МИНОБРНАУКИ РОССИИ

Федеральное государственное образовательное

учреждение высшего профессионального образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

**Кафедра Математического обеспечения и применения ЭВМ**

 

**ОТЧЕТ**

**по лабораторной работе №2**

на тему «Сервис-ориентированная архитектура.

Фреймворк Axis2»

по курсу «Архитектура и проектирование программных систем»

Выполнил:

студент группы КТбо4-12

Паньшин А.А.

Проверил:

Ассистент каф. МОП ЭВМ

Данилов И.Г.

Оценка

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2016г.

Таганрог 2016

Содержание

[Цель работы 3](#_Toc441524804)

[Задание 3](#_Toc441524805)

[Ход работы 4](#_Toc441524806)

[1.1 Описание форматов данных 4](#_Toc441524807)

[1.1.1 Описание сущностей 4](#_Toc441524808)

[1.1.2 Сущность «Subscribers» 4](#_Toc441524809)

[1.1.3 Сущность «Topics» 4](#_Toc441524810)

[1.1.4 Сущность «Messages» 4](#_Toc441524811)

[1.1.5 Сущность «Info» 4](#_Toc441524812)

[1.1.6 Описание последовательности взаимодействия 5](#_Toc441524813)

[1.2 Описание типов данных(.xsd) 5](#_Toc441524814)

[1.3 Описание веб-сервиса(.wsdl) 7](#_Toc441524815)

[1.4 Файлы сборки проекта 9](#_Toc441524816)

[1.5 Сборка проекта 12](#_Toc441524817)

[Выводы 13](#_Toc441524818)

[Листинг программы 14](#_Toc441524819)

# Цель работы

Создание и описание распределенной системы на основе сервис-ориентированной архитектуры с применением фреймворка Axis2/C, системы контроля версий и системы сборки проектов CMake.

# Задание

Необходимо разработать распределенную систему «Подписчики», основой которой является сервис Subscribers со следующими операциями:

* **Subscribe**. Входные параметры: тема подписки (topic). Выходной параметр: идентификатор подписки (id) и сообщение (message);
* **Unsubscribe**. Входные параметры: идентификатор подписки (id). Выходной параметр: сообщение (message);
* **GetStructure**. Входные параметры: отсутствуют. Выходной параметр: структура системы, в которой содержатся все подписчики сервера и подписчики подписчиков и т. д.; дополнительно в элементе структуры, соответствующему конкретному подписчику, должна содержаться информация о теме, на которую он подписан, и все события-сообщения, сгенерированные на эту тему функцией GenerateTopicEvent;
* **GenerateTopicEvent**. Входные параметры: тема подписки (topic) и строковое сообщение на данную тему. Выходной параметр: отсутствует;
* **GenerateUnsubscribeEvent**. Входные параметры: отсутствуют. Выходной параметр: сообщение. Данная операция должна вызывать операцию Unsubscribe, если сервис был подписан ранее к другому сервису. Операция Unsubscribe должна также вызываться в деструкторе сервиса (функция axis2\_svc\_skel\_Subscribers\_free).

Кроме самого сервиса необходимо разработать клиент (графический или консольный), который по выбору пользователя вызывает одну из трех операций сервиса: GetStructure, GenerateTopicEvent, GenerateUnsubscribeEvent. Необходимо предусмотреть задание IP-адреса и порта сервиса, чью операцию пользователь захочет вызвать (т. к. апробацию системы проще всего будет производить на одном локальном компьютере, то каждый сервер Axis2c-контейнер веб-сервисов будет запускаться с одинаковым IP – localhost, но на разных портах).

При разработке сервиса и клиента необходимо применять подход «**сверху-вниз**»: написание типов данных в .xsd-схеме, написание .wsdl, генерация файлов на языке C.

# Ход работы

## Описание форматов данных

### Описание сущностей

В системе нами были определены следующие сущности:

* Subscribers;
* Topics;
* Messages;
* Info.

### Сущность «Subscribers»

Определяет подписчика системы, структура сущности представлена в таблице 1.

