Інформаційної системи для аналізу валютних коливань

Файл index.html

1. <!DOCTYPE html**>**
2. **<html** lang="en"**>**
3. **<head>**
4. **<link** rel="shortcut icon" href="/images/fav\_icon.ico" type="image/x-icon" **/>**
5. **<meta** charset="utf-8"**>**
6. **<meta** http-equiv="X-UA-Compatible" content="IE=edge"**>**
7. **<meta** name="viewport" content="width=device-width, initial-scale=1"**>**
8. <!-- The above 3 meta tags \*must\* come first in the head; any other head content must come \*after\* these tags -->
9. **<title>**Exchange Rates**</title>**
11. <!-- Loader -->
12. **<style>**
13. .wrapp {
14. background: #29ABE2;
15. position: absolute;
16. top: 0;
17. left: 0;
18. height: 100vh;
19. width: 100%;
20. }
21. .spinner {
22. width: 100px;
23. height: 100px;
24. background-color: #fff;
25. margin: 40vh auto;
26. -webkit-animation: sk-rotateplane 1.2s infinite ease-in-out;
27. animation: sk-rotateplane 1.2s infinite ease-in-out;
28. }
29. @-webkit-keyframes sk-rotateplane {
30. 0% { -webkit-transform: perspective(120px) }
31. 50% { -webkit-transform: perspective(120px) rotateY(180deg) }
32. 100% { -webkit-transform: perspective(120px) rotateY(180deg)  rotateX(180deg) }
33. }
35. @keyframes sk-rotateplane {
36. 0% {
37. transform: perspective(120px) rotateX(0deg) rotateY(0deg);
38. -webkit-transform: perspective(120px) rotateX(0deg) rotateY(0deg)
39. } 50% {
40. transform: perspective(120px) rotateX(-180.1deg) rotateY(0deg);
41. -webkit-transform: perspective(120px) rotateX(-180.1deg) rotateY(0deg)
42. } 100% {
43. transform: perspective(120px) rotateX(-180deg) rotateY(-179.9deg);
44. -webkit-transform: perspective(120px) rotateX(-180deg) rotateY(-179.9deg);
45. }
46. }
47. **</style>**
48. **<div** class="wrapp"**>**
49. **<div** class="spinner"**></div>**
50. **</div>**
51. **<link** rel='stylesheet prefetch' href='https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css'**>**
52. **<link** rel='stylesheet prefetch' href='https://cdnjs.cloudflare.com/ajax/libs/animate.css/3.2.3/animate.min.css'**>**
53. **<link** rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-awesome.min.css"**>**
55. <!-- Bootstrap -->
56. **<link** href="css/bootstrap.min.css" rel="stylesheet"**>**
58. **<link** href="css/style.css" rel="stylesheet"**>**
60. **<script** src="//jashkenas.github.io/underscore/underscore-min.js"**></script>**
62. <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->
63. <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
64. <!--[if lt IE 9]**>**
65. **<script** src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"**></script>**
66. **<script** src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"**></script>**
67. <![endif]--**>**
68. **</head>**
69. **<body>**
70. **<div** class="globalWrapper hidden"**>**
71. <!-- MENU -->
72. **<div** class="visible-lg visible-md hidden-sm hidden-xs menu"**>**
73. **<div** class="menuItem itemOne itemOneCustom"**>**
74. **<span** class="glyphicon glyphicon-home" aria-hidden = "true"**></span> <span** class="spanOne"**>**Home**</span>**
75. **</div>**
76. **<div** class="menuItem itemTwo"**>**
77. **<span** class="glyphicon glyphicon-cog" aria-hidden="true"**></span> <span** class="span"**>**Statistics**</span>**
78. **</div>**
79. **<div** class="menuItem itemThree"**>**
80. **<span** class="glyphicon glyphicon-signal" aria-hidden="true"**></span> <span** class="span"**>**Charts**</span>**
81. **</div>**
82. **<div** class="menuItem itemFour"**>**
83. **<span** class="glyphicon glyphicon-font" aria-hidden="true"**></span> <span** class="span"**>**About**</span>**
84. **</div>**
85. **</div>**
87. <!-- Navigation -->
88. **<nav** class="visible-sm visible-xs hidden-lg hidden-md navbar navbar-inverse navbar-fixed-top" role="navigation"**>**
89. **<div** class="container"**>**
90. <!-- Brand and toggle get grouped for better mobile display -->
91. **<div** class="navbar-header"**>**
92. **<button** type="button" class="navbar-toggle" data-toggle="collapse"
93. data-target="#bs-example-navbar-collapse-1"**>**
94. **<span** class="sr-only"**>**Toggle navigation**</span>** **<span**
95. class="icon-bar"**></span>** **<span** class="icon-bar"**></span>**
96. **<span** class="icon-bar"**></span>**
97. **</button>**
98. **<a** class="navbar-brand itemOne smallItems" href="#"**>**Home**</a>**
99. **</div>**
100. **<div** class="collapse navbar-collapse"
101. id="bs-example-navbar-collapse-1"**>**
102. **<ul** class="nav navbar-nav"**>**
103. **<li><a** href="#" class="itemTwo smallItems"**>**Statistics**</a></li>**
104. **<li><a** href="#" class="itemThree smallItems"**>**Charts**</a></li>**
105. **<li><a** href="#" class="itemFour smallItems"**>**About**</a></li>**
106. **</ul>**
107. **</div>**
108. **</div>**
109. **</nav>**
110. **<div** class="visible-sm visible-xs mobilePadding"**></div>**
111. **<div** class="container-fluid" id="mainContainer"**>**
112. **<div** class="row"**>**
113. **<div** class="col-lg-4"**>**
114. **<div** class="pad"**>**
115. **<div** class="col-lg-12" id="bloks"**>**
116. **<div** class="panel"**>**
117. **<div** class="panel-heading"**>**
118. **<h3** class="panel-title"**>**Single Date**</h3>**
119. **</div>**
120. **<div** class="panel-body"**>**
121. **<form** name="oneDateForm" id="oneDate"**>**
122. **<div** class="form-group"**>**
123. **<label** for="specificDate"**>**Choose Date**</label>**
124. **<input** type="date" class="form-control" name="specificDate"**>**
125. **</div>**
126. **</form>**
127. **<button** type="button" class="btn btn-block nbuSingle" id="buttonsFormat" data-loading-text= "Loading..."**>** National Bank Of Ukraine**</button>**
128. **<button** type="button" class="btn btn-block pbSingle" id="buttonsFormat" data-loading-text="Loading..."**>** Private Bank**</button>**
129. **</div>**
130. **</div>**
131. **</div>**
132. **<div** class="col-lg-12" id="bloks"**>**
133. **<div** class="panel"**>**
134. **<div** class="panel-heading"**>**
135. **<h3** class="panel-title"**>**Diapason Dates **</h3>**
136. **</div>**
137. **<div** class="panel-body"**>**
138. **<form** name="diapasonDatesForm" id="twoDates"**>**
139. **<div** class="form-group"**>**
140. **<label** for="startDate"**>**Start Date **</label>**
141. **<input** type="date" class="form-control" id="buttonsFormat" name="startDate"**>**
142. **</div>**
143. **<div** class="form-group"**>**
144. **<label** for="endDate"**>**End Date**</label>**
145. **<input** type="date" class="form-control" id="buttonsFormat" name="endDate"**>**
146. **</div>**
147. **</form>**
148. **<button** type="button" class="btn btn-block nbuDiapason" id="buttonsFormat" data-loading-text="Loading..."**>**National Bank Of Ukraine**</button>**
149. **<button** type="button" class="btn btn-block pbDiapason" id="buttonsFormat" data-loading-text="Loading..."**>**Private Bank**</button>**
150. **</div>**
151. **</div>**
152. **</div>**
153. **</div>**
154. **</div>**
155. **<div** class="col-lg-8 pad-table" id="tableContainer"**>**
156. **<div** class="col-lg-12" id="table1"**></div>**
157. **<div** class="col-lg-12" id="table2"**></div>**
158. **<script** id="templateRootTable" type="text/template"**>**
159. **<div** class="panel panel-default" id="mainPanel"**>**
160. **<div** class="panel-heading" id="panelHead"**>**
161. **<**%var a;
162. if (data[0].bank === "NBU") {
163. a = 'National Bank Of Ukraine(NBU)';
164. }else if(data[0].bank === "PrivatBank") {
165. a = 'PrivatBank(PB)';
166. }
167. %**>**
168. **<**%= a %**>**
169. **</div>**
170. **<table** class="table table-bordered table-hover" style="width: 100%;"**>**
171. **<thead** class="thead-inverse"**>**
172. **<tr>**
173. **<th>**Currency**</th>**
174. **<th>**Date**</th>**
175. **<th>**Buy**</th>**
176. **<th>**Sale**</th>**
177. **</tr>**
178. **</thead>**
179. **<tbody>**
180. **<**% data.forEach(function(item) { %**>**
181. **<tr>**
182. **<td><**%= item.rate %**></td>**
183. **<td><**%= item.date %**></td>**
184. **<td><**%= item.buy %**></td>**
185. **<td><**%= item.sale %**></td>**
186. **</tr>**
187. **<**% }); %**>**
188. **</tbody>**
189. **</table>**
190. **</div>**
191. **</script>**
192. **</div>**
193. **</div>**
194. **</div>**
196. **<div** class="container-fluid" id="chartContainer"**>**
197. **<div** class="row"**>**
198. **<div** class="col-lg-4"**>**
199. **<div** class="pad"**>**
200. **<div** class="col-lg-12" id="bloks"**>**
201. **<div** class="panel"**>**
202. **<div** class="panel-heading"**>**
203. **<h3** class="panel-title"**>**Parameters for building Chart**</h3>**
204. **</div>**
205. **<div** class="panel-body"**>**
206. **<div** class="rateOneBank"**>**
207. **<form** name="byRateOneBankForm"**>**
208. **<div** class="form-group"**>**
209. **<label** for="chooseRate"**>**Choose  Rate**</label>**
210. **<select** class="form-control" name="chooseRate" id="rate"**>**
211. **<option>**EUR**</option>**
212. **<option>**USD**</option>**
213. **<option>**RUB**</option>**
214. **</select>**
215. **</div>**
216. **<div** class="form-group"**>**
217. **<label** for="oneBankStartDate"**>** Start Date**</label>**
218. **<input** type="date" class="form-control" name="oneBankStartDate"**>**
219. **</div>**
220. **<div** class="form-group"**>**
221. **<label** for="oneBankEndDate"**>**End Date**</label>**
222. **<input** type="date" class="form-control" name="oneBankEndDate"**>**
223. **</div>**
224. **</form>**
225. **<div** class="form-group"**>**
226. **<label** for="chartType"**>**Choose Type  Of Chart**</label>**
227. **<select** class="form-control" name="typeOfChart" id="chartType"**>**
228. **<option** value="line"**>**Line**</option>**
229. **<option** value="bar"**>**Bar**</option>**
230. **</select>**
231. **</div>**
232. **<button** type="button" class="btn btn-block nbuRateOneBank" id="buttonsFormat" data-loading-text="Loading..."**>**National Bank Of Ukraine**</button>**
233. **<button** type="button" class="btn btn-block pbRateOneBank" id="buttonsFormat" data-loading-text="Loading..."**>**Private Bank**</button>**
234. **</div>**
235. **</div>**
236. **</div>**
237. **</div>**
238. **</div>**
239. **</div>**
240. **<div** class="col-lg-8"**>**
241. **<div** id="graph-container"**>**
242. **<div** id="chart"**>**
243. **<canvas** id="myChart"**></canvas>**
244. **</div>**
246. **<div** class="sk-cube-grid chart-loader hidden"**>**
247. **<div** class="sk-cube sk-cube1"**></div>**
248. **<div** class="sk-cube sk-cube2"**></div>**
249. **<div** class="sk-cube sk-cube3"**></div>**
250. **<div** class="sk-cube sk-cube4"**></div>**
251. **<div** class="sk-cube sk-cube5"**></div>**
252. **<div** class="sk-cube sk-cube6"**></div>**
253. **<div** class="sk-cube sk-cube7"**></div>**
254. **<div** class="sk-cube sk-cube8"**></div>**
255. **<div** class="sk-cube sk-cube9"**></div>**
256. **</div>**
257. **<div** class="alert alert-success alert-dismissible" role="alert"**>**
258. **<button** type="button" class="close" data-dismiss="alert" aria-label="Close"**><span** aria-hidden="true"**></span></button>**
259. **<strong>**Great! Buy Tendency**</strong><p** class="alert-text"**></p>**
260. **</div>**
261. **<div** class="alert alert-warning alert-dismissible" role="alert"**>**
262. **<button** type="button" class="close" data-dismiss="alert" aria-label="Close"**><span** aria-hidden="true"**>**×**</span></button>**
263. **<strong>**Warning! Buy Tendency:**</strong>** **<p** class= "alert-text"**></p>**
264. **</div>**
265. **</div>**
266. **</div>**
267. **</div>**
268. **</div>**
269. **<div** class="container-fluid" id="about"**>**
270. **<section** class="about-us"**>**
271. **<div** class="container"**>**
272. **<div** class="row"**>**
273. **<div** class="nopadding col-xs-12 col-sm-6 col-md-3 about-info"**>**
274. **<ul>**
275. **<li><h4>**ADDRESS:**</h4><span>**Chernivtsi, Ukraine**</span></li>**
276. **<li><h4>**Date of birth:**</h4><span>**04/05/1995**</span> </li>**
277. **<li><h4>**Email:**</h4><span>**[b.emilian7@gmail.com**</span**](mailto:b.emilian7@gmail.com%3c/span)**> </li>**
278. **<li><h4>**Phone:**</h4><span>**+(380) 66 94 03 779**</span> </li>**
279. **<li><h4>**Skype:**</h4><span>**flush95**</span></li>**
280. **<li><h4>**Languages:**</h4><span>**English, Italian,French, Ukrainian**</span></li>**
281. **</ul>**
282. **</div>**
283. **<div** class="col-xs-12 col-md-6 about-content"**>**
284. **<div>**
285. **<h3>**Flush's studio**</h3>**
286. **<span>**Java Development**</span>**
287. Exchange Rates App.**<br>**
288. **<p>**Backend Part Based On: WildFly Application Server**</p>**
289. **<p>**Server Logic is written using: Java EE 8,  Hibernate, JBoss RESTEasy(JAX-RS).**</p>**
290. **<p>**Design Pattern: MVC**</p>**
291. **<p>**Database Used: MySQL**</p>**
292. **<p>**FrontEnd part builded using:BootstrapjQuery,AJAX, Notify.js, Underscore.js and others**</p>**
293. **<ul>**
294. **<li>**
295. **<a** href="https://www.facebook.com/profile. php?id=100011139099245" target="\_blank"**><i** class="fa fa-facebook"**></i></a>**
296. **</li>**
297. **<li>**
298. **<a** href="https://github.com/Flush95/erates/ tree/master/src/main/java/org/flush/erates" target="\_blank"**><i** class="fa fa-github"**></i></a>**
299. **</li>**
300. **<li>**
301. **<a** href="https://vk.com/milio" target="\_blank"**><i** class="fa fa-vk"**></i></a>**
302. **</li>**
303. **</ul>**
304. **</div>**
305. **</div>**
306. **<div** class="nopadding col-xs-12 col-sm-6 col-md-3 about-image"**>**
307. **<img** src="http://www.rv-designs.nl/wp-content/themes/TheNewDesign/dist/images/people.png" alt="" class="img-responsive hoverZoomLink"**>**
308. **</div>**
309. **</div>**
310. **</div>**
311. **</section>**
312. **</div>**
313. **<div** class="container-fluid" id="statisticsContainer"**>**
314. **<div** class="row"**>**
315. **<div** id="jumbotron"**>**
316. **<div** class="page-header"**>**
317. **<h1** class="text-center"**>**Statistics for 2016**</h1>**
318. **</div>**
319. **<div** class="col-lg-6" id="borders"**>**
320. **<canvas** id="assetsChart"**></canvas>**
321. **</div>**
322. **<div** class="col-lg-6" id="borders"**>**
323. **<canvas** id="loansToIndividualsChart"**></canvas>**
324. **</div>**
325. **</div>**
326. **</div>**
327. **</div>**
328. **</div>**
329. **<script** src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"**></script>**
330. **<script** src="js/bootstrap.min.js"**></script>**
331. **<script** src="js/main.js"**></script>**
332. **<script** src="js/Chart.js"**></script>**
333. **<script** src="js/notify.js"**></script>**
334. **<script** src="js/notify.min.js"**></script>**
335. **</body>**
336. **</html>**

