

Финальный проект - Егор

Ссылка на задачу <div>✓ JUS-234: Финальный проект <small>к выполнению</small></div>	
Статус страницы	В работе
Описание	Документация к сайту Bank Saint-Petersburg

 [Документация к сайту](#) [Bank Saint-Petersburg](#)

▼ Фичи, к которым идут требования

- 1. Форма авторизации
- 2. Форма подтверждения полем для ввода смс
- 3. Кнопка “Вернуться ко входу”

▼ Описание требований к фичам

1. Требования к полю логина:

- a. Должен быть от 4 до 20 символов включительно
- b. Логин может состоять из латиницы верхнего и нижнего регистра и цифирных значений 0-9
- c. Не должен содержать пробелов, кириллицу, заглавные латинские буквы, специальные символы.

2. Требования к полю пароля:

- a. Использовать только латинские буквы
- b. Использовать цифры
- c. Использовать специальные знаки: ! ? / @ & \$ %
- d. Использовать от 8 до 40 символов включительно

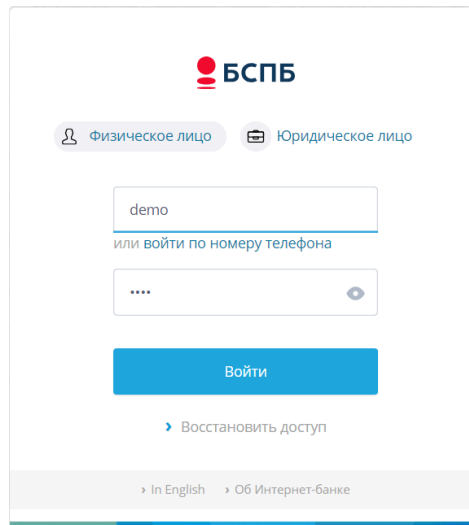
3. Требования к кнопке «Войти»

Кнопка должна быть активна только в том случае, если соблюдены условия к полям пароля и логина. Иначе, кнопка должна быть неактивна.

В активном состоянии кнопка «Войти» должна быть голубого цвета и доступна для нажатия с переходом на следующую страницу. В неактивном — серого цвета и недоступна для нажатия.

4. Требования к кнопке «Восстановить доступ»

Эта кнопка всегда должна быть активной. При её нажатии должно открываться окошко с контактной информацией банка.



Требования к странице ввода кода подтверждения из SMS

1. Код должен состоять ровно из 4 цифр.
2. В коде не должно быть букв и специальных символов.
3. Через 15 секунд после начала ввода должна появляться подсказка: «Если код не пришёл, проверьте правильность ввода пароля». Слова «ввода пароля» должны быть ссылкой на страницу авторизации.
4. Справа от поля для ввода кода должна находиться подсказка «Снова через 120 сек», которая ведёт отсчёт времени в обратном порядке. Когда время доходит до нуля, подсказка должна меняться на «Отправить снова».




Требования к кнопке «Подтвердить»

1. Кнопка «Подтвердить» должна быть активна — её можно нажать. Она должна быть голубого цвета.
2. Если пользователь введёт неверный код и нажмёт кнопку «Подтвердить», появится подсказка: «Неверный код».
3. При вводе верного кода и нажатии кнопки «Подтвердить» пользователь должен переходить на страницу личного кабинета.




Требования к кнопке «Вернуться к входу»

Кнопка «Вернуться к входу» должна быть активной. При её нажатии пользователь должен переходить на страницу авторизации.