Таблица 1 – Сущность «Subscribers»

|  |  |  |
| --- | --- | --- |
| Наименование поля | Тип данных | Назначение |
| Id | Bigint | Уникальный идентификатор |
| TopicId | Bigint | ID темы сообщения |
| About | Varchar(128) | Информация о подписчике |
| Address | Varchar(128) | Адрес подписчика |

### Сущность «Topics»

Определяет тему подписки пользователя сервиса и представлена в таблице 2.

Таблица 2 – Сущность «Topics»

|  |  |  |
| --- | --- | --- |
| Наименование поля | Тип данных | Назначение |
| Id | Bigint | Уникальный идентификатор |
| Name | Varchar(128) | Название темы |

### Сущность «Messages»

Определяет сообщение, для обмена информацией между подписчиками сервиса и представлена в таблице 3.

Таблица 3 – Сущность «Messages»

|  |  |  |
| --- | --- | --- |
| Наименование поля | Тип данных | Назначение |
| TopicId | Bigint | ID темы |
| Message | Varchar(128) | Текст сообщения |

### Сущность «Info»

Идентифицирует сервис в сети, структура сущности представлена в таблице 4.

Таблица 4 – Сущность «Info»

|  |  |  |
| --- | --- | --- |
| Наименование поля | Тип данных | Назначение |
| Id | Bigint | Название темы |
| Address | Varchar(128) | Адрес сервиса |

Вышеописанные сущности представляют собой базу данных, которая будет использоваться сервисом. В качестве программного обеспечения, реализующего работу с БД мы выбрали SQLite в силу ее легковесности и скорости выполнения запросов. К тому же, базы данных SQLite, используемые приложениями, являются локальными, не требуется запускать множество служб для того, чтобы с ними работать, как, например, в случае с Microsoft SQL Server.

### Описание последовательности взаимодействия

При запуске сервиса будет производиться автоматическая подписка его к уже работающему сервису для формирования иерархии сервисов. После этого к сервису будет подключаться клиент для генерации определенных событий для сервиса.

## Описание типов данных(.xsd)

Типы данных описываются в файле \*.xsd и имеют следующую структуру:

<?xml version="1.0" encoding="UTF-8"?>

<schema

targetNamespace="http://mopevm.ru/axis2/services/types"

xmlns='http://www.w3.org/2001/XMLSchema'

xmlns:types="http://mopevm.ru/axis2/services/types"

xmlns:tns="http://mopevm.ru/axis2/services/types"

xmlns:xsd='http://www.w3.org/2001/XMLSchema'

xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

<element name='topic'>

<complexType>

<sequence>

<element name="name" type="xsd:string" minOccures="1" maxOccures="1"/>

<element name="about" type="xsd:string" maxOccures="unbounded"/>

<element name="address" type="xsd:string" minOccures="1" maxOccures="1"/>

</sequence>

</complexType>

</element>

<complexType name='message'>

<attribute name="errorLevel" type="xsd:int" use="required"/>

<sequence>

<element name="about" type="xsd:string" maxOccures="unbounded"/>

</sequence>

</complexType>

<element name='subscribeResponse'>

<complexType>

<sequence>

<element name="id" type="xsd:int" minOccures="1" maxOccures="1"/>

<element name="msg" type="types:message" minOccures="0" maxOccures="unbounded"/>

</sequence>

</complexType>

</element>

<element name='generateTopicEventRequest'>

<complexType>

<sequence>

<element name="nameOfTopic" type="xsd:string" minOccures="1" maxOccures="1"/>

<element name="message" type="xsd:string" minOccures="1" maxOccures="unbounded"/>

</sequence>

</complexType>

</element>

<element name='generateUnsubscribeEventRequest'>

<complexType>

<sequence>

<element name="number" type="xsd:int" minOccures="1" maxOccures="1"/>

</sequence>

</complexType>

</element>

<element name='generateUnsubscribeEventResponse'>

<complexType>

<sequence>

<element name="message" type="xsd:string" minOccures="1" maxOccures="unbounded"/>

</sequence>

</complexType>

</element>

<element name='unsubscribeRequest'>

<complexType>

<sequence>

<element name="id" type="xsd:int" minOccures="1" maxOccures="1"/>

</sequence>

</complexType>

</element>

<element name='unsubscribeResponse'>

<complexType>

<sequence>

<element name="message" type="xsd:string" minOccures="1" maxOccures="unbounded"/>