Файл main.js

1. $(document).ready(**function**() {
2. /\* Remove Loader From DOM \*/
3. $('.wrapp').remove();
4. $('.globalWrapper').toggleClass('hidden'); /\* Show Site Content \*/
5. /\* Hide modules \*/
6. $('#chartContainer').hide();
7. $('#about').hide();
8. $('#statisticsContainer').hide();
10. $('.alert-success').hide();
11. $('.alert-warning').hide();
13. /\* Events Listeners \*/
14. $('.itemOne').click(**function**() {
15. $('#chartContainer').slideUp();
16. $('#about').slideUp();
17. $('#statisticsContainer').slideUp();
18. $('#mainContainer').show(800);
19. });
20. $('.itemTwo').click(**function**() {
21. $('#mainContainer').slideUp();
22. $('#about').slideUp();
23. $('#chartContainer').slideUp();
24. $('#statisticsContainer').show();
26. });
27. $('.itemThree').click(**function**() {
28. $('#mainContainer').slideUp();
29. $('#about').slideUp();
30. $('#statisticsContainer').slideUp();
31. $('#chartContainer').slideDown(800);
32. });
33. $('.itemFour').click(**function**() {
34. $('#mainContainer').slideUp();
35. $('#chartContainer').slideUp()
36. $('#statisticsContainer').slideUp();
37. $('#about').slideDown(800);
38. });

41. **var** typeOfChart = "";
42. $('#chartType').change(**function**() {
43. **if** ($('#chartType').val() === 'line') typeOfChart = 'line';
44. **else** **if** ($('#chartType').val() === 'bar') typeOfChart = 'bar';
45. });
46. /\* Submit form with AJAX and Export to .xls \*/
47. **function** submitForm(formName, path, method, tableNumber) {
48. **var** formData = $('form[name=' + formName + ']').serializeArray();
49. **var** dataSendObj = {};
51. formData.forEach(**function**(item) {
52. dataSendObj[item.name] = item.value
53. });
55. **return** $.ajax({
56. url : '/rest/currencies/' + path,
57. method : method,
58. data : dataSendObj,
59. dataType : 'json'
60. }).error(**function**(result) {
61. $.notify("Error Code: " + result.responseJSON.errorCode + " Error Message: " + result.responseJSON.errorMessage);
62. }).success(**function**(result) {
64. **var** dataSet = result.map(**function**(item) {
65. **return** item;
66. });
67. **var** tableTemplate = \_.template($("#templateRootTable").html());
68. **var** compiledTableTemplate = tableTemplate({
69. data : dataSet
70. });
71. **if** (tableNumber === 'nbu') {
72. $("#table1").html(compiledTableTemplate);
73. } **else** **if**(tableNumber === 'pb') {
74. $("#table2").html(compiledTableTemplate);
75. }