▼ баг репорты

1.  JUS-637: (Егор Бар 1) Поехала верстка главной страницы для устройства Ipad air [готово](#)
2.  JUS-640: (Егор Бар 2) Страница не найдена при нажатии на персональные предложения с аккаунта "Королева Ольга" [готово](#)
3.  JUS-669: (Егор Бар 3) Не производится автоматический перевод валюты при конвертации. [готово](#)

▼ Тест-кейсы

1.  Создание запроса на перевод с карты на сумму меньше 100.pdf
2.  Блокировка карты Золотая.pdf
3.  Получение выписки по счету за текущий месяц.pdf

 Postman

▼ Коллекция

1 {

```

2  "info": {
3    "_postman_id": "9942505c-b612-4111-9660-c556afbe8977",
4    "name": "petstore",
5    "description": "# 🚀 Get started here\n\nThis template guides you through CRUD operations (GET, POST, PUT, DELETE)\n\nThis is a Postman collection for a petstore API. It contains a single endpoint for creating a new account. The endpoint is /api/users and it accepts a JSON body with the following fields: id, username, firstName, and lastName. The response is a JSON object with the same fields as the request body, plus an additional field called code. The code field represents the status of the account creation. A code of 200 indicates success, while any other code indicates an error. The response also includes a message field with a human-readable description of the status. The collection includes a test script that verifies the response data length, content type, status code, response time, and the presence of the expected fields in the response schema. The test script is written in JavaScript and uses the Postman test runner API (pm). The test script is located in the 'event' tab of the collection item. The collection is exported as a JSON file and can be imported into Postman. The collection is a good starting point for testing a petstore API. It can be extended with more endpoints and tests as needed. The collection is a valuable resource for anyone looking to test a petstore API.
6    "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",
7    "_exporter_id": "35315705"
8  },
9  "item": [
10    {
11      "name": "Create account",
12      "event": [
13        {
14          "listen": "test",
15          "script": {
16            "exec": [
17              "// Verify the length of the response data\r",
18              "pm.test(\"The response data length must be greater than zero\", function () {\r",
19                "    const responseData = pm.response.json();\r",
20                "    pm.expect(Object.keys(responseData).length).to.be.greaterThan(0, \"Response data length must be greater than zero\");\r",
21              "});\r",
22              "// Verify the content type of the response\r",
23              "pm.test(\"Response content type is application/json\", function () {\r",
24                "    pm.expect(pm.response.headers.get(\"Content-Type\")).to.include(\"application/json\");\r",
25              "});\r",
26              "\r",
27              "pm.test(\"Response status code is 200\", function () {\r",
28                "    pm.expect(pm.response.code).to.equal(200);\r",
29              "});\r",
30              "\r",
31              "\r",
32              "pm.test(\"Response time is less than 500ms\", function () {\r",
33                "    pm.expect(pm.response.responseTime).to.be.below(500);\r",
34              "});\r",
35              "\r",
36              "\r",
37              "pm.test(\"Presence of code, type, and message fields in response schema\", function () {\r",
38                "    const responseData = pm.response.json();\r",
39                "    \r",
40                "    pm.expect(responseData).to.be.an('object');\r",
41                "    pm.expect(responseData).to.have.property('code');\r",
42                "    pm.expect(responseData).to.have.property('type');\r",
43                "    pm.expect(responseData).to.have.property('message');\r",
44              "});\r",
45              "\r",
46              ""
47            ],
48            "type": "text/javascript",
49            "packages": {}
50          }
51        ]
52      },
53      "request": {
54        "method": "POST",
55        "header": [],
56        "body": {
57          "mode": "raw",
58          "raw": "{\r\n  \"id\": 0,\r\n  \"username\": \"malaronz\",\r\n  \"firstName\": \"string\", \r\n  \"lastName\": \"string\"\r\n}",
59          "options": {}

