Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №3

по «Низкоуровневое программирование»

Вариант XML

Выполнил:

Студент группы P33301

Акимов Роман Иванович

Преподаватель:

Кореньков Юрий Дмитриевич

Санкт-Петербург

2023

Задание

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 фала 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.

1 Изучить выбранную библиотеку

a. Библиотека должна обеспечивать сериализацию и десериализацию с валидацией в соответствии со схемой

b. Предпочтителен выбор библиотек, поддерживающих кодогенерацию на основе схемы

c. Библиотека может поддерживать передачу данных посредством TCP соединения

- Иначе, использовать сетевые сокеты посредством API ОС

d. Библиотека может обеспечивать диспетчеризацию удалённых вызовов

- Иначе, реализовать диспетчеризацию вызовов на основе информации о виде команды

2 На основе существующей библиотеки реализовать модуль, обеспечивающий взаимодействие

a. Описать схему протокола в поддерживаемом библиотекой формате

- Описание должно включать информацию о командах, их аргументах и результатах
- Схема может включать дополнительные сущности (например, для итератора)

b. Подключить библиотеку к проекту и сформировать публичный интерфейс модуля с использованием встроенных или сгенерированных структур данных используемой библиотеки

- Поддержать установление соединения, отправку команд и получение их результатов
- Поддержать приём входящих соединений, приём команд и отправку их результатов

c. Реализовать публичный интерфейс посредством библиотеки в соответствии с п1

3 Реализовать серверную часть в виде консольного приложения

a. В качестве аргументов командной строки приложение принимает:

- Адрес локальной конечной точки для прослушивания входящих соединений
- Имя файла данных, который необходимо открыть, если он существует, иначе создать

b. Работает с файлом данных посредством модуля из задания 1

c. Принимает входящие соединения и взаимодействует с клиентами посредством модуля из п2

d. Поступающая информация о запрашиваемых операциях преобразуется из структур данных модуля взаимодействия к структурам данных модуля управления данными и наоборот

4 Реализовать клиентскую часть в виде консольного приложения

a. В качестве аргументов командной строки приложение принимает адрес конечной точки для подключения

b. Подключается к серверу и взаимодействует с ним посредством модуля из п2

c. Читает со стандартного ввода текст команд и анализирует их посредством модуля из задания 2

d. Преобразует результат разбора команды к структурам данных модуля из п2, передаёт их для обработки на сервер, возвращаемые результаты выводит в стандартный поток вывода

5 Результаты тестирования представить в виде отчёта, в который включить:

d. В части 3 привести пример сеанса работы разработанных программ

e. В части 4 описать решение, реализованное в соответствии с пп.2-4

f. В часть 5 включить составленную схему п.2a

Ход работы

Клиент для передачи запроса по сети перед отправкой упаковывает его в формат XML. Библиотека libxml для упаковки. Сервер после получения запроса преобразует его в XML и на его основе выбирает нужную crud функцию, по итогам операции посылает ответ.

Пример работы программы

Добавление элемента

```
Enter query: create (c:Company {name:"ITMO", workers:100});

Message: Successfully created

Enter query: create (c:Company {name:"Blizzard", workers:150});

Message: Successfully created

Enter query: match (c:Company) return c;

Message: Successfully matched

NODE 1
        name: "ITMO"
        workers: 100

NODE 2
        name: "Blizzard"
        workers: 150
```

Обновление элемента

```
Enter query: match (c:Company) where c.name="ITMO" set c.workers=200;

Message: Successfully updated

Enter query: match (c:Company) return c;

Message: Successfully matched

NODE 1
        name: "ITMO"
        workers: 200

NODE 2
        name: "Blizzard"
        workers: 150
```

Поиск по условию

```
Enter query: match (c:Company) where c.workers=200 or c.workers=3 return c;

Message: Successfully matched

NODE 1
        name: "ITMO"
        workers: 200

NODE 2
        name: "Galera"
        workers: 3
```

Удаление элемента

```
Enter query: match (c:Company) return c;

Message: Successfully matched

NODE 1
        name: "ITMO"
        workers: 200

NODE 2
        name: "Blizzard"
        workers: 150

NODE 3
        name: "Galera"
        workers: 3

Enter query: match (c:Company) where c.name="Galera" delete c;

Message: Successfully deleted

Enter query: match (c:Company) return c;

Message: Successfully matched

NODE 1
        name: "ITMO"
        workers: 200

NODE 2
        name: "Blizzard"
        workers: 150
```

Аспекты реализации

Упаковка запроса происходит в документ формата XML.