78. $("#exportButton").click(**function** () {
79. console.log("I am here");
80. // parse the HTML table element having an id=exportTable
81. **var** dataSource = shield.DataSource.create({
82. data: "#tableExport",
83. schema: {
84. type: "table",
85. fields: {
86. Currency: { type: String },
87. Date: {type: String },
88. Buy: { type: String },
89. Sale: { type: String }
90. }
91. }
92. });
94. // when parsing is done, export the data to Excel
95. dataSource.read().then(**function** (data) {
96. **new** shield.exp.OOXMLWorkbook({
97. author: "Flush95",
98. worksheets: [
99. {
100. name: "CurrencyTable",
101. rows: [
102. {
103. cells: [
104. {
105. style: {
106. bold: **true**
107. },
108. type: String,
109. value: "Currency"
110. },
111. {
112. style: {
113. bold: **true**
114. },
115. type: String,
116. value: "Date"
117. },
118. {
119. style: {
120. bold: **true**
121. },
122. type: String,
123. value: "Buy"
124. },
125. {
126. style: {
127. bold: **true**
128. },
129. type: String,
130. value: "Sale"
131. }
132. ]
133. }
134. ].concat($.map(data, **function**(item) {
135. **return** {
136. cells: [
137. { type: String, value: item.Currency },
138. { type: String, value: item.Date },
139. { type: String, value: item.Buy },
140. { type: String, value: item.Sale }
141. ]
142. };
143. }))
144. }
145. ]
146. }).saveAs({
147. fileName: "currencies"
148. });
149. });
150. });


154. });
155. }
157. **function** buildOneBankOneRateChart(formName, path) {
158. **var** formData = $('form[name=' + formName + ']').serializeArray();
159. $(".chart-loader").removeClass('hidden');
160. $('#myChart').hide();
162. **var** dataSendObj = {};
164. formData.forEach(**function**(item) {
165. dataSendObj[item.name] = item.value
166. });
168. **return** $.ajax({
169. url : '/rest/currencies/' + path,
170. method : 'POST',
171. data : dataSendObj,
172. dataType : 'json'
173. }).error(**function**(result) {
174. $.notify("Error Code: " + result.responseJSON.errorCode + " Error Message: " + result.responseJSON.errorMessage);
175. $(".chart-loader").addClass('hidden');
176. $('#myChart').show();
177. }).success(**function**(result) {
178. $(".chart-loader").addClass('hidden');
179. $('#myChart').show();

182. **var** bankName = result.map(**function**(item) {
183. **return** item.bank;
184. });
185. **var** dataSetSale = result.map(**function**(item) {
186. **return** item.sale;
187. });
188. **var** dataSetBuy = result.map(**function**(item) {
189. **return** item.buy;
190. });
191. **var** dataSetRate = result.map(**function**(item) {
192. **return** item.date;
193. });
195. **var** tendency = parseFloat(dataSetBuy[dataSetBuy.length-1]) - parseFloat(dataSetBuy[0]);
197. $('#myChart').remove();
198. $('.alert-text').empty().append(tendency);
200. **if** (tendency > 0) {
201. $('#chart').append('<canvas id="myChart"></canvas>');
202. $('.alert-success').hide();
203. $('.alert-warning').show();
204. }**else** {
205. $('#chart').append('<canvas id="myChart"></canvas>');
206. $('.alert-warning').hide();
207. $('.alert-success').show();
208. }

211. **var** ctx = document.getElementById("myChart");
213. Chart.defaults.global.animation.duration = 2000;
215. **var** myChart = **new** Chart(ctx, {
216. type: typeOfChart != ""? typeOfChart : "line",
217. data: {
218. labels: dataSetRate || [],
219. datasets: [{
220. label: $('#rate').val() + ' Buy',
221. fill: **false**,
222. lineTension: 0.1,
223. backgroundColor: "rgba(75,192,192,0.4)",
224. borderColor: "rgba(75,192,192,1)",
225. borderCapStyle: 'butt',
226. borderDash: [],
227. borderDashOffset: 0.0,
228. borderJoinStyle: 'miter',
229. pointBorderColor: "rgba(75,192,192,1)",
230. pointBackgroundColor: "#fff",
231. pointBorderWidth: 1,
232. pointHoverRadius: 5,
233. pointHoverBackgroundColor: "rgba(75,192,192,1)",
234. pointHoverBorderColor: "rgba(220,220,220,1)",
235. pointHoverBorderWidth: 2,
236. pointRadius: 1,
237. pointHitRadius: 10,
238. scaleFontColor: "white",
239. data: dataSetBuy || []
240. }, {
241. label: $('#rate').val() + ' Sale',
242. borderColor: "red",
243. data: dataSetSale || []
244. }]
246. },
247. options: {
248. scales: {
249. yAxes: [{
250. ticks: {
251. beginAtZero: **false**,
252. fontColor: "white",
253. fontSize: 13,
254. fontFamily: "'Damion', cursive"
255. },
256. }],
257. xAxes: [{
258. ticks: {
259. fontColor: "white",
260. fontSize: 13,
261. fontFamily: "'Damion', cursive",
262. }
263. }]
264. },
265. responsive: **true**,
266. maintainAspectRatio: **true**,
267. title: {
268. display: **true**,
269. fontColor: "white",
270. text: bankName[0]
271. },
272. }
273. });
274. });
275. }
276. /\* Options Click(submit form buttons) \*/
277. $('.nbuSingle').on('click', **function**() {
278. **var** $btn = $(**this**).button('loading');
279. submitForm("oneDateForm", "buildnbutable", "POST", "nbu").success(**function**() {
280. $btn.button('reset');
281. }).error(**function**() {
282. $btn.button('reset');
283. });
284. });
285. $('.pbSingle').on('click', **function**() {
286. **var** $btn = $(**this**).button('loading');
287. submitForm("oneDateForm", "buildpbtable", "POST", "pb").success(**function**() {
288. $btn.button('reset');
289. }).error(**function**() {
290. $btn.button('reset');
291. });
292. });


296. $('.nbuDiapason').on('click', **function**() {
297. **var** $btn = $(**this**).button('loading');
298. submitForm("diapasonDatesForm", "diapasonnbutable", "POST", "nbu").success(**function**() {
299. $btn.button('reset');
300. }).error(**function**() {
301. $btn.button('reset');
302. });
303. });
304. $('.pbDiapason').on('click', **function**() {
305. **var** $btn = $(**this**).button('loading');
306. submitForm("diapasonDatesForm", "diapasonpbtable", "POST", "pb").success(**function**() {
307. $btn.button('reset');
308. }).error(**function**() {
309. $btn.button('reset');
310. });
311. });

314. $('.nbuRateOneBank').on('click', **function**() {
315. **var** $btn = $(**this**).button('loading');
316. buildOneBankOneRateChart("byRateOneBankForm", "nbuonebank").success(**function**() {
317. $btn.button('reset');
318. }).error(**function**() {
319. $btn.button('reset');
320. });
321. });
322. $('.pbRateOneBank').on('click', **function**() {
323. **var** $btn = $(**this**).button('loading');
324. buildOneBankOneRateChart("byRateOneBankForm", "pbonebank").success(**function**() {
325. $btn.button('reset');
326. }).error(**function**() {
327. $btn.button('reset');
328. });
329. });

332. $('.rateDiffBanks').on('click', **function**() {
333. **var** $btn = $(**this**).button('loading');
334. buildAllBanksAllRatesCharts("byRateDifferentBanksForm", "differentbanks").success(**function**() {
335. $btn.button('reset');
336. }).error(**function**() {
337. $btn.button('reset');
338. });
339. });
340. /\* Statistics for 2016 \*/
341. **var** dataAssets = {
342. labels: [
343. "PrivatBank", "OschadBank", "UkreximBank", "ProminvestBank",
344. "Russian Sberbank", "Raiffeisen Bank Aval", "Ukrsotsbank",
345. "Alfa-Bank", "UkrSibBank", "Pumb"
346. ],
347. datasets: [
348. {
349. backgroundColor: "rgba(255,99,132,0.2)",
350. borderColor: "rgba(255,99,132,1)",
351. pointBackgroundColor: "rgba(255,99,132,1)",
352. pointBorderColor: "#fff",
353. pointHoverBackgroundColor: "#fff",
354. pointHoverBorderColor: "rgba(255,99,132,1)",
355. data: [29, 18, 17, 7, 6, 5, 5, 5, 4, 4],
357. label: 'Top banks in terms of assets %'
358. }]
359. };
360. **var** assetsChart = document.getElementById("assetsChart");
361. Chart.defaults.global.animation.duration = 2000;
362. **var** assetsChart = **new** Chart(assetsChart, {
363. type: 'radar',
364. data: dataAssets,
365. animation:{
366. animateScale:**true**,
367. animateRotate: **true**
368. }
369. });



374. **var** dataLoans = {
375. labels: [
376. "PrivatBank", "Ukrsotsbank", "UkrSibBank", "Raiffeisen Bank Aval",
377. "OTP Bank", "Alfa-Bank", "Pumb", "Platinum Bank",
378. "OschadBank", "UniversalBank",
379. ],
380. datasets: [
381. {
382. backgroundColor: "rgba(255,99,132,0.2)",
383. borderColor: "orange",
384. pointBackgroundColor: "orange",
385. pointBorderColor: "#fff",
386. pointHoverBackgroundColor: "#fff",
387. pointHoverBorderColor: "orange",
388. data: [33, 22, 9, 9, 8, 5, 4, 4, 4, 2],
389. label: 'Top banks on loans to individuals in %',
390. }]
391. };
392. **var** assetsChart = document.getElementById("loansToIndividualsChart");
393. Chart.defaults.global.animation.duration = 2000;
394. **var** assetsChart = **new** Chart(assetsChart, {
395. type: 'radar',
396. data: dataLoans,
397. animation:{
398. animateScale: **true**,
399. animateRotate: **true**
400. }
401. });
402. });