```

```

60         "raw": {
61             "language": "json"
62         }
63     },
64 },
65     "url": {
66         "raw": "{{URL}}/user",
67         "host": [
68             "{{URL}}"
69         ],
70         "path": [
71             "user"
72         ]
73     },
74 },
75     "response": []
76 },
77 {
78     "name": "Login",
79     "event": [
80         {
81             "listen": "test",
82             "script": {
83                 "exec": [
84                     "// Test to check the response content type\r",
85                     "pm.test(\"Response has content type application/json\", function () {\r",
86                     "    pm.expect(pm.response).to.have.header(\"Content-Type\", \"application/json\");\r",
87                     "});\r",
88                     "\r",
89                     "pm.test(\"Response status code is 200\", function () {\r",
90                     "    pm.expect(pm.response.code).to.equal(200);\r",
91                     "});\r",
92                     "\r",
93                     "\r",
94                     "pm.test(\"Response time is less than 200ms\", function () {\r",
95                     "    pm.expect(pm.response.responseTime).to.be.below(200);\r",
96                     "});\r",
97                     "\r",
98                     "\r",
99                     "pm.test(\"Validate the response schema for the presence of properties 'code',",
100                     "    const responseData = pm.response.json();\r",
101                     "\r",
102                     "    pm.expect(responseData).to.be.an('object');\r",
103                     "    pm.expect(responseData).to.have.property('code');\r",
104                     "    pm.expect(responseData).to.have.property('type');\r",
105                     "    pm.expect(responseData).to.have.property('message');\r",
106                     "});\r",
107                     "\r",
108                     "\r",
109                     "pm.test(\"Response has content type application/json\", function () {\r",
110                     "    pm.expect(pm.response.headers.get(\"Content-Type\")).to.include(\"applicat",
111                     "});\r",
112                     "\r",
113                     "\r",
114                     "pm.test(\"Response has Content-Type header with value 'application/json'\", fu",
115                     "    pm.expect(pm.response.headers.get(\"Content-Type\")).to.equal(\"applicatio",
116                     "});\r",
117                     "\r",

```



```

176         "pm.test(\"Type is a non-empty string\", function () {\r",
177         "     const responseData = pm.response.json();\r",
178         "     \r",
179         "     pm.expect(responseData).to.be.an('object');\r",
180         "     pm.expect(responseData.type).to.be.a('string').and.to.have.lengthOf.at.least",
181         "});\r",
182         "\r",
183         "\r",
184         "pm.test(\"Message is a non-empty string\", function () {\r",
185         "     const responseData = pm.response.json();\r",
186         "     \r",
187         "     pm.expect(responseData.message).to.be.a('string').and.to.have.lengthOf.at.le",
188         "});\r",
189         "\r",
190         ""
191     ],
192     "type": "text/javascript",
193     "packages": {}
194 }
195 }
196 ],
197 "request": {
198     "method": "PUT",
199     "header": [],
200     "body": {
201         "mode": "raw",
202         "raw": "\r\n{\r\n  \"id\": 0,\r\n  \"username\": \"egor2423\",\r\n  \"firstName\": \"Ale",
203         "options": {
204             "raw": {
205                 "language": "json"
206             }
207         }
208     },
209     "url": {
210         "raw": "{{URL}}/user/malaronz",
211         "host": [
212             "{{URL}}"
213         ],
214         "path": [
215             "user",
216             "malaronz"
217         ]
218     }
219 },
220 "response": []
221 },
222 {
223     "name": "delete account",
224     "event": [
225         {
226             "listen": "test",
227             "script": {
228                 "exec": [
229                     "// Test for the response content type\r",
230                     "pm.test(\"Response content type is application/json\", function () {\r",
231                     "     pm.expect(pm.response.headers.get(\"Content-Type\")).to.include(\"applicat",
232                     "});\r",
233                     "\r",