</sequence>

</complexType>

</element>

<element name='subscriberRequest'>

<complexType>

<sequence>

<element name="number" type="xsd:int" minOccures="1" maxOccures="1"/>

</sequence>

</complexType>

</element>

<complexType name='subscriber'>

<sequence>

<element name="id" type="xsd:int" nillable="true" minOccures="1" maxOccures="1"/>

<element name="nameOfTopic" type="xsd:string" nillable="true" minOccures="1" maxOccures="1"/>

<element name="about" type="xsd:string" nillable="true" minOccures="1" maxOccures="1"/>

<element name="messages" type="xsd:string" nillable="true" minOccures="0" maxOccures="unbounded"/>

<element name="subscriber" type="types:subscriber" nillable="true" minOccures="0" maxOccures="1"/>

</sequence>

</complexType>

<element name='structure'>

<complexType>

<sequence>

<element name="subscribers" type="types:subscriber" nillable="true" minOccures="0" maxOccures="unbounded"/>

</sequence>

</complexType>

</element>

</schema>

## Описание веб-сервиса(.wsdl)

Каркас сервиса описан в файле \*.wsdl и имеет следующую структуру:

<?xml version="1.0" encoding="UTF-8"?>

<wsdl:definitions

targetNamespace="http://mopevm.ru/axis2/services/Subscribers"

xmlns:tns="http://mopevm.ru/axis2/services/Subscribers"

xmlns:types="http://mopevm.ru/axis2/services/types"

xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'

xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/08/addressing'

xmlns:xsd='http://www.w3.org/2001/XMLSchema'

xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

<wsdl:types>

<schema

targetNamespace="http://mopevm.ru/axis2/services/types"

xmlns='http://www.w3.org/2001/XMLSchema'>

<include schemaLocation="../schemas/types.xsd"/>

</schema>

</wsdl:types>

<!-- ############################################# MESSAGES ############################################# -->

<wsdl:message name="SubscribeMsg">

<wsdl:part element="types:topic" name="parameter" />

</wsdl:message>

<wsdl:message name="SubscribeResponseMsg">

<wsdl:part element="types:subscribeResponse" name="parameter" />

</wsdl:message>

<wsdl:message name="GenerateTopicEventMsg">

<wsdl:part element="types:generateTopicEventRequest" name="parameter" />

</wsdl:message>

<wsdl:message name="GenerateUnsubscribeEventRequestMsg">

<wsdl:part element="types:generateUnsubscribeEventRequest" name="parameter" />

</wsdl:message>

<wsdl:message name="GenerateUnsubscribeEventResponseMsg">

<wsdl:part element="types:generateUnsubscribeEventResponse" name="parameter" />

</wsdl:message>

<wsdl:message name="UnsubscribeMsg">

<wsdl:part element="types:unsubscribeRequest" name="parameter" />

</wsdl:message>

<wsdl:message name="UnsubscribeResponseMsg">

<wsdl:part element="types:unsubscribeResponse" name="parameter" />

</wsdl:message>

<wsdl:message name="GetStructureRequestMsg">

<wsdl:part element="types:subscriberRequest" name="parameter" />

</wsdl:message>

<wsdl:message name="GetStructureMsg">

<wsdl:part element="types:structure" name="parameter" />

</wsdl:message>

<!-- ############################################# portType ############################################# -->

<wsdl:portType name="SubscribersPortType">

<wsdl:operation name="Subscribe">

<wsdl:input message="tns:SubscribeMsg" name="SubscribersSubscribe"/>

<wsdl:output message="tns:SubscribeResponseMsg" name="SubscribersSubscribeResponse"/>

</wsdl:operation>

<wsdl:operation name="GenerateTopicEvent">

<wsdl:input message="tns:GenerateTopicEventMsg" name="SubscribersGenerateTopicEvent"/>

</wsdl:operation>

<wsdl:operation name="GenerateUnsubscribeEvent">

<wsdl:input message="tns:GenerateUnsubscribeEventRequestMsg" name="SubscribersGenerateUnsubscribeEventRequest"/>

<wsdl:output message="tns:GenerateUnsubscribeEventResponseMsg" name="SubscribersGenerateUnsubscribeEventResponse"/>

