

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ
НАПРАВЛЕНИЕ СИСТЕМНОГО И ПРИКЛАДНОГО ПРОГРАММНОГО
ОБЕСПЕЧЕНИЯ

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3
курса «Низкоуровневое программирование»
по теме: «Клиент-серверное взаимодействие»
Вариант № 2

Выполнил студент:
Тюрин Иван Николаевич
группа: Р33102

Преподаватель:
Кореньков Ю. Д.

Санкт-Петербург, 2024 г.

Содержание

Лабораторная работа № 3. Клиент-серверное взаимодействие	2
1. Задание варианта № 2	2
2. Выполнение задания	3
3. Описание работы	3
1. JSON схема запроса	4
2. JSON схема ответа	10
4. Результат работы приложения	11
5. Вывод	12

Лабораторная работа № 3

Клиент-серверное взаимодействие

1. Задание варианта № 2

На базе данного транспортного формата описать схему протокола обмена информацией и воспользоваться существующей библиотекой по выбору для реализации модуля, обеспечивающего его функционирование. Протокол должен включать представление информации о командах создания, выборки, модификации и удаления данных в соответствии с данной формой, и результатах их выполнения.

Используя созданные в результате выполнения заданий модули, разработать в виде консольного приложения две программы: клиентскую и серверную части. Серверная часть – получающая по сети запросы и операции описанного формата и последовательно выполняющая их над файлом данных с помощью модуля из первого задания. Имя файла данных для работы получать с аргументами командной строки, создавать новый в случае его отсутствия. Клиентская часть – в цикле получающая на стандартный ввод текст команд, извлекающая из него информацию о запрашиваемой операции с помощью модуля из второго задания и пересылающая её на сервер с помощью модуля для обмена информацией, получающая ответ и выводящая его в человеко-понятном виде в стандартный вывод.

Формат транспортного протокола: JSON.

2. Выполнение задания

В процессе работы были выполнены следующие ключевые задачи:

1. Спроектирован формат запросов и ответов
2. Спроектированы JSON схемы для представления запросов и ответов
3. В результате длительного поиска и изучения информации. Выбрана библиотека C++ `nlohmann/json` как наиболее удобная для выполнения поставленной задачи.
4. Для кроссплатформенной работы разработана обертка над API сокетов в системах Windows и *Nix (TcpSocket).
5. Созданы реализации (при помощи препроцессора выбирается какой .cpp файл использовать) для Windows и *Nix
6. Созданы и реализованы классы `Server` и `Client` отвечающие соответственно за работу серверного и клиентского приложений.
7. Клиентское и серверное приложения используют модули для хранения данных и синтаксического разбора запроса, разработанные в предыдущих лабораторных работах.

3. Описание работы

Проект состоит из следующих модулей, опуская модули из предыдущих лабораторных работ:

- **client** - содержит исходный код клиентского приложения, который делает синтаксический разбор запросов XPath-подобного синтаксиса в AST, переводит AST в JSON, валидирует полученный запрос, отправляет его серверу, ожидает результат в ответ в формате JSON. Полученный ответ, он десериализует в объект порции данных для дальнейшей работы с ним, например, вывода на экран.
- **server** - содержит исходный код серверного приложения, которое принимает поддерживает соединение, принимает запросы, которые валидирует разработанной JSON схемой запроса, и передает на дальнейшую обработку классу `StorageAdapter`, который выполняет соответствующие запросы с хранилищем и возвращает результат в виде JSON обратно. Полученный результат сервер отправляет обратно клиенту по сети в соответствии со схемой ответа.

- **common** - содержит общие для клиента и сервера заголовки структур данных и класс TcpSocket - обертку на Socket API, и платформенно-зависимые реализации.

3. 1. JSON схема запроса

```

1 {
2   "$schema": "http://json-schema.org/draft-07/schema#",
3   "$id": "./query_schema.json",
4   "title": "Query",
5   "description": "Query json representation",
6   "definitions": {
7     "element": {
8       "type": "object",
9       "properties": {
10        "elementType": {
11          "type": "string"
12        },
13        "attributes": {
14          "type": "array",
15          "items": {
16            "oneOf": [
17              {
18                "type": "string"
19              },
20              {
21                "type": "number"
22              },
23              {
24                "type": "boolean"
25              }
26            ]
27          }
28        }
29      },
30      "required": [
31        "elementType",
32        "attributes"
33      ],
34      "additionalProperties": false
35    },
36    "attributesPattern": {
37      "oneOf": [
38        {
39          "description": "binary operation",
40          "type": "object",
41          "properties": {
42            "patternType": {
43              "const": "BIN_OP"
44            },
45            "operation": {
46              "enum": [
47                "AND",
48                "OR"
49              ]
50            },
51            "lhs": {
52              "$ref": "#/definitions/attributesPattern"