```

```

234         "pm.test(\"Response status code is 200\", function () {\r",
235         "     pm.expect(pm.response.code).to.equal(200);\r",
236         "});\r",
237         "\r",
238         "\r",
239         "pm.test(\"Response content type is application/json\", function () {\r",
240         "     pm.expect(pm.response.headers.get(\"Content-Type\")).to.include(\"applicat",
241         "});\r",
242         "\r",
243         "\r",
244         "pm.test(\"Response time is less than 200ms\", function () {\r",
245         "     pm.expect(pm.response.responseTime).to.be.below(200);\r",
246         "});\r",
247         "\r",
248         "\r",
249         "pm.test(\"Presence of fields - code, type, and message\", function () {\r",
250         "     const responseData = pm.response.json();\r",
251         "     \r",
252         "     pm.expect(responseData).to.be.an('object');\r",
253         "     pm.expect(responseData).to.have.property('code');\r",
254         "     pm.expect(responseData).to.have.property('type');\r",
255         "     pm.expect(responseData).to.have.property('message');\r",
256         "});\r",
257         "\r",
258         "\r",
259         "pm.test(\"Message is a non-empty string\", function () {\r",
260         "     const responseData = pm.response.json();\r",
261         "     \r",
262         "     pm.expect(responseData.message).to.be.a('string').and.to.have.lengthOf.at.lea",
263         "});\r",
264         "\r",
265         ""
266     ],
267     "type": "text/javascript",
268     "packages": {}
269 }
270 }
271 ],
272 "request": {
273     "method": "DELETE",
274     "header": [],
275     "url": {
276         "raw": "{{URL}}/user/malaronz",
277         "host": [
278             "{{URL}}"
279         ],
280         "path": [
281             "user",
282             "malaronz"
283         ]
284     }
285 },
286 "response": []
287 }
288 ],
289 "event": [
290     {
291         "listen": "prerequisite",

```

```

292     "script": {
293         "type": "text/javascript",
294         "exec": [
295             ""
296         ]
297     },
298 },
299 {
300     "listen": "test",
301     "script": {
302         "type": "text/javascript",
303         "exec": [
304             ""
305         ]
306     }
307 },
308 ],
309 "variable": [
310     {
311         "key": "id",
312         "value": "1"
313     },
314     {
315         "key": "base_url",
316         "value": "https://postman-rest-api-learner.glitch.me/"
317     }
318 ]
319 }

```

▼ Окружение

```

1  {
2      "id": "fb8aa890-8ad9-4e02-947e-08a02f83e645",
3      "name": "petstore",
4      "values": [
5          {
6              "key": "URL",
7              "value": "https://petstore.swagger.io/v2",
8              "type": "default",
9              "enabled": true
10         }
11     ],
12     "_postman_variable_scope": "environment",
13     "_postman_exported_at": "2024-09-05T18:20:33.746Z",
14     "_postman_exported_using": "Postman/11.10.0"
15 }

```

▼ Запуск коллекции из терминала

```

1  newman run petstore.postman_collection.json --environment env.json --insecure --reporters cli,htmlextra --re
2  newman: could not find "htmlextra" reporter
3      ensure that the reporter is installed in the same directory as newman
4      please install reporter using npm
5
6  newman
7
8  petstore
9
10 → Create account

```



```

11 POST https://petstore.swagger.io/v2/user [200 OK, 387B, 776ms]
12 ✓ The response data length must be greater than zero
13 ✓ Response content type is application/json
14 ✓ Response status code is 200
15 1. Response time is less than 500ms
16 ✓ Presence of code, type, and message fields in response schema
17
18 → Login
19 GET https://petstore.swagger.io/v2/user/login [200 OK, 471B, 146ms]
20 ✓ Response has content type application/json
21 ✓ Response status code is 200
22 ✓ Response time is less than 200ms
23 ✓ Validate the response schema for the presence of properties 'code', 'type', and 'message'
24 ✓ Response has content type application/json
25 ✓ Response has Content-Type header with value 'application/json'
26
27 → update account data
28 PUT https://petstore.swagger.io/v2/user/malaronz [200 OK, 387B, 148ms]
29 ✓ Response has content type application/json
30 ✓ Response status code is 200
31 ✓ Response body contains the required fields - code, type, and message
32 ✓ Code is a non-negative integer
33 ✓ Type is a non-empty string
34 ✓ Message is a non-empty string
35
36 → delete account
37 DELETE https://petstore.swagger.io/v2/user/malaronz [200 OK, 376B, 148ms]
38 ✓ Response content type is application/json
39 ✓ Response status code is 200
40 ✓ Response content type is application/json
41 ✓ Response time is less than 200ms
42 ✓ Presence of fields - code, type, and message
43 ✓ Message is a non-empty string
44