Файл style.css

1. ::-webkit-scrollbar {  /\* Customize ScrollBar \*/
2. **width**: 12px;
3. }
4. ::-webkit-scrollbar-track {
5. **background-color**: #2c3e50;
6. **border-left**: 1px solid #2F4F4F;
7. }
8. ::-webkit-scrollbar-thumb {
9. **background-color**: #1abc9c;
10. }
11. ::-webkit-scrollbar-thumb:hover {
12. **background-color**: #1abc9c;
13. }
14. .navbar {
15. **background**: #333;
16. }
18. body {
19. **background**: #2c3e50/\*linear-gradient(to right, #2c3e50 15%, #1abc9c)radial-gradient(circle farthest-corner at center, #3C4B57 0%, #1C262B 100%)\*/;
20. **color**: #2F4F4F;
21. **font-family**: 'Damion', cursive;
22. **height**: 100%;
23. }
24. .globalWrapper {
26. }
27. #title {
28. **color**: #2F4F4F;
29. **font-family**: 'Damion', cursive;
30. **font-size**: 20em;
31. **text-shadow**: 3px 3px 0px #2c3e50;
32. }
34. #mainPanel {
35. **color**: black;
36. }
37. #pad {
38. **padding**: 2%;
39. }
41. #panelHead {
42. **font-weight**: bold;
43. **background**: #2c3e50;
44. **color**: white;
45. }
47. table tbody tr :hover {
48. **background**: #1abc9c;
49. }
50. /\* Menu Style \*/
51. .menu {
52. top: 0;
53. left: 0;
54. position : fixed;
55. **height**: 100%;
56. **width**: 5%;
57. **min-width**: 65px;
58. **background**: #2c3e50;
59. transition: width 1s;
60. **overflow**: hidden;
61. **font-family**: 'Damion', cursive;
62. **z-index**: 99;
63. **position**: fixed;
64. box-shadow: 2px 2px 2px #333;
65. }
67. .menu:hover {
68. **width**: 250px;
69. }
71. .menuItem .span {
72. **position**: absolute;
73. left: 80px;
74. top: 20px;
75. transition: color 1s;
76. **color**: #1abc9c;
77. }
78. .menuItem .spanOne {
79. **position**: absolute;
80. left: 80px;
81. top: 20px;
82. transition: color 1s;
83. **color**: #1abc9c;
84. }
85. .menuItem .span a {
86. **text-decoration**: none;
87. }
89. .menu:hover .menuItem .span {
90. **color**: #1abc9c;
91. }
92. .menu:hover .menuItem .spanOne {
93. **color**: #2c3e50;
94. }
96. .menuItem {
97. **position**: relative;
98. **padding**: 20px;
99. transition: border .5s, background .2s;
100. }
102. .menuItem:hover {
103. **background**: #F5F5F5;
104. **cursor**: pointer;
105. }
107. .itemOneCustom {
108. **background**: #1abc9c;
109. **color**: #2c3e50;
110. **border-left**: 5px solid #1abc9c;
111. **border-bottom**: 1px solid;
112. }
114. .itemOneCustom:hover {
115. **border-left**: 20px solid #1abc9c;
116. }
118. .itemTwo {
119. **border-left**: 5px solid;
120. **color**: #1abc9c;
121. }
123. .itemTwo:hover {
124. **border-left**: 20px solid;
125. }
127. .itemThree {
128. **border-left**: 5px solid;
129. **color**: #1abc9c;
130. }
132. .itemThree:hover {
133. **border-left**: 20px solid;
134. }
136. .itemFour {
137. **border-left**: 5px solid;
138. **color**: #1abc9c;
139. }
141. .itemFour:hover {
142. **border-left**: 20px solid;
143. }
145. .itemFive {
146. **border-left**: 5px solid;
147. **color**: #1abc9c;
148. }
150. .itemFive:hover {
151. **border-left**: 20px solid;
152. }
154. .panel {
155. border-radius: 0;
156. box-shadow: 0 4px 8px 0 #333, 0 6px 20px 0 #333;
157. }
159. button {
160. **background-color**: #2c3e50;
161. **color**: #1abc9c;
162. }
163. button:hover {
164. **background-color**: #1abc9c;
165. **color**: #2c3e50;
166. }

169. .panel {
170. **border**: 0px;
171. **color**: #2c3e50;
172. }
173. .panel-heading {
174. border-top-left-radius: 0;
175. border-top-right-radius: 0;
176. **background-color**: #2c3e50;
177. **border**: 0px solid;
178. **color**: white;
179. }
181. .pad {
182. **padding-top**: 3.5%;
183. **margin-left**: 55px;
184. **border-right**: 1px solid white;
185. }
186. .mobilePadding {
187. **height**: 50px;
188. }
189. .pad-table {
190. **margin-top**: 1.4%;
191. }
193. #bloks {
194. **margin-top**: 5px;
195. **margin-bottom**: 5px;
196. **float**: none;
197. }

200. #buttonsFormat {
201. **width**: 100%;
202. }
204. #graph-container {
205. **padding-top**: 3%;
206. }
207. #myChart {
208. **width**: 100%;
209. **height**: auto;
210. }
212. #jumbotron {
213. **padding-top**: 5vh;
214. **height**: 100vh;
215. **background**: white;
216. }
218. #about {
219. **position**: relative;
220. overflow-x: hidden;
221. **margin**: 0;
222. **height**: 100vh;
223. **padding-left**: 50px;
224. **padding-top**: 5vh;
225. **color**: #afaeae;
226. **font**: 300 14px Lato,sans-serif;
227. **letter-spacing**: .5px;
228. **background**: #2c3e50;
229. }

232. h1,h2,h3,h4 {
233. **margin**: 0 0 10px;
234. **color**: #1e1e1e;
235. **text-transform**: uppercase
236. }
238. h2,h3 {
239. **font-weight**: 900;
240. **text-transform**: none
241. }
243. h1 {
244. **font-weight**: 900;
245. **font-size**: 64px;
246. **letter-spacing**: 4px
247. }
249. h2 {
250. **font-size**: 32px;
251. **letter-spacing**: 2px;
252. **text-align**: left
253. }
255. h2::before {
256. **position**: absolute;
257. top: 13px;
258. left: 0;
259. -webkit-transform: rotate(45deg);
260. -ms-transform: rotate(45deg);
261. transform: rotate(45deg);
262. **background**: #dcdcdc;
263. **width**: 8px;
264. **height**: 8px
265. }
267. h3 {
268. **font-size**: 16px;
269. **letter-spacing**: 1px
270. }
272. li {
273. **list-style-type**: none;
274. }
275. .nopadding{
276. **padding**: 0;
277. }
278. .about-image img{
279. **height**: 320px;
280. }
281. .about-info ul li {
282. **padding**: 14px 0;
283. **border-bottom**: 1px dashed #1abc9c;
284. }
285. .about-info ul li h4 {
286. **display**: inline-block;
287. **margin-bottom**: 5px;
288. **padding-right**: 5px;
289. **font-weight**: 700;
290. **font-size**: 12px;
291. **letter-spacing**: 1.5px;
292. **color**: #EF971D;
293. }
294. .about-info{
295. **color**: #fff;
296. }
297. .about-content div {
298. **background**: #2e2e2e;
299. **position**: relative;
300. **padding**: 30px;
301. **padding-left**: 50px;
302. }
304. .about-content h3 {
305. **font-size**: 42px;
306. **letter-spacing**: 1.5px
307. }
309. .about-content span {
310. **display**: block;
311. **margin-bottom**: 10px;
312. **color**: #EF971D;
313. **font-weight**: 300;
314. **font-size**: 18px;
315. **letter-spacing**: 1px
316. }
317. .about-content p{
318. **line-height**: 24px;
319. }
320. .about-content ul{
321. **margin**: 0;
322. **padding**: 0;
323. }
324. .about-content ul li {
325. **display**: inline-block;
326. **margin-right**: 8px;
327. **text-align**: center;
328. **margin-top**: 20px
329. }
331. .about-content ul li:last-child {
332. **margin**: 0
333. }
335. .about-content ul li a {
336. **display**: block;
337. **width**: 42px;
338. **height**: 42px;
339. **color**: #a0a0a0;
340. **font-size**: 13px;
341. **line-height**: 38px;
342. }
344. .about-content ul li a{
345. **background**: #1e1e1e;
346. **border-color**: #1e1e1e;
347. **color**: #fff;
348. }
349. .logo {
350. **width**: 33.3%;
351. **height**: 33.3%;
352. }
354. .sk-cube-grid {
355. **margin**: 15% auto;
356. **z-index**: 999;
357. **width**: 200px;
358. **height**: 200px;
359. }
361. .sk-cube-grid .sk-cube {
362. **width**: 33%;
363. **height**: 33%;
364. **background-color**: white;
365. **float**: left;
366. -webkit-animation: sk-cubeGridScaleDelay 1.3s infinite ease-in-out;
367. animation: sk-cubeGridScaleDelay 1.3s infinite ease-in-out;
368. }
369. .sk-cube-grid .sk-cube1 {
370. -webkit-animation-delay: 0.2s;
371. animation-delay: 0.2s; }
372. .sk-cube-grid .sk-cube2 {
373. -webkit-animation-delay: 0.3s;
374. animation-delay: 0.3s; }
375. .sk-cube-grid .sk-cube3 {
376. -webkit-animation-delay: 0.4s;
377. animation-delay: 0.4s; }
378. .sk-cube-grid .sk-cube4 {
379. -webkit-animation-delay: 0.1s;
380. animation-delay: 0.1s; }
381. .sk-cube-grid .sk-cube5 {
382. -webkit-animation-delay: 0.2s;
383. animation-delay: 0.2s; }
384. .sk-cube-grid .sk-cube6 {
385. -webkit-animation-delay: 0.3s;
386. animation-delay: 0.3s; }
387. .sk-cube-grid .sk-cube7 {
388. -webkit-animation-delay: 0s;
389. animation-delay: 0s; }
390. .sk-cube-grid .sk-cube8 {
391. -webkit-animation-delay: 0.1s;
392. animation-delay: 0.1s; }
393. .sk-cube-grid .sk-cube9 {
394. -webkit-animation-delay: 0.2s;
395. animation-delay: 0.2s; }
397. @-webkit-keyframes sk-cubeGridScaleDelay {
398. 0%, 70%, 100% {
399. -webkit-transform: scale3D(1, 1, 1);
400. transform: scale3D(1, 1, 1);
401. } 35% {
402. -webkit-transform: scale3D(0, 0, 1);
403. transform: scale3D(0, 0, 1);
404. }
405. }
407. @keyframes sk-cubeGridScaleDelay {
408. 0%, 70%, 100% {
409. -webkit-transform: scale3D(1, 1, 1);
410. transform: scale3D(1, 1, 1);
411. } 35% {
412. -webkit-transform: scale3D(0, 0, 1);
413. transform: scale3D(0, 0, 1);
414. }
415. }