</wsdl:operation>

<wsdl:operation name="Unsubscribe">

<wsdl:input message="tns:UnsubscribeMsg" name="SubscribersUnsubscribe"/>

<wsdl:output message="tns:UnsubscribeResponseMsg" name="SubscribersUnsubscribeResponse"/>

</wsdl:operation>

<wsdl:operation name="GetStructure">

<wsdl:input message="tns:GetStructureRequestMsg" name="SubscribersGetStructureRequest"/>

<wsdl:output message="tns:GetStructureMsg" name="SubscribersGetStructure"/>

</wsdl:operation>

</wsdl:portType>

<!-- ############################################# binding ############################################# -->

<wsdl:binding name="SubscribersBinding" type="tns:SubscribersPortType">

<soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>

<wsdl:operation name="Subscribe">

<soap:operation soapAction="Subscribers#Subscribe"/>

<wsdl:input name="SubscribersSubscribe">

<soap:body use="literal"/>

</wsdl:input>

<wsdl:output name="SubscribersSubscribeResponse">

<soap:body use="literal"/>

</wsdl:output>

</wsdl:operation>

<wsdl:operation name="GenerateTopicEvent">

<soap:operation soapAction="Subscribers#GenerateTopicEvent"/>

<wsdl:input name="SubscribersGenerateTopicEvent">

<soap:body use="literal"/>

</wsdl:input>

</wsdl:operation>

<wsdl:operation name="GenerateUnsubscribeEvent">

<soap:operation soapAction="Subscribers#GenerateUnsubscribeEvent"/>

<wsdl:input name="SubscribersGenerateUnsubscribeEventRequest">

<soap:body use="literal"/>

</wsdl:input>

<wsdl:output name="SubscribersGenerateUnsubscribeEventResponse">

<soap:body use="literal"/>

</wsdl:output>

</wsdl:operation>

<wsdl:operation name="GetStructure">

<soap:operation soapAction="Subscribers#GetStructure"/>

<wsdl:input name="SubscribersGetStructureRequest">

<soap:body use="literal"/>

</wsdl:input>

<wsdl:output name="SubscribersGetStructure">

<soap:body use="literal"/>

</wsdl:output>

</wsdl:operation>

<wsdl:operation name="Unsubscribe">

<soap:operation soapAction="Subscribers#Unsubscribe"/>

<wsdl:input name="SubscribersUnsubscribe">

<soap:body use="literal"/>

</wsdl:input>

<wsdl:output name="SubscribersUnsubscribeResponse">

<soap:body use="literal"/>

</wsdl:output>

</wsdl:operation>

</wsdl:binding>

<!-- ############################################# service ############################################# -->

<wsdl:service name="Subscribers">

<wsdl:port binding="tns:SubscribersBinding" name="Subscribers\_SOAPport\_http">

<soap:address location="http://localhost:9090/axis2/services/Subscribers"/>

</wsdl:port>

</wsdl:service>

</wsdl:definitions>

## Файлы сборки проекта

Файл makelists сервиса

cmake\_minimum\_required (VERSION 2.6)

set (PROJECT

Subscribers)

project (${PROJECT})

file(GLOB HEADERS

"src/\*.h")

file(GLOB SOURCES

"src/\*.cpp"

"src/\*.c")

set (EXTERNAL\_LIBRARIES

axutil

axis2\_parser

axis2\_engine

axis2\_http\_sender

axis2\_http\_receiver

guththila)

if (NOT DEFINED ENV{AXIS2C\_HOME})

message (FATAL\_ERROR "AXIS2C\_HOME is not set")

endif (NOT DEFINED ENV{AXIS2C\_HOME})

if (NOT DEFINED ENV{AXIS2C\_INCLUDE})

message (FATAL\_ERROR "AXIS2C\_INCLUDE is not set")

endif (NOT DEFINED ENV{AXIS2C\_INCLUDE})

message("AXIS: $ENV{AXIS2C\_INCLUDE}")

include\_directories (src)

include\_directories ($ENV{AXIS2C\_INCLUDE})

include\_directories ($ENV{AXIS2C\_INCLUDE}/platforms)

link\_directories ($ENV{AXIS2C\_HOME}/lib)

if (MSVC)

add\_definitions (/W3)

set (EXTERNAL\_LIBRARIES ${EXTERNAL\_LIBRARIES} axiom)