```

	executed	failed
iterations	1	0
requests	4	0
test-scripts	8	0
prerequisite-scripts	4	0
assertions	23	1
total run duration: 1771ms		
total data received: 250B (approx)		
average response time: 304ms [min: 146ms, max: 776ms, s.d.: 272ms]		

```

64
65 # failure          detail
66
67 1. AssertionError    Response time is less than 500ms
68                     expected 776 to be below 500

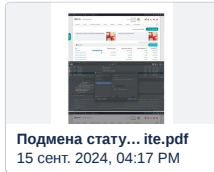
```

69
70

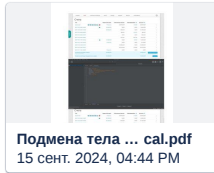
at assertion:3 in test-script
inside "Create account"

Charles

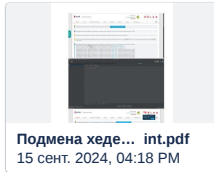
▼ Подмена статус-кода с помощью Rewrite



▼ Подмена тела ответа с помощью Map-Local



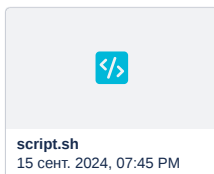
▼ Подмена хедера с помощью Breakpoint



Bash commands

▼ script

Скрипт запрашивает у пользователя количество создаваемых файлов, их название и путь для сохранения. Затем он создаёт файлы с последовательными номерами в указанном месте.



▼ commands

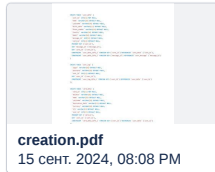
Перейти в домашнюю директорию: `cd ~`
Создать новую папку: `mkdir new_folder`
Перейти в нее: `cd new_folder`
Создать в ней новый файл: `touch my_file.txt`
Перейти в текстовый редактор: `vim my_file.txt`
Посмотреть конкретное словосочетание в файле: `grep 'словосочетание' my_file.txt`
Перенаправить поток в новый файл: `grep 'словосочетание' my_file.txt > new_file.txt`
Посмотреть, что находится в вашей директории: `ls`
Посмотреть, что находится в вашей директории, включая скрытые файлы: `ls -a`
Посмотреть, где вы находитесь: `pwd`
Создать 2 папки: `mkdir folder1 folder2`
Перейти в первую папку: `cd folder1`

Создать файл: touch file1.txt

Скопировать и перенести файл: cp file1.txt ../folder2/

Data base

Создание (команды)



Запуск mysql через docker

docker pull mysql #загрузка образа

docker run --name some-mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql #создание контейнера

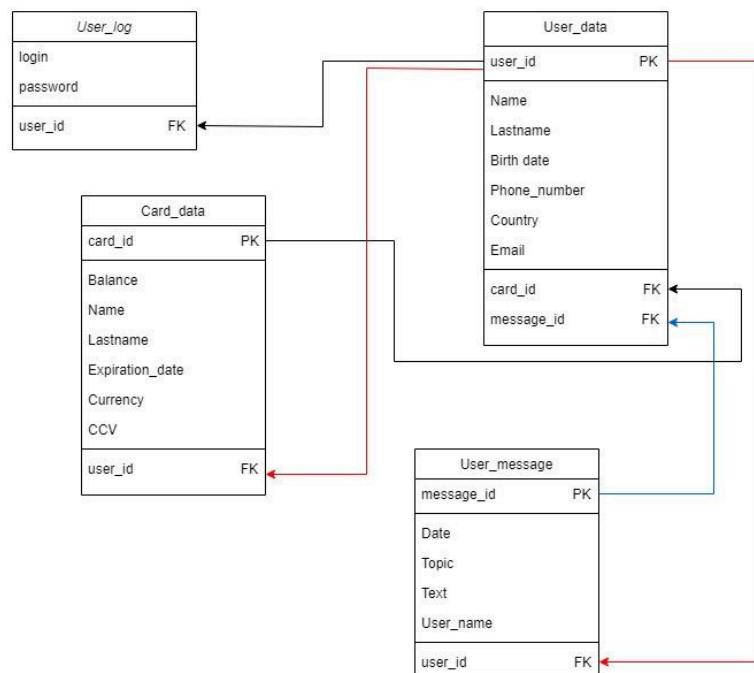
docker exec -it some-mysql bash #открытие оболочки mysql