Файл Url\_Container.java

1. **package** org.flush.erates.container;
3. **public** **interface** UrlContainer {  /\* Interface with links to API \*/
4. **static** **final** String PB\_TODAY\_URL = "https://api.privatbank.ua/p24api/pubinfo?json&exchange&coursid=5";
5. **static** **final** String PB\_NEEDED\_DATE\_URL = "https://api.privatbank.ua/p24api/exchange\_rates?json&date=";
6. **static** **final** String NBU\_TODAY\_URL = "https://api.privatbank.ua/p24api/pubinfo?json&exchange&coursid=3";
7. **static** **final** String NBU\_NEEDED\_DATE\_URL = "https://api.privatbank.ua/p24api/exchange\_rates?json&date=";
8. }

Файл DatabaseClass.java

1. **package** org.flush.erates.database;
3. **import** java.util.LinkedList;
4. **import** java.util.List;
6. **import** org.flush.erates.dto.Rates;
7. **import** org.hibernate.Criteria;
8. **import** org.hibernate.Session;
9. **import** org.hibernate.SessionFactory;
10. **import** org.hibernate.cfg.Configuration;
11. **import** org.hibernate.criterion.Restrictions;
13. **public** **class** DatabaseClass {
14. **static** SessionFactory sessionFactory = **null**;
15. /\* Receive data from database considering date and bank \*/
16. **public** **static** List<Rates> getNeededDateList(String date, String bank) {
17. SessionFactory sessionFactory = **new** Configuration().configure()
18. .buildSessionFactory();
19. Session session = sessionFactory.openSession();
20. session.beginTransaction();  //looking for commit start
22. Criteria criteria = session.createCriteria(Rates.**class**);
23. criteria.add(Restrictions.like("bank", bank)); //criteria for receiving data from database
24. criteria.add(Restrictions.like("date", date));

27. **final** List<Rates> rates = **new** LinkedList<>();
28. **for** (**final** Object obj : criteria.list()) {//Converting LinkedList to List for
29. rates.add((Rates) obj);   //exclude type warning
30. }
31. session.getTransaction().commit();   //push session
32. **return** rates;
33. }
34. /\* Receive unique Rate \*/
35. **public** **static** List<Rates> getRate(Rates rate) {
36. sessionFactory = **new** Configuration().configure().buildSessionFactory();
37. Session session = sessionFactory.openSession();
38. session.beginTransaction();
40. Criteria criteria = session.createCriteria(Rates.**class**);
41. criteria.add(Restrictions.like("bank", rate.getBank()));
42. criteria.add(Restrictions.like("date", rate.getDate()));
44. **if** (rate.getRate().length() > 0) {
45. criteria.add(Restrictions.like("rate", rate.getRate()));
46. }
48. **final** List<Rates> rates = **new** LinkedList<>();
49. **for** (**final** Object obj : criteria.list()) {
50. rates.add((Rates) obj);
51. }
52. session.getTransaction().commit();
53. **return** rates;
54. }
55. /\* Insert List of Rates into Database \*/
56. **public** **static** **void** insertToRates(List<Rates> listRates) {
57. Rates local = **new** Rates();
58. **for** (Rates rates: listRates) {
59. SessionFactory sessionFactory = **new** Configuration().configure()
60. .buildSessionFactory();
61. Session session = sessionFactory.openSession();
62. session.beginTransaction();
64. local.setBank(rates.getBank());  //pushing info to entity class for writing
65. local.setRate(rates.getRate());  //it to database
66. local.setDate(rates.getDate());
67. local.setBuy(rates.getBuy());
68. local.setSale(rates.getSale());
70. session.save(local);  //save entity object to database
71. session.getTransaction().commit();
73. }
74. }
75. /\* Insert Single Rate \*/
76. **public** **static** **void** insertSingleToRates(Rates rateObj) {
77. Rates local = **new** Rates();
79. SessionFactory sessionFactory = **new** Configuration().configure()
80. .buildSessionFactory();
81. Session session = sessionFactory.openSession();
82. session.beginTransaction();
83. local.setBank(rateObj.getBank());
84. local.setRate(rateObj.getRate());
85. local.setDate(rateObj.getDate());
86. local.setBuy(rateObj.getBuy());
87. local.setSale(rateObj.getSale());
89. session.save(local);
90. session.getTransaction().commit();
91. }
92. }

Файл DateLogic.java

1. **package** org.flush.erates.date;
3. **import** java.time.LocalDate;
5. **public** **class** DateLogic {
6. /\* Get today date \*/
7. **public** **static** String getTodayDate() {
8. LocalDate localDate = LocalDate.now();
9. **return** localDate.toString();
10. }
11. **public** **static** String convertResponseDate(String dateFromResponse) {
12. String dateM[] = dateFromResponse.split(" ");
13. **return** dateM[0];
14. }
15. /\* Convert 2015.11.21 to 21.11.2015 \*/
16. **public** **static** String formatPbDate(String date) {
17. String dateY\_M\_D[] = date.split("-");
19. **return** dateY\_M\_D[2] + "." + dateY\_M\_D[1] + "." + dateY\_M\_D[0];
20. }

23. }

Файл ErrorMessage.java

1. **package** org.flush.erates.dto;
3. **import** javax.xml.bind.annotation.XmlRootElement;
5. @XmlRootElement
6. **public** **class** ErrorMessage {  /\* Setting Error Message Class for custom exceptions \*/
7. **private** String errorMessage;
8. **private** **int** errorCode;
10. **public** ErrorMessage() {
12. }
13. **public** ErrorMessage(String errorMessage, **int** errorCode) {
14. **super**();
15. **this**.errorMessage = errorMessage;
16. **this**.errorCode = errorCode;
17. }
18. **public** String getErrorMessage() {
19. **return** errorMessage;
20. }
21. **public** **void** setErrorMessage(String errorMessage) {
22. **this**.errorMessage = errorMessage;
23. }
24. **public** **int** getErrorCode() {
25. **return** errorCode;
26. }
27. **public** **void** setErrorCode(**int** errorCode) {
28. **this**.errorCode = errorCode;
29. }
31. }

Файл Rates.java

1. **package** org.flush.erates.dto;
3. **import** javax.persistence.Entity;
4. **import** javax.persistence.GeneratedValue;
5. **import** javax.persistence.GenerationType;
6. **import** javax.persistence.Id;
7. **import** javax.xml.bind.annotation.XmlRootElement;
8. **import** javax.xml.bind.annotation.XmlTransient;
10. @XmlRootElement
11. @Entity
12. **public** **class** Rates {  /\* Entity Class for database and RESTfull Web Services \*/
13. @Id @GeneratedValue(strategy=GenerationType.AUTO)
14. **private** **long** id;
15. **private** String bank;
16. **private** String rate;
17. **private** String date;
18. **private** Double buy;
19. **private** Double sale;
21. **public** Rates() {}
22. //getters and setters
23. **public** Rates(String bank, String rate, String date, Double buy, Double sale) {
24. **this**.bank = bank;
25. **this**.rate = rate;
26. **this**.date = date;
27. **this**.buy = buy;
28. **this**.sale = sale;
29. }
30. @XmlTransient
31. **public** **long** getId() {
32. **return** id;
33. }
34. **public** **void** setId(**long** id) {
35. **this**.id = id;
36. }
37. **public** String getBank() {
38. **return** bank;
39. }
40. **public** **void** setBank(String bank) {
41. **this**.bank = bank;
42. }
43. **public** String getRate() {
44. **return** rate;
45. }
46. **public** **void** setRate(String rate) {
47. **this**.rate = rate;
48. }
49. **public** String getDate() {
50. **return** date;
51. }
52. **public** **void** setDate(String date) {
53. **this**.date = date;
54. }
55. **public** Double getBuy() {
56. **return** buy;
57. }
58. **public** **void** setBuy(Double buy) {
59. **this**.buy = buy;
60. }
61. **public** Double getSale() {
62. **return** sale;
63. }
64. **public** **void** setSale(Double sale) {
65. **this**.sale = sale;
66. }
68. }

Файл ErrorMessage.java

1. **package** org.flush.erates.errormessages;
3. **import** javax.ws.rs.core.Response;
4. **import** javax.ws.rs.core.Response.Status;
6. **import** org.flush.erates.dto.ErrorMessage;
8. **public** **class** ErrorMessages {  /\* Class for building error message \*/
9. **private** **static** ErrorMessage message;
10. //build custom exception for clean diapason
11. **public** **static** Response diapasonBadDatesException() {
12. message = **new** ErrorMessage("Bad Date", 400);
13. **return** Response.status(Status.BAD\_REQUEST)
14. .entity(message)
15. .build();
16. }
17. //custom exception for start date before end date
18. **public** **static** Response diapasonDateBeforeException() {
19. message = **new** ErrorMessage("Start Date Is Before End Date", 400);
20. **return** Response.status(Status.BAD\_REQUEST)
21. .entity(message)
22. .build();
23. }
24. //rate not found exception
25. **public** **static** Response rateForDateException() {
26. message = **new** ErrorMessage("Rate for this date Not Found", 404);
27. **return** Response.status(Status.BAD\_REQUEST)
28. .entity(message)
29. .build();
30. }
32. }