elseif (CMAKE\_COMPILER\_IS\_GNUCXX)

set (EXTERNAL\_LIBRARIES ${EXTERNAL\_LIBRARIES} axis2\_axiom)

add\_definitions (-Wall -pedantic)

else ()

message ("Unknown compiler")

endif ()

source\_group ("Header Files" FILES ${HEADERS})

source\_group ("Source Files" FILES ${SOURCES})

add\_library (${PROJECT} SHARED ${HEADERS} ${SOURCES})

target\_link\_libraries (${PROJECT} ${EXTERNAL\_LIBRARIES})

install(TARGETS ${PROJECT} DESTINATION ${CMAKE\_INSTALL\_PREFIX}/services/Subscribers/)

install(FILES ${CMAKE\_CURRENT\_SOURCE\_DIR}/resources/services.xml ${CMAKE\_CURRENT\_SOURCE\_DIR}/resources/mcc.wsdl DESTINATION ${CMAKE\_INSTALL\_PREFIX}/services/Subscribers/)

Файл makelists клиента

cmake\_minimum\_required (VERSION 2.6)

set (PROJECT

Subscribers)

project (${PROJECT})

file(GLOB HEADERS

"src/\*.h")

file(GLOB SOURCES

"src/\*.cpp"

"src/\*.c")

set (EXTERNAL\_LIBRARIES

axutil

axis2\_parser

axis2\_engine

axis2\_http\_sender

axis2\_http\_receiver

guththila)

if (NOT DEFINED ENV{AXIS2C\_HOME})

message (FATAL\_ERROR "AXIS2C\_HOME is not set")

endif (NOT DEFINED ENV{AXIS2C\_HOME})

if (NOT DEFINED ENV{AXIS2C\_INCLUDE})

message (FATAL\_ERROR "AXIS2C\_INCLUDE is not set")

endif (NOT DEFINED ENV{AXIS2C\_INCLUDE})

message("AXIS: $ENV{AXIS2C\_INCLUDE}")

include\_directories (src)

include\_directories ($ENV{AXIS2C\_INCLUDE})

include\_directories ($ENV{AXIS2C\_INCLUDE}/platforms)

link\_directories ($ENV{AXIS2C\_HOME}/lib)

if (MSVC)

add\_definitions (/W3)

set (EXTERNAL\_LIBRARIES ${EXTERNAL\_LIBRARIES} axiom)

elseif (CMAKE\_COMPILER\_IS\_GNUCXX)

set (EXTERNAL\_LIBRARIES ${EXTERNAL\_LIBRARIES} axis2\_axiom)

add\_definitions (-Wall -pedantic)

else ()

message ("Unknown compiler")

endif ()

source\_group ("Header Files" FILES ${HEADERS})

source\_group ("Source Files" FILES ${SOURCES})

add\_executable(main ${HEADERS} ${SOURCES})

target\_link\_libraries (main ${EXTERNAL\_LIBRARIES})

## Сборка проекта

Поскольку при попытках собрать необходимые проекты в ОС windows x86 мы столкнулись с некоторыми трудностями (точнее, при построении некоторых подпроектов Staff’а), было принято решение собрать и запустить проект под linux. Для этого мы выбрали систему Debian 8.2.0. Jessie x86, установили ее в качестве виртуальной машины под управлением VirtualBox 5, сконфигурировали, установили необходимые утилиты и библиотеки.

Сборка и запуск проекта в Debian прошли успешно запуск клиента и двух сервисов.

В соответствии с заданием сервисы должны запускаться и подписываться друг на друга, если при этом уже запущен какой-то сервис, иначе сервис должен просто идентифицировать себя. Клиент после запуска был подписан на 9091-й порт, на котором был запущен первый сервис.

После запуска можно работать с клиентом. При завершении работы сервисов, они вначале отпишутся от тех, на кого подписались и затем уже полностью завершат свою работу.

# Выводы

В процессе выполнения 2-й лабораторной работы я познакомился с фреймворком Axis2/C, описал сервис “Подписчики” в виде WSDL файла, сгенерировал каркас приложения, после чего реализовал его на практике.