Для валидации запросов от клиента и ответов от сервера в XML формате используются XSD-схемы. Парсинг и построение сообщений осуществляет библиотека libxml2.

query.xsd (запрос на сервер):

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:element name="IDENTIFIER" type="xs:string"/>
    <xs:element name="INTEGER" type="xs:integer"/>
    <xs:element name="STRING" type="xs:string"/>
    <xs:element name="FLOAT" type="xs:float"/>
    <xs:element name="BOOLEAN" type="xs:boolean"/>


    <xs:element name="EQ">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="IDENTIFIER"/>
                <xs:choice>
                    <xs:element ref="INTEGER"/>
                    <xs:element ref="STRING"/>
                    <xs:element ref="FLOAT"/>
                    <xs:element ref="BOOLEAN"/>
                </xs:choice>
            </xs:sequence>
        </xs:complexType>
    </xs:element>


    <xs:element name="GT">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="IDENTIFIER"/>
                <xs:choice>
                    <xs:element ref="INTEGER"/>
                    <xs:element ref="FLOAT"/>
                </xs:choice>
            </xs:sequence>
        </xs:complexType>
    </xs:element>


    <xs:element name="LT">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="IDENTIFIER"/>
                <xs:choice>
                    <xs:element ref="INTEGER"/>
                    <xs:element ref="FLOAT"/>
                </xs:choice>
```

```
            </xs:sequence>
        </xs:complexType>
    </xs:element>


    <xs:element name="NE">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="IDENTIFIER"/>
                <xs:choice>
                    <xs:element ref="INTEGER"/>
                    <xs:element ref="STRING"/>
                    <xs:element ref="FLOAT"/>
                    <xs:element ref="BOOLEAN"/>
                </xs:choice>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="AND">
        <xs:complexType>
            <xs:sequence>
                <xs:choice>
                    <xs:element ref="EQ"/>
                    <xs:element ref="GT"/>
                    <xs:element ref="LT"/>
                    <xs:element ref="NE"/>
                </xs:choice>
                <xs:choice>
                    <xs:element ref="EQ"/>
                    <xs:element ref="GT"/>
                    <xs:element ref="LT"/>
                    <xs:element ref="NE"/>
                    <xs:element ref="AND"/>
                    <xs:element ref="OR"/>
                </xs:choice>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="OR">
        <xs:complexType>
            <xs:sequence>
                <xs:choice>
                    <xs:element ref="EQ"/>
                    <xs:element ref="GT"/>
                    <xs:element ref="LT"/>
                    <xs:element ref="NE"/>
                </xs:choice>
                <xs:choice>
                    <xs:element ref="EQ"/>
                    <xs:element ref="GT"/>
                    <xs:element ref="LT"/>
                    <xs:element ref="NE"/>
                    <xs:element ref="AND"/>
                    <xs:element ref="OR"/>
                </xs:choice>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="WHERE">
        <xs:complexType>
            <xs:choice>
```

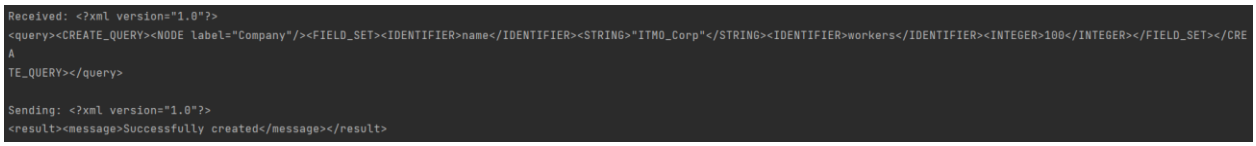
```
                <xs:element ref="AND"/>
                <xs:element ref="OR"/>
                <xs:element ref="EQ"/>
                <xs:element ref="GT"/>
                <xs:element ref="LT"/>
                <xs:element ref="NE"/>
            </xs:choice>
        </xs:complexType>
    </xs:element>

    <xs:element name="FIELD_SET">
        <xs:complexType>
            <xs:sequence maxOccurs="unbounded">
                <xs:element ref="IDENTIFIER"/>
                <xs:choice>
                    <xs:element ref="INTEGER"/>
                    <xs:element ref="STRING"/>
                    <xs:element ref="FLOAT"/>
                    <xs:element ref="BOOLEAN"/>
                </xs:choice>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="NODE">
        <xs:complexType>
            <xs:attribute name="label" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>