```

```

53         },
54         "rhs": {
55             "$ref": "#/definitions/attributesPattern"
56         }
57     },
58     "required": [
59         "patternType",
60         "operation",
61         "lhs",
62         "rhs"
63     ],
64     "additionalItems": false
65 },
66 {
67     "description": "condition node",
68     "type": "object",
69     "properties": {
70         "patternType": {
71             "const": "COND"
72         },
73         "operation": {
74             "enum": [
75                 "EQ",
76                 "NEQ",
77                 "GT",
78                 "LT",
79                 "GTE",
80                 "LTE"
81             ]
82         },
83         "index": {
84             "type": "integer"
85         },
86         "value": {
87             "oneOf": [
88                 {
89                     "type": "number"
90                 },
91                 {
92                     "type": "string"
93                 }
94             ]
95         }
96     },
97     "required": [
98         "patternType",
99         "operation",
100         "index",
101         "value"
102     ],
103     "additionalItems": false
104 },
105 {
106     "description": "condition node",
107     "type": "object",
108     "properties": {
109         "patternType": {
110             "const": "COND"
111         },
112         "operation": {

```

```

113         "const": "CONTAINS"
114     },
115     "index": {
116         "type": "integer"
117     },
118     "value": {
119         "type": "string"
120     }
121 },
122 "required": [
123     "patternType",
124     "operation",
125     "index",
126     "value"
127 ],
128 "additionalItems": false
129 }
130 ]
131 },
132 "linkPattern": {
133     "type": "object",
134     "properties": {
135         "linkType": {
136             "type": "string"
137         },
138         "target": {
139             "$ref": "#/definitions/elementPattern"
140         }
141     },
142     "required": [
143         "linkType"
144     ],
145     "additionalProperties": false
146 },
147 "elementPattern": {
148     "type": "object",
149     "properties": {
150         "elementType": {
151             "type": "string"
152         },
153         "attributes": {
154             "$ref": "#/definitions/attributesPattern"
155         },
156         "linksIn": {
157             "type": "array",
158             "items": {
159                 "$ref": "#/definitions/linkPattern"
160             }
161         },
162         "linksOut": {
163             "type": "array",
164             "items": {
165                 "$ref": "#/definitions/linkPattern"
166             }
167         }
168     },
169     "required": [
170         "elementType",
171         "linksIn",
172         "linksOut"

```

```

173     ],
174     "additionalProperties": false
175   },
176 },
177 "type": "object",
178 "oneOf": [
179   {
180     "properties": {
181       "queryType": {
182         "const": "create_vertex"
183       },
184       "element": {
185         "$ref": "#/definitions/element"
186       }
187     },
188     "required": [
189       "queryType",
190       "element"
191     ],
192     "additionalProperties": false
193   },
194   {
195     "properties": {
196       "queryType": {
197         "enum": [
198           "drop_vertex",
199           "match"
200         ]
201       },
202       "pattern": {
203         "$ref": "#/definitions/elementPattern"
204       }
205     },
206     "required": [
207       "queryType",
208       "pattern"
209     ],
210     "additionalProperties": false
211   },
212   {
213     "properties": {
214       "queryType": {
215         "const": "create_edge"
216       },
217       "linkType": {
218         "type": "string"
219       },
220       "sourcePattern": {
221         "$ref": "#/definitions/elementPattern"
222       },
223       "dstPattern": {
224         "$ref": "#/definitions/elementPattern"
225       }
226     },
227     "required": [
228       "queryType",
229       "linkType",
230       "sourcePattern",
231       "dstPattern"
232     ],

```



```

233     "additionalProperties": false
234 },
235 {
236     "properties": {
237         "queryType": {
238             "const": "update_vertex"
239         },
240         "pattern": {
241             "$ref": "#/definitions/elementPattern"
242         },
243         "attributes": {
244             "type": "array",
245             "items": {
246                 "type": "object",
247                 "properties": {
248                     "index": {
249                         "type": "integer"
250                     },
251                     "value": {
252                         "oneOf": [
253                             {
254                                 "type": "number"
255                             },
256                             {
257                                 "type": "string"
258                             },
259                             {
260                                 "type": "boolean"
261                             }
262                         ]
263                     }
264                 },
265                 "required": [
266                     "index",
267                     "value"
268                 ],
269                 "additionalProperties": false
270             }
271         }
272     },
273     "required": [
274         "queryType",
275         "pattern",
276         "attributes"
277     ],
278     "additionalProperties": false
279 },
280 {
281     "properties": {
282         "queryType": {
283             "const": "create_vertex_type"
284         },
285         "attributes": {
286             "type": "array",
287             "items": {
288                 "enum": [
289                     "INT32",
290                     "INT64",
291                     "DOUBLE",
292                     "BOOL",