Реализация веб-службы в виде приложения, принимающего запросы и возвращающие результат в виде SOAP-конвертов, является очень гибким средством реализации кроссплатформенного приложения, так как язык разметки XML является универсальным средством общения между системами. Схемы XSD обеспечивают целостность пакетов.

# Листинг программы

Исходный код сервиса.

Файл “axis2\_skel\_Subscribers.c”

/\*\*

\* axis2\_skel\_Subscribers.c

\*

\* This file was auto-generated from WSDL for "Subscribers|http://mopevm.ru/axis2/services/Subscribers" service

\* by the Apache Axis2/C version: 1.6.2 Built on : Mar 14, 2013 (10:39:28 MSK)

\* axis2\_skel\_Subscribers Axis2/C skeleton for the axisService

\*/

#include "axis2\_skel\_Subscribers.h"

#include "sqlite3.h"

int GLOBAL\_ID = 0;

/\*\*

\* auto generated function definition signature

\* for "Subscribe|http://mopevm.ru/axis2/services/Subscribers" operation.

\* @param env environment ( mandatory)\* @param MessageContext the outmessage context

\* @param \_topic of the adb\_topic\_t\*

\*

\* @return adb\_subscribeResponse\_t\*

\*/

adb\_subscribeResponse\_t\* axis2\_skel\_Subscribers\_Subscribe(const axutil\_env\_t \*env , axis2\_msg\_ctx\_t \*msg\_ctx,

adb\_topic\_t\* \_topic )

{

// add subscriber with topic to DB

sqlite3 \*db;

sqlite3\_stmt \*stmt;

int rc;

if (SQLITE\_OK == sqlite3\_open("Subscribers.db", &db))

{

printf("Database opened\n");

}

else

{

printf("Error opening database\n");

}

char \*query = "insert into 'subscribers' ('id', 'topic', 'about', 'address') values(?, ?, ?, ?)";

sqlite3\_prepare(db, query, -1, &stmt, 0);

sqlite3\_bind\_int(stmt, 1, GLOBAL\_ID);

sqlite3\_bind\_text(stmt, 2, adb\_topic\_get\_name(\_topic, env), -1, SQLITE\_STATIC);

sqlite3\_bind\_text(stmt, 3, adb\_topic\_get\_about(\_topic, env), -1, SQLITE\_STATIC);

sqlite3\_bind\_text(stmt, 4, adb\_topic\_get\_address(\_topic, env), -1, SQLITE\_STATIC);

rc=sqlite3\_step(stmt);

sqlite3\_finalize(stmt);

sqlite3\_close(db);

// returnе id and message text to subcsriber

adb\_subscribeResponse\_t\* reply = adb\_subscribeResponse\_create(env);

adb\_subscribeResponse\_set\_id(reply, env, GLOBAL\_ID++);

adb\_subscribeResponse\_set\_msg(reply, env, adb\_message\_create\_with\_values(env, "Ok. I subscribe you", 0));

return reply;

}

/\*\*

\* auto generated function definition signature

\* for "GenerateTopicEvent|http://mopevm.ru/axis2/services/Subscribers" operation.

\* @param env environment ( mandatory)\* @param MessageContext the outmessage context

\* @param \_generateTopicEventRequest of the adb\_generateTopicEventRequest\_t\*

\*

\* @return

\*/

axis2\_status\_t axis2\_skel\_Subscribers\_GenerateTopicEvent(const axutil\_env\_t \*env , axis2\_msg\_ctx\_t \*msg\_ctx,

adb\_generateTopicEventRequest\_t\* \_generateTopicEventRequest )

{

sqlite3 \*db;

sqlite3\_stmt \*stmt;

int rc;

if (SQLITE\_OK == sqlite3\_open("Subscribers.db", &db))

{

printf("Database opened\n");

}

else

{

printf("Error opening database\n");

}

char \*query = "insert into 'messages' ('topic', 'message') values(?, ?)";

sqlite3\_prepare(db, query, -1, &stmt, 0);

sqlite3\_bind\_text(stmt, 1, adb\_generateTopicEventRequest\_get\_nameOfTopic(\_generateTopicEventRequest, env), -1, SQLITE\_STATIC);

sqlite3\_bind\_text(stmt, 2, adb\_generateTopicEventRequest\_get\_message(\_generateTopicEventRequest, env), -1, SQLITE\_STATIC);

rc=sqlite3\_step(stmt);

sqlite3\_finalize(stmt);

sqlite3\_close(db);

return AXIS2\_SUCCESS;

}

/\*\*

\* auto generated function definition signature

\* for "GenerateUnsubscribeEvent|http://mopevm.ru/axis2/services/Subscribers" operation.