Файл Parser.java

1. **package** org.flush.erates.parsing;
3. **import** java.io.BufferedReader;
4. **import** java.io.IOException;
5. **import** java.io.InputStreamReader;
6. **import** java.net.HttpURLConnection;
7. **import** java.net.URL;
8. **import** java.util.ArrayList;
9. **import** java.util.List;
11. **import** javax.ws.rs.WebApplicationException;
13. **import** org.flush.erates.database.DatabaseClass;
14. **import** org.flush.erates.date.DateLogic;
15. **import** org.flush.erates.dto.Rates;
16. **import** org.flush.erates.errormessages.ErrorMessages;
17. **import** org.json.JSONArray;
18. **import** org.json.JSONObject;
20. **public** **class** Parser {
22. **private** URL urlObj;
23. **private** HttpURLConnection connection;
24. **private** **int** responseCode;
25. **private** List<Rates> listRates;
26. **private** Rates rateObj = **null**;
27. /\* Open connection to api and check response code \*/
28. **public** HttpURLConnection openConnection(String url, String date) {
29. **try** {
30. urlObj = **new** URL(date != "" ? (url.concat(date)) : url);//if date not equals at null – concat date to url, else return url
31. connection = (HttpURLConnection) urlObj.openConnection();//open stream
32. connection.setRequestMethod("POST");//sending request method
33. responseCode = connection.getResponseCode();//receive response from conn…
34. } **catch** (Exception e) {
35. e.printStackTrace();
36. }
37. **return** responseCode == HttpURLConnection.HTTP\_OK ? connection : **null**;
38. }
39. /\* read http connection and convert it to string \*/
40. **public** String getJSONString(HttpURLConnection connection) {
41. StringBuffer responseBuffer = **new** StringBuffer();
42. **try** (BufferedReader in = **new** BufferedReader(**new** InputStreamReader
43. (connection.getInputStream()));) {
44. String inputLine;
45. **while** ((inputLine = in.readLine()) != **null**) { //write connection obj
46. responseBuffer.append(inputLine);
47. }
48. } **catch** (IOException e) {
49. e.printStackTrace();
50. }
51. **return** responseBuffer.toString();
52. }
54. **public** List<Rates> parseTodayPB(String jsonStr, String bank) {  /\* Parse today rates from response \*/
55. JSONArray array = **new** JSONArray(jsonStr);
56. listRates = **new** ArrayList<>();
58. **if** (array.equals("[]") || array.equals("{}") || array.length() <= 0) {
59. **throw** **new** WebApplicationException(ErrorMessages.diapasonBadDatesException());
60. } // check cunstom exception
61. todayPbService(array, bank);
62. **return** listRates;
63. }
65. **public** **void** todayPbService(JSONArray array, String bank) {
66. **for** (**int** i = 0; i < array.length(); i++) {
67. **if** ((((JSONObject) array.get(i)).getString("ccy").toString())
68. .equals("EUR")) {
69. listRates.add(**new** Rates(bank, ((JSONObject) array.get(i))
70. .getString("ccy").toString(),
71. DateLogic.formatPbDate(DateLogic
72. .getTodayDate()),
73. ((JSONObject)array.get(i)).getDouble("buy"),
74. ((JSONObject) array.get(i))
75. .getDouble("sale")));
76. } **else** **if** ((((JSONObject) array.get(i)).getString("ccy").toString())
77. .equals("USD")) {
78. listRates.add(**new** Rates(bank, ((JSONObject) array.get(i))
79. .getString("ccy").toString(),
80. DateLogic.formatPbDate(DateLogic.getTodayDate()),
81. ((JSONObject) array.get(i))
82. .getDouble("buy"),
83. ((JSONObject) array.get(i)).getDouble("sale")));
84. } **else** **if** ((((JSONObject) array.get(i)).getString("ccy")
85. .toString()).equals("RUR")) {
86. listRates.add(**new** Rates(bank, "RUB", DateLogic
87. .formatPbDate(DateLogic.getTodayDate()),
88. ((JSONObject) array.get(i)).getDouble("buy"),
89. ((JSONObject) array.get(i)).getDouble("sale")));
90. }
91. }
92. }
93. /\* Parse specific date in PB \*/
94. **public** List<Rates> parseSpecificDatePB(String jsonStr, String date, String bank) {
95. listRates = **new** ArrayList<>();
97. JSONObject firstLvl = **new** JSONObject(jsonStr);
98. JSONArray array = (JSONArray) firstLvl.get("exchangeRate");
100. **if** (array.equals("[]") || array.equals("{}") || array.length() <= 0) {
101. **throw** **new** WebApplicationException(ErrorMessages
102. .diapasonBadDatesException());
103. }
104. **if** (bank.equals("PrivatBank"))
105. specificDatePbService(array, "PrivatBank", "purchaseRate",
106. "saleRate", date);
107. **else** **if** (bank.equals("NBU"))
108. specificDatePbService(array, "NBU", "purchaseRateNB",
109. "saleRateNB", date);
110. **return** listRates;
111. }
112. /\* Convert data to List \*/
113. **public** **void** specificDatePbService(JSONArray array, String bank,
114. String getBuy, String getSale, String date) {
116. **for** (**int** i = 0; i < array.length(); i++) {
118. **if** ((((JSONObject) array.get(i)).getString("currency").toString())
119. .equals("EUR")) {
120. rateObj = **new** Rates(bank, ((JSONObject) array.get(i))
121. .getString("currency").toString(), date,
122. ((JSONObject) array.get(i)).getDouble(getBuy),
123. ((JSONObject) array.get(i)).getDouble(getSale));
124. **if** (DatabaseClass.getRate(rateObj).size() > 0)
125. listRates.addAll(DatabaseClass.getRate(rateObj));
126. **else**
127. listRates.add(rateObj);
128. } **else** **if** ((((JSONObject) array.get(i)).getString("currency")
129. .toString()).equals("USD")) {
130. rateObj = **new** Rates(bank, ((JSONObject) array.get(i))
131. .getString("currency").toString(),
132. date,  ((JSONObject) array.get(i)).getDouble(getBuy),
133. ((JSONObject) array.get(i)).getDouble(getSale));
134. **if** (DatabaseClass.getRate(rateObj).size() > 0)
135. listRates.addAll(DatabaseClass.getRate(rateObj));
136. **else**
137. listRates.add(rateObj);
138. } **else** **if** ((((JSONObject) array.get(i)).getString("currency")
139. .toString()).equals("RUB")) {
140. rateObj = **new** Rates(bank, ((JSONObject) array.get(i)).getString("currency").toString(), date,
141. ((JSONObject) array.get(i)).getDouble(getBuy),
142. ((JSONObject) array.get(i)).getDouble(getSale));
144. **if** (DatabaseClass.getRate(rateObj).size() > 0)
145. listRates.addAll(DatabaseClass.getRate(rateObj));
146. **else**
147. listRates.add(rateObj);
148. }
149. }
150. }
152. /\* getting nbu rates \*/
153. **public** List<Rates> getNBUDataSource(String jsonStr, String date) {
154. JSONObject obj = **new** JSONObject(jsonStr);
155. JSONObject eurObj = (JSONObject) obj.get("eur");
156. JSONObject usdObj = (JSONObject) obj.get("usd");
157. JSONObject rubObj = (JSONObject) obj.get("rub");
159. listRates = **new** ArrayList<>();
161. listRates.add(**new** Rates("NBU", "EUR", DateLogic.formatPbDate(date), ((JSONObject) eurObj).getDouble("ask"),
162. ((JSONObject) eurObj).getDouble("bid")));
163. listRates.add(**new** Rates("NBU", "USD", DateLogic.formatPbDate(date), ((JSONObject) usdObj).getDouble("ask"),
164. ((JSONObject) usdObj).getDouble("bid")));
165. listRates.add(**new** Rates("NBU", "RUB", DateLogic.formatPbDate(date), ((JSONObject) rubObj).getDouble("ask"),
166. ((JSONObject) rubObj).getDouble("bid")));
168. **return** listRates;
169. }