docker ps #проверка статуса контейнера

docker stop some-mysql #остановка контейнера

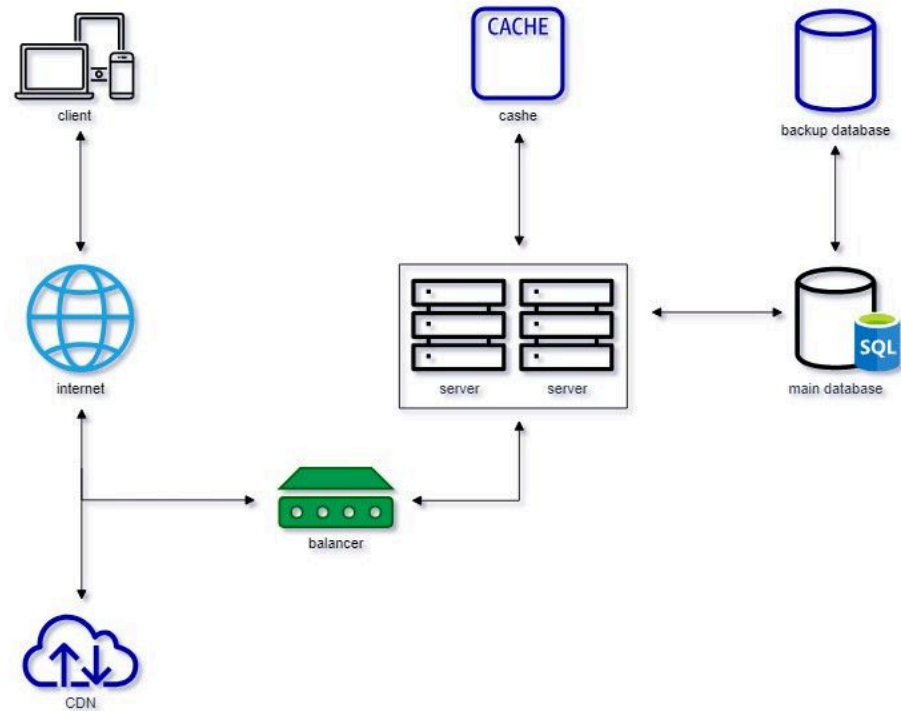
docker rm some-mysql #удаление контейнера

Таблицы



Клиент-серверная архитектура

▼ Клиент-серверная архитектура



JSON

▼ Пример кода JSON

```
1  [
2    {
3      "id": 1,
4      "name": "Иван Иванов",
5      "age": 30,
6      "hobby": "чтение книг",
7      "position": "программист",
8      "skills": [
9        "JavaScript",
10       "Python",
11       "C#"
12     ]
13   },
14   {
15     "id": 2,
16     "name": "Петр Петров",
17     "age": 40,
18     "hobby": "спорт",
19     "position": "менеджер проектов",
20     "team": [
21       {
22         "id": 1,
23         "name": "Василий Васильев",
```

```
24     "age": 25,  
25     "hobby": "фотография",  
26     "position": "разработчик"  
27 },  
28 {  
29     "id": 2,  
30     "name": "Сергей Сергеев",  
31     "age": 35,  
32     "hobby": "рыбалка",  
33     "position": "тестировщик"  
34 }  
35 ]  
36 }  
37 ]
```