```

```

293         "STRING"
294     ]
295 }
296 },
297 "elementType": {
298     "type": "string"
299 },
300 },
301 "required": [
302     "queryType",
303     "attributes",
304     "elementType"
305 ],
306 "additionalProperties": false
307 },
308 {
309     "properties": {
310         "queryType": {
311             "const": "create_edge_type"
312         },
313         "linkType": {
314             "type": "string"
315         }
316     },
317     "required": [
318         "queryType",
319         "linkType"
320     ],
321     "additionalProperties": false
322 },
323 {
324     "properties": {
325         "queryType": {
326             "const": "drop_vertex_type"
327         },
328         "elementType": {
329             "type": "string"
330         }
331     },
332     "required": [
333         "queryType",
334         "elementType"
335     ],
336     "additionalProperties": false
337 },
338 {
339     "properties": {
340         "queryType": {
341             "const": "drop_edge_type"
342         },
343         "linkType": {
344             "type": "string"
345         }
346     },
347     "required": [
348         "queryType",
349         "linkType"
350     ],
351     "additionalProperties": false
352 }

```

```
353 ]
354 }
```

Листинг 1.1: Схема запроса

3. 2. JSON схема ответа

```
1 {
2   "$schema": "http://json-schema.org/draft-07/schema#",
3   "$id": "./response_schema.json",
4   "title": "Response",
5   "description": "Response json representation",
6   "type": "object",
7   "properties": {
8     "error": {
9       "enum": [
10        0,
11        1
12      ]
13    },
14    "result": {
15      "type": "array",
16      "items": {
17        "type": "object",
18        "properties": {
19          "elementType": {
20            "type": "string"
21          },
22          "attributes": {
23            "type": "array",
24            "items": {
25              "oneOf": [
26                {
27                  "type": "object",
28                  "properties": {
29                    "attributeType": {
30                      "enum": [
31                        "int64",
32                        "int32",
33                        "bool"
34                      ]
35                    },
36                    "value": {
37                      "type": "integer"
38                    }
39                  },
40                  "additionalProperties": false,
41                  "required": [
42                    "attributeType",
43                    "value"
44                  ]
45                },
46                {
47                  "type": "object",
48                  "properties": {
49                    "attributeType": {
50                      "const": "double"
51                    },
```

```

52         "value": {
53             "type": "number"
54         }
55     },
56     "additionalProperties": false,
57     "required": [
58         "attributeType",
59         "value"
60     ]
61 },
62 {
63     "type": "object",
64     "properties": {
65         "attributeType": {
66             "const": "string"
67         },
68         "value": {
69             "type": "string"
70         }
71     },
72     "additionalProperties": false,
73     "required": [
74         "attributeType",
75         "value"
76     ]
77 }
78 ]
79 }
80 }
81 },
82 "required": [
83     "elementType",
84     "attributes"
85 ],
86 "additionalProperties": false
87 }
88 }
89 },
90 "required": [
91     "error"
92 ],
93 "additionalProperties": false
94 }

```

Листинг 1.2: Схема ответа

4. Результат работы приложения

Приложение было протестировано, пример сеанса работы можно видеть на листинге:

```

1 >db.insert({pole/vtoroe_pole=true})
2 Ok
3 >db.find({pole/vtoroe_pole})
4 Ok: true
5 >db.insert({pole/vtoroe_pole=false})
6 Ok

```

```

7 >db.find({pole/vtoroe_pole})
8 Ok: false
9 >db.insert({pole/tretye_pole="lol"})
10 Ok
11 >db.find({pole/tretye_pole})
12 Ok: "lol"
13 >db.find({pole/*})
14 Ok: Count elements: 2
15 vtoroe_pole=false
16 tretye_pole="lol"
17 >db.delete({pole/vtoroe_pole=true})
18 Fail: No such node
19 >db.delete({pole/vtoroe_pole=false})
20 Ok
21 >db.find({pole/*})
22 Ok: Count elements: 1

```

Листинг 1.3: Пример сеанса работы

На листинге видно *безсхемную* натуру базы данных: каждое значение, которое записывается в нее, является узлом в дереве, на уровне с узлами-объектами (как в JSON).

5. Вывод

В результате выполнения работы удалось разработать базу данных, имеющую в основе документное дерево в качестве формы данных. При этом в работе использовались ранее разработанные модули для хранения данных и синтаксического разбора текстовых запросов. Приложение определенно нуждается в дальнейшем улучшении для его повсеместного использования. Во время выполнения работы были получены полезные навыки разработки сложной программной системы, укреплены навыки программирования на языке C++ и навыки работы с различными операционными системами, в том числе работа с вводом-выводом через отображение файла в виртуальное адресное пространство.

Исходный код проекта размещен в личном репозитории на сайте GitHub <https://github.com/elTurin/itmo-llp/tree/lab-3>.