172. **public** List<Rates> getNbuDiapasonByDays(List<String> jsonStrList, List<String> dateList) {
173. listRates = **new** ArrayList<>();
174. diapasonService(jsonStrList, dateList, "NBU", "purchaseRateNB", "saleRateNB");
176. **return** listRates;
177. }
179. **public** List<Rates> getPbDiapasonByDays(List<String> jsonStrList, List<String> dateList) {
180. listRates = **new** ArrayList<>();
181. diapasonService(jsonStrList, dateList, "PrivatBank", "purchaseRate", "saleRate");
183. **return** listRates;
184. }
185. /\* checking diapasone of dates \*/
186. **public** **void** diapasonService(List<String> jsonStrList, List<String> dateList, String bank, String getBuy, String getSale) {
187. **for** (**int** i = 0; i < jsonStrList.size(); i++) {
188. **if** (jsonStrList.get(i).equals("[]") || jsonStrList.get(i).equals("{}") || jsonStrList.get(i).length() <= 0)
189. **continue**;
191. String date = dateList.get(i);
193. JSONObject firstLvl = **new** JSONObject(jsonStrList.get(i));
194. JSONArray array = (JSONArray) firstLvl.get("exchangeRate");
196. **for** (**int** j = 0; j < array.length(); j++) {
197. **if** ((((JSONObject) array.get(j)).getString("currency").toString()).equals("EUR")) {
198. rateObj = **new** Rates(bank, ((JSONObject) array.get(j)).getString("currency").toString(),
199. DateLogic.formatPbDate(date), ((JSONObject) array.get(j)).getDouble(getBuy),
200. ((JSONObject) array.get(j)).getDouble(getSale));
201. listRates.add(rateObj);
202. **if** (DatabaseClass.getRate(rateObj).size() > 0) {
203. listRates.addAll(DatabaseClass.getRate(rateObj));
204. } **else** {
205. listRates.add(rateObj);
206. DatabaseClass.insertSingleToRates(rateObj);
207. }
209. } **else** **if** ((((JSONObject) array.get(j)).getString("currency").toString()).equals("USD")) {
210. rateObj = **new** Rates(bank, ((JSONObject) array.get(j)).getString("currency").toString(),
211. DateLogic.formatPbDate(date), ((JSONObject) array.get(j)).getDouble(getBuy),
212. ((JSONObject) array.get(j)).getDouble(getSale));
213. listRates.add(rateObj);
214. **if** (DatabaseClass.getRate(rateObj).size() > 0) {
215. listRates.addAll(DatabaseClass.getRate(rateObj));
216. } **else** {
217. listRates.add(rateObj);
218. DatabaseClass.insertSingleToRates(rateObj);
219. }
221. } **else** **if** ((((JSONObject) array.get(j)).getString("currency").toString()).equals("RUB")) {
222. rateObj = **new** Rates(bank, ((JSONObject) array.get(j)).getString("currency").toString(),
223. DateLogic.formatPbDate(date), ((JSONObject) array.get(j)).getDouble(getBuy),
224. ((JSONObject) array.get(j)).getDouble(getSale));
225. listRates.add(rateObj);
226. **if** (DatabaseClass.getRate(rateObj).size() > 0) {
227. listRates.addAll(DatabaseClass.getRate(rateObj));
228. } **else** {
229. listRates.add(rateObj);
230. DatabaseClass.insertSingleToRates(rateObj);
231. }
233. }
234. }
235. }
236. }
238. /\* Get nbu rates for diapasone \*/
239. **public** List<Rates> getNbuByRate(String rate, List<String> jsonStrList, List<String> dateList) {
240. listRates = **new** ArrayList<>();
241. oneRateChartService(rate, jsonStrList, dateList, "NBU", "purchaseRateNB", "saleRateNB");
243. **return** listRates;
244. }
245. /\* get Pb rates by needed rate \*/
246. **public** List<Rates> getPbByRate(String rate, List<String> jsonStrList, List<String> dateList) {
247. listRates = **new** ArrayList<>();
248. oneRateChartService(rate, jsonStrList, dateList, "PrivatBank", "purchaseRate", "saleRate");
250. **return** listRates;
251. }
252. /\* Rates for chart \*/
253. **public** **void** oneRateChartService(String rate, List<String> jsonStrList, List<String> dateList, String bank,
254. String getBuy, String getSale) {
256. **for** (**int** i = 0; i < jsonStrList.size(); i++) {
258. **if** (jsonStrList.get(i).equals("[]") || jsonStrList.get(i).equals("{}") || jsonStrList.get(i).length() <= 0)
259. **continue**;
261. String date = dateList.get(i);
263. JSONObject firstLvl = **new** JSONObject(jsonStrList.get(i));
265. JSONArray array = (JSONArray) firstLvl.get("exchangeRate");
266. **for** (**int** j = 0; j < array.length(); j++) {
267. **if** (rate.equals("EUR")) {
268. **if** ((((JSONObject) array.get(j)).getString("currency").toString()).equals("EUR")) {
270. rateObj = **new** Rates(bank, ((JSONObject) array.get(j)).getString("currency").toString(),
271. DateLogic.formatPbDate(date), ((JSONObject) array.get(j)).getDouble(getBuy),
272. ((JSONObject) array.get(j)).getDouble(getSale));
273. listRates.add(rateObj);
274. **if** (DatabaseClass.getRate(rateObj).size() > 0) {
275. listRates.addAll(DatabaseClass.getRate(rateObj));
276. } **else** {
277. listRates.add(rateObj);
278. DatabaseClass.insertSingleToRates(rateObj);
279. }
280. }
281. } **else** **if** (rate.equals("USD")) {
282. **if** ((((JSONObject) array.get(j)).getString("currency").toString()).equals("USD")) {
283. rateObj = **new** Rates(bank, ((JSONObject) array.get(j)).getString("currency").toString(),
284. DateLogic.formatPbDate(date), ((JSONObject) array.get(j)).getDouble(getBuy),
285. ((JSONObject) array.get(j)).getDouble(getSale));
286. listRates.add(rateObj);
287. **if** (DatabaseClass.getRate(rateObj).size() > 0) {
288. listRates.addAll(DatabaseClass.getRate(rateObj));
289. } **else** {
290. listRates.add(rateObj);
291. DatabaseClass.insertSingleToRates(rateObj);
292. }
293. }
294. } **else** **if** (rate.equals("RUB")) {
295. **if** ((((JSONObject) array.get(j)).getString("currency").toString()).equals("RUB")) {
296. rateObj = **new** Rates(bank, ((JSONObject) array.get(j)).getString("currency").toString(),
297. DateLogic.formatPbDate(date), ((JSONObject) array.get(j)).getDouble(getBuy),
298. ((JSONObject) array.get(j)).getDouble(getSale));
299. listRates.add(rateObj);
300. **if** (DatabaseClass.getRate(rateObj).size() > 0) {
301. listRates.addAll(DatabaseClass.getRate(rateObj));
302. } **else** {
303. listRates.add(rateObj);
304. DatabaseClass.insertSingleToRates(rateObj);
305. }
306. }
307. }
308. }
309. }
310. }
312. }

Файл CurrencyResource.java

1. **package** org.flush.erates.resources;
3. **import** java.io.IOException;
4. **import** java.util.List;
6. **import** javax.ws.rs.Consumes;
7. **import** javax.ws.rs.FormParam;
8. **import** javax.ws.rs.POST;
9. **import** javax.ws.rs.Path;
10. **import** javax.ws.rs.Produces;
11. **import** javax.ws.rs.core.MediaType;
13. **import** org.flush.erates.date.DateLogic;
14. **import** org.flush.erates.dto.Rates;
15. **import** org.flush.erates.service.CurrencyService;
17. @Path("/currencies")
18. @Consumes(MediaType.APPLICATION\_JSON)
19. @Produces(MediaType.APPLICATION\_JSON)
20. **public** **class** CurrencyResouce {
21. /\* Listener for unique pb date \*/
22. @POST
23. @Path("/buildpbtable")
24. @Consumes(MediaType.APPLICATION\_FORM\_URLENCODED)
25. @Produces(MediaType.APPLICATION\_JSON)
26. **public** List<Rates> getListPBFromDate(@FormParam("specificDate") String date) {
27. date = ((date.length() > 0) && (date != **null**))?
28. DateLogic.formatPbDate(date):
29. DateLogic.formatPbDate(DateLogic.getTodayDate());
30. **return** CurrencyService.getAllCurrenciesFromPb(date);
31. }
32. /\* Listener for unique nbu date \*/
33. @POST
34. @Path("/buildnbutable")
35. @Consumes(MediaType.APPLICATION\_FORM\_URLENCODED)
36. @Produces(MediaType.APPLICATION\_JSON)
37. **public** List<Rates> getListNBUFromDate(@FormParam("specificDate") String date)**throws**  IOException {
38. date = ((date.length() > 0) && (date != **null**))?
39. date: DateLogic.getTodayDate();
40. **return** CurrencyService.getAllCurrenciesFromNbu(date);
41. }
42. /\* Listener for diapason of dates in nbu \*/
43. @POST
44. @Path("/diapasonnbutable")
45. @Consumes(MediaType.APPLICATION\_FORM\_URLENCODED)
46. @Produces(MediaType.APPLICATION\_JSON)
47. **public** List<Rates> getListNBUFromDiapason(@FormParam("startDate") String startDate,
48. @FormParam("endDate") String endDate) {
49. **return** CurrencyService.getAllDiapasonCurrenciesFromNbu(startDate, endDate);
50. }
51. /\* Listener for diapason of dates in pb \*/
52. @POST
53. @Path("/diapasonpbtable")
54. @Consumes(MediaType.APPLICATION\_FORM\_URLENCODED)
55. @Produces(MediaType.APPLICATION\_JSON)
56. **public** List<Rates> getListPBFromDiapason(@FormParam("startDate") String startDate,
57. @FormParam("endDate") String endDate) {
58. **return** CurrencyService.getCurrencyPBFromDiapason(startDate, endDate);
59. }
60. /\* Listenner for nbu diapasone \*/
61. @POST
62. @Path("/nbuonebank")
63. @Consumes(MediaType.APPLICATION\_FORM\_URLENCODED)
64. @Produces(MediaType.APPLICATION\_JSON)
65. **public** List<Rates>getNbuOneBankList(@FormParam("chooseRate") String selectedRate,
66. @FormParam("oneBankStartDate")String startDate,
67. @FormParam("oneBankEndDate") String endDate) {
68. **return** CurrencyService.getRateFromNbu(selectedRate, startDate, endDate);
69. }
70. /\* Listener for pb diapasone \*/
71. @POST
72. @Path("/pbonebank")
73. @Consumes(MediaType.APPLICATION\_FORM\_URLENCODED)
74. @Produces(MediaType.APPLICATION\_JSON)
75. **public** List<Rates> getPbOneBankList(@FormParam("chooseRate") String selectedRate,
76. @FormParam("oneBankStartDate")String startDate,
77. @FormParam("oneBankEndDate") String endDate) {
78. **return** CurrencyService.getRateFromPb(selectedRate, startDate, endDate);
79. }
80. }