\* @param env environment ( mandatory)\* @param MessageContext the outmessage context

\* @param \_generateUnsubscribeEventRequest of the adb\_generateUnsubscribeEventRequest\_t\*

\*

\* @return adb\_generateUnsubscribeEventResponse\_t\*

\*/

adb\_generateUnsubscribeEventResponse\_t\* axis2\_skel\_Subscribers\_GenerateUnsubscribeEvent(const axutil\_env\_t \*env , axis2\_msg\_ctx\_t \*msg\_ctx,

adb\_generateUnsubscribeEventRequest\_t\* \_generateUnsubscribeEventRequest )

{

axis2\_stub\_t \*stub = NULL;

const axis2\_char\_t \*address = NULL;

const axis2\_char\_t \*client\_home = NULL;

int id;

sqlite3 \*db;

sqlite3\_stmt \*stmt;

int rc;

// env = axutil\_env\_create\_all("subscribers.log", AXIS2\_LOG\_LEVEL\_TRACE);

client\_home = AXIS2\_GETENV("AXIS2C\_HOME");

if (!client\_home || !strcmp(client\_home, ""))

client\_home = "../..";

if (SQLITE\_OK == sqlite3\_open("Subscribers.db", &db))

{

printf("Database opened\n");

}

else

{

printf("Error opening database\n");

}

char\* query = "select \* from info";

sqlite3\_prepare(db, query, -1, &stmt, 0);

rc=sqlite3\_step(stmt);

id = sqlite3\_column\_int( stmt, 0 );

address = sqlite3\_column\_text( stmt, 1 );

sqlite3\_finalize(stmt);

sqlite3\_close(db);

stub = axis2\_stub\_create\_Subscribers(env, client\_home, address);

adb\_unsubscribeResponse\_t\* unsubscribeResponse = axis2\_stub\_op\_Subscribers\_Unsubscribe( stub, env,

adb\_unsubscribeRequest\_create\_with\_values(env, id));

adb\_generateUnsubscribeEventResponse\_t\* reply = adb\_generateUnsubscribeEventResponse\_create(env);

adb\_generateUnsubscribeEventResponse\_set\_message(reply, env, adb\_unsubscribeResponse\_get\_message(unsubscribeResponse, env));

return reply;

}

/\*\*

\* auto generated function definition signature

\* for "Unsubscribe|http://mopevm.ru/axis2/services/Subscribers" operation.

\* @param env environment ( mandatory)\* @param MessageContext the outmessage context

\* @param \_unsubscribeRequest of the adb\_unsubscribeRequest\_t\*

\*

\* @return adb\_unsubscribeResponse\_t\*

\*/

adb\_unsubscribeResponse\_t\* axis2\_skel\_Subscribers\_Unsubscribe(const axutil\_env\_t \*env , axis2\_msg\_ctx\_t \*msg\_ctx,

adb\_unsubscribeRequest\_t\* \_unsubscribeRequest )

{

sqlite3 \*db;

sqlite3\_stmt \*stmt;

int rc;

int id;

if (SQLITE\_OK == sqlite3\_open("Subscribers.db", &db))

{

printf("Database opened\n");

}

else

{

printf("Error opening database\n");

}

id = adb\_unsubscribeRequest\_get\_id(\_unsubscribeRequest, env);

char\* query = "delete from 'subscribers' where id = ?";

sqlite3\_prepare(db, query, -1, &stmt, 0);

sqlite3\_bind\_int(stmt, 1, id);

rc=sqlite3\_step(stmt);

sqlite3\_finalize(stmt);

sqlite3\_close(db);

adb\_unsubscribeResponse\_t\* reply = adb\_unsubscribeResponse\_create(env);

adb\_unsubscribeResponse\_set\_message(reply, env, "Ok. I unsubscribed you");

return reply;

}

adb\_structure\_t\* axis2\_skel\_Subscribers\_GetStructure(const axutil\_env\_t \*env , axis2\_msg\_ctx\_t \*msg\_ctx,

adb\_subscriberRequest\_t\* \_subscriberRequest )