    <xs:element name="MATCH_QUERY">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="NODE"/>
                <xs:element ref="WHERE" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="CREATE_QUERY">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="NODE"/>
                <xs:element ref="FIELD_SET"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="DELETE_QUERY">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="NODE"/>
                <xs:element ref="WHERE" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="SET_QUERY">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="NODE"/>
                <xs:element ref="WHERE" minOccurs="0"/>
                <xs:element ref="FIELD_SET"/>
            </xs:sequence>
```

```
        </xs:complexType>
    </xs:element>

    <xs:element name="query">
        <xs:complexType>
            <xs:choice>
                <xs:element ref="MATCH_QUERY"/>
                <xs:element ref="CREATE_QUERY"/>
                <xs:element ref="DELETE_QUERY"/>
                <xs:element ref="SET_QUERY"/>
            </xs:choice>
        </xs:complexType>
    </xs:element>

</xs:schema>
```

response.xsd (ответ сервера):

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:element name="result" type="resultType"/>

    <xs:complexType name="resultType">
        <xs:sequence>
            <xs:element name="message" type="xs:string"/>
            <xs:element name="node" type="nodeType" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="nodeType">
        <xs:sequence maxOccurs="unbounded">
            <xs:element name="attr_name" type="xs:string"/>
            <xs:element name="attr_value" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>

</xs:schema>
```

Пример упакованных запросов




```
Received: <?xml version="1.0"?>
<query><MATCH_QUERY><NODE label="Company"/><WHERE><OR><EQ><IDENTIFIER>workers</IDENTIFIE
R><INTEGER>200</INTEGER></EQ><EQ><IDENTIFIER>workers</IDENTIFIER><INTEGER>3</INTEGER></E
Q></OR></WHERE></MATCH_QUERY></query>

Sending: <?xml version="1.0"?>
<result><message>Successfully matched</message><node><attr_name>name</attr_name><attr_va
lue>"ITMO"</attr_value><attr_name>workers</attr_name><attr_value>200</attr_value></node>
<node><attr_name>name</attr_name><attr_value>"Galera"</attr_value><attr_name>workers</at
tr_name><attr_value>3</attr_value></node></result>
```

```
Received: <?xml version="1.0"?>
<query><DELETE_QUERY><NODE label="Company"/><WHERE><EQ><IDENTIFIER>name</IDENTIFIER><STRING>"Galera"</STRING></EQ></WHERE></DELETE_QUERY></query>

Sending: <?xml version="1.0"?>
<result><message>Successfully deleted</message></result>
```

Пример работоспособности схемы:

клиент:

```
Enter query: create (c:Person);

Message: Invalid request
```

сервер:

```
Received: <?xml version="1.0"?>
<query><CREATE_QUERY><NODE label="Person"/><FIELD_SET/></CREATE_QUERY></query>

Entity: line 2: element FIELD_SET: Schemas validity error : Element 'FIELD_SET': Missing child element(s). Expected is ( IDENTIFIER ).
Invalid request
```

Передача по сети происходит через API ОС.

```c
int Socket(int domain, int type, int protocol) {
    int res = socket(domain, type, protocol);
    if (res == -1) {
        perror( s: "socket failed");
        exit( status: EXIT_FAILURE);
    }
    return res;
}

void Bind(int sock_fd, const struct sockaddr *addr, socklen_t addr_len) {
    int res = bind( fd: sock_fd, addr,  len: addr_len);
    if (res == -1) {
        perror( s: "bind failed");
        exit( status: EXIT_FAILURE);
    }
}

void Listen(int sock_fd, int back_log) {
    int res = listen( fd: sock_fd,  n: back_log);
    if (res == -1) {
        perror( s: "listen failed");
        exit( status: EXIT_FAILURE);
    }
}

int Accept(int sock_fd, struct sockaddr *addr, socklen_t *addr_len) {
    int res = accept( fd: sock_fd, addr, addr_len);
    if (res == -1) {
        perror( s: "accept failed");
        exit( status: EXIT_FAILURE);
    }
    return res;
}
```

```
void Connect(int sock_fd, const struct sockaddr *addr, socklen_t addr_len) {
    int res = connect( fd: sock_fd, addr,  len: addr_len);
    if (res == -1) {
        perror( s: "connect failed");
        exit( status: EXIT_FAILURE);
    }
}

void Inet_pton(int af, const char *src, void *dst) {
    int res = inet_pton(af,  cp: src,  buf: dst);
    if (res == 0) {
        printf( format: "inet error\n");
        exit( status: EXIT_FAILURE);
    }
    if (res == -1) {
        perror( s: "inet failed");
        exit( status: EXIT_FAILURE);
    }
}
```

Вывод:

Изучил библиотеку libxml, а также разработал клиент-серверное приложение с передачей данных по сети через API ОС.