Файл CurrencyService.java

1. **package** org.flush.erates.service;
3. **import** java.io.IOException;
4. **import** java.net.HttpURLConnection;
5. **import** java.time.DayOfWeek;
6. **import** java.time.LocalDate;
7. **import** java.util.ArrayList;
8. **import** java.util.List;
10. **import** javax.ws.rs.WebApplicationException;
12. **import** org.flush.erates.database.DatabaseClass;
13. **import** org.flush.erates.date.DateLogic;
14. **import** org.flush.erates.dto.Rates;
15. **import** org.flush.erates.errormessages.ErrorMessages;
16. **import** org.flush.erates.parsing.Parser;
17. **import** org.flush.erates.container.UrlContainer;
19. **public** **class** CurrencyService {
21. **private** **static** Parser parser = **new** Parser();
23. **private** **static** List<Rates> rates;
24. **private** **static** List<String> daysList;
25. **private** **static** List<String> jsonStringList;
27. /\* Service method for all currencies from pb \*/
28. **public** **static** List<Rates> getAllCurrenciesFromPb(String date) {
29. rates = **new** ArrayList<>();
30. rates = DatabaseClass.getNeededDateList(date, "PrivatBank");
32. **if** (rates.size() != 0) {
33. **return** rates;
34. } **else** **if** (rates.size() == 0) {
35. **if** (!(date.equals(DateLogic.formatPbDate(DateLogic.getTodayDate())))) {
36. HttpURLConnection connectionObj = parser.openConnection(UrlContainer.PB\_NEEDED\_DATE\_URL, date);
37. String jsonStr = parser.getJSONString(connectionObj);
38. rates = parser.parseSpecificDatePB(jsonStr, date, "PrivatBank");
39. DatabaseClass.insertToRates(rates);
40. **return** rates;
41. } **else** **if** (date.equals(DateLogic.formatPbDate(DateLogic.getTodayDate()))) {
42. HttpURLConnection connectionObj = parser.openConnection(UrlContainer.PB\_TODAY\_URL, "");
43. String jsonStr = parser.getJSONString(connectionObj);
44. rates = parser.parseTodayPB(jsonStr, "PrivatBank");
45. DatabaseClass.insertToRates(rates);
46. **return** rates;
47. }
48. }
49. **return** rates;
50. }
52. /\* Service method for all currencies in nbu \*/
53. **public** **static** List<Rates> getAllCurrenciesFromNbu(String date) **throws** IOException {
54. rates = **new** ArrayList<>();
55. rates = DatabaseClass.getNeededDateList(DateLogic.formatPbDate(date), "NBU");
57. **if** (rates.size() != 0) {
58. **return** rates;
59. } **else** **if** (rates.size() == 0) {
60. **if** (!(DateLogic.formatPbDate(date).equals(DateLogic.formatPbDate(DateLogic.getTodayDate())))) {
61. HttpURLConnection connectionObj = parser.openConnection(UrlContainer.PB\_NEEDED\_DATE\_URL, DateLogic.formatPbDate(date));
62. String jsonStr = parser.getJSONString(connectionObj);
63. rates = parser.parseSpecificDatePB(jsonStr, DateLogic.formatPbDate(date), "NBU");
64. DatabaseClass.insertToRates(rates);
65. **return** rates;
66. } **else** **if** (DateLogic.formatPbDate(date).equals(DateLogic.formatPbDate(DateLogic.getTodayDate()))) {
67. HttpURLConnection connectionObj = parser.openConnection(UrlContainer.NBU\_TODAY\_URL, "");
68. String jsonStr = parser.getJSONString(connectionObj);
70. rates = parser.parseTodayPB(jsonStr, "NBU");
72. DatabaseClass.insertToRates(rates);
73. **return** rates;
74. }
75. }
76. **return** rates;
77. }
79. /\* Service for diapason in nbu \*/
80. **public** **static** List<Rates> getAllDiapasonCurrenciesFromNbu(String startDate, String endDate) {
81. rates = **new** ArrayList<>();
83. **if** (((startDate.length() <= 0) || (startDate == **null**)) || ((endDate.length() <= 0) || endDate == **null**)) {
84. **throw** **new** WebApplicationException(ErrorMessages.diapasonBadDatesException());
85. }
86. **if** (!Services.checkDiapason(startDate, endDate)) {
87. **throw** **new** WebApplicationException(ErrorMessages.diapasonDateBeforeException());
88. }
90. checkDayOfWeek(startDate, endDate, "PB");
91. rates.addAll(parser.getNbuDiapasonByDays(jsonStringList, daysList));
93. **return** rates;
94. }
96. /\* Service for diapason in pb \*/
97. **public** **static** List<Rates> getCurrencyPBFromDiapason(String startDate, String endDate) {
98. rates = **new** ArrayList<>();
99. **if** (((startDate.length() <= 0) || (startDate == **null**)) || ((endDate.length() <= 0) || endDate == **null**)) {
100. **throw** **new** WebApplicationException(ErrorMessages.diapasonBadDatesException());
101. }
102. **if** (!Services.checkDiapason(startDate, endDate)) {
103. **throw** **new** WebApplicationException(ErrorMessages.diapasonDateBeforeException());
104. }
105. checkDayOfWeek(startDate, endDate, "PB");
106. rates.addAll(parser.getPbDiapasonByDays(jsonStringList, daysList));
108. **return** rates;
109. }
111. /\* Service for unique rate \*/
112. **public** **static** List<Rates> getRateFromNbu(String selectedRate,
113. String startDate, String endDate) {
114. rates = **new** ArrayList<>();
116. **if** (((startDate.length() <= 0) || (startDate == **null**)) || ((endDate.length() <= 0) || endDate == **null**)) {
117. **throw** **new** WebApplicationException(ErrorMessages.diapasonBadDatesException());
118. }
119. **if** (!Services.checkDiapason(startDate, endDate)) {
120. **throw** **new** WebApplicationException(ErrorMessages.diapasonDateBeforeException());
121. }
123. checkDayOfWeek(startDate, endDate, "PB");
124. rates.addAll(parser.getNbuByRate(selectedRate, jsonStringList, daysList));
126. **return** rates;
127. }
129. /\* Service for unique rate from pb \*/
130. **public** **static** List<Rates> getRateFromPb(String selectedRate,
131. String startDate, String endDate) {
132. rates = **new** ArrayList<>();
134. **if** (((startDate.length() <= 0) || (startDate == **null**)) || ((endDate.length() <= 0) || endDate == **null**)) {
135. **throw** **new** WebApplicationException(ErrorMessages.diapasonBadDatesException());
136. }
137. **if** (!Services.checkDiapason(startDate, endDate)) {
138. **throw** **new** WebApplicationException(ErrorMessages.diapasonDateBeforeException());
139. }
141. checkDayOfWeek(startDate, endDate, "PB");
142. rates.addAll(parser.getPbByRate(selectedRate, jsonStringList, daysList));
144. **return** rates;
145. }
147. /\* Checking Saturday and Sunday \*/
148. **public** **static** **void** checkDayOfWeek(String startDate, String endDate, String bank) {
149. String start[] = startDate.split("-");
150. String end[] = endDate.split("-");
151. LocalDate startDateObj = LocalDate.of(Integer.valueOf(start[0]), Integer.valueOf(start[1]),
152. Integer.valueOf(start[2]));
153. LocalDate endDateObj = LocalDate.of(Integer.valueOf(end[0]), Integer.valueOf(end[1]), Integer.valueOf(end[2]));
155. daysList = **new** ArrayList<>();
156. jsonStringList = **new** ArrayList<>();
158. **for** (LocalDate date = startDateObj; date.isBefore(endDateObj.plusDays(1)); date = date.plusDays(1)) {
159. **if** (bank.equals("NBU")) {
160. **if** (date.getDayOfWeek() == DayOfWeek.SATURDAY || date.getDayOfWeek() == DayOfWeek.SUNDAY) {
161. **continue**;
162. } **else** {
163. daysList.add(date.toString());
165. HttpURLConnection connectionObj = parser.openConnection(UrlContainer.NBU\_NEEDED\_DATE\_URL, date.toString());
166. jsonStringList.add(parser.getJSONString(connectionObj));
167. }
168. } **else** **if** (bank.equals("PB")) {
169. daysList.add(date.toString());
170. HttpURLConnection connectionObj = parser.openConnection(UrlContainer.PB\_NEEDED\_DATE\_URL, DateLogic.formatPbDate(date.toString()));
171. jsonStringList.add(parser.getJSONString(connectionObj));
172. }
173. }
174. }
175. }

Файл конфігуріції hibernate.cfg.xml

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. <!DOCTYPE hibernate-configuration PUBLIC
3. "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
4. "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd"**>**
5. **<hibernate-configuration>**
6. **<session-factory>**
7. **<property** name="hibernate.connection.driver\_class"**>**com.mysql.jdbc.Driver**</property>**
8. **<property** name="hibernate.connection.password"**>**uQjM4lkrVn3j**</property>**
9. **<property** name="hibernate.connection.url"**>**jdbc:mysql://127.11.76.130:3306/exchangerates?useSSL=false**</property>**
10. **<property** name="hibernate.connection.username"**>**adminUJhnJpB**</property>**
11. **<property** name="hibernate.dialect"**>**org.hibernate.dialect.MySQLDialect**</property>**
12. **<property** name="hibernate.c3p0.min\_size"**>**5**</property>**
13. **<property** name="hibernate.c3p0.max\_size"**>**20**</property>**
14. **<property** name="hibernate.c3p0.timeout"**>**1000**</property>**
15. **<property** name="hibernate.c3p0.max\_statements"**>**50**</property>**
16. **<property** name="hibernate.show\_sql"**>**false**</property>**
17. **<property** name="hibernate.hbm2ddl.auto"**>**update**</property>**
18. **<mapping** class="org.flush.erates.dto.Rates" **/>**
19. **</session-factory>**
20. **</hibernate-configuration>**

Файл-дескриптор web.xml

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
3. **<web-app** version="3.1"
4. xmlns="http://xmlns.jcp.org/xml/ns/javaee"
5. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6. xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"
7. metadata-complete="false"**>**
9. **<display-name>**exrates**</display-name>**
10. **<welcome-file-list>**
11. **<welcome-file>**index.html**</welcome-file>**
12. **</welcome-file-list>**
14. <!-- Auto scan REST service -->
15. **<context-param>**
16. **<param-name>**resteasy.scan**</param-name>**
17. **<param-value>**true**</param-value>**
18. **</context-param>**
20. <!-- this need same with resteasy servlet url-pattern -->
21. **<context-param>**
22. **<param-name>**resteasy.servlet.mapping.prefix**</param-name>**
23. **<param-value>**/rest**</param-value>**
24. **</context-param>**
26. **<listener>**
27. **<listener-class>**
28. org.jboss.resteasy.plugins.server.servlet.ResteasyBootstrap
29. **</listener-class>**
30. **</listener>**
32. **<servlet>**
33. **<servlet-name>**resteasy-servlet**</servlet-name>**
34. **<servlet-class>**
35. org.jboss.resteasy.plugins.server.servlet.HttpServletDispatcher
36. **</servlet-class>**
37. **</servlet>**
39. **<servlet-mapping>**
40. **<servlet-name>**resteasy-servlet**</servlet-name>**
41. **<url-pattern>**/rest/\***</url-pattern>**
42. **</servlet-mapping>**
43. **</web-app>**