{

axis2\_stub\_t \*stub = NULL;

const axis2\_char\_t \*address = NULL;

const axis2\_char\_t \*client\_home = NULL;

adb\_subscriberRequest\_t\* subscriberRequest;

sqlite3 \*db;

sqlite3\_stmt \*stmt1, \*stmt2, \*stmt3, \*stmt4;

int rc1, rc2, rc3, rc4;

char \*query1, \*query2, \*query3, \*query4;

axis2\_char\_t\* nameOfTopic;

axis2\_char\_t\* about;

axis2\_char\_t\*\* messages;

int id;

int i, j = 0;

int countOfMessages = 0;

int countOfSubscribers = 0;

int k =0;

adb\_structure\_t\* structure = adb\_structure\_create(env);

adb\_subscriber\_t\*\* subscribers = adb\_subscriber\_create(env);

adb\_subscriber\_t\* subscriber = adb\_subscriber\_create(env);

if (SQLITE\_OK == sqlite3\_open("Subscribers.db", &db))

{

printf("Database opened\n");

}

else

{

printf("Error opening database\n");

}

// subscribers count

query1 = "select count(\*) from subscribers";

sqlite3\_prepare(db, query1, -1, &stmt1, 0);

rc1=sqlite3\_step(stmt1);

countOfSubscribers = sqlite3\_column\_int( stmt1, 0 );

printf("%d\n", countOfSubscribers);

sqlite3\_finalize(stmt1);

// Allocate memory for subscribers’ pointers

subscribers = malloc(countOfSubscribers \* sizeof(adb\_subscriber\_t\*));

query1 = "select \* from subscribers";

sqlite3\_prepare(db, query1, -1, &stmt1, 0);

while(1)

{

rc1=sqlite3\_step(stmt1);

if (rc1 == SQLITE\_ROW)

{

id = sqlite3\_column\_int( stmt1, 0 );

nameOfTopic = sqlite3\_column\_text( stmt1, 1 );

about = sqlite3\_column\_text( stmt1, 2 );

printf("%d\n", id);

printf("%s\n", nameOfTopic);

printf("%s\n", about);

//rowcount

query3 = "select count(\*) from messages as t1 where t1.topic = ?";

sqlite3\_prepare(db, query3, -1, &stmt3, 0);

sqlite3\_bind\_text(stmt3, 1, nameOfTopic, -1, SQLITE\_STATIC);

rc3=sqlite3\_step(stmt3);

countOfMessages = sqlite3\_column\_int( stmt3, 0 );

sqlite3\_finalize(stmt3);

// Allocate memory for row pointers

printf("%d\n", countOfMessages);

messages = malloc(countOfMessages \* sizeof(axis2\_char\_t\*));

i = 0;

printf("YYYYYYYYYY");

query2 = "select t1.message from messages as t1 where t1.topic = ?";

sqlite3\_prepare(db, query2, -1, &stmt2, 0);

sqlite3\_bind\_text(stmt2, 1, nameOfTopic, -1, SQLITE\_STATIC);

while(1)

{

rc2=sqlite3\_step(stmt2);

if (rc2 == SQLITE\_ROW)

{

messages[i] = sqlite3\_column\_text( stmt2, 0 );

printf("%s\n", messages[i]);

i++;

}

else

break;

}

printf("YYYYYYYYYY");

sqlite3\_finalize(stmt2);

adb\_subscriber\_set\_id(subscriber, env, id);

adb\_subscriber\_set\_nameOfTopic(subscriber, env, nameOfTopic);

adb\_subscriber\_set\_about(subscriber, env, about);

printf("YYYYYYYYYY");

adb\_subscriber\_set\_messages(subscriber, env, messages);

subscribers[j] = adb\_subscriber\_create(env);

if (subscribers[j] == NULL)

printf("YES");

subscribers[j] = adb\_subscriber\_create(env);

subscribers[j] = subscriber;

free(messages);

}

else

break;

}

free(subscribers);

return structure;

}

Исходный код клиента.

Файл “subscribers\_client.c”

#include "axis2\_stub\_Subscribers.h"

#include <stdio.h>

#include <axiom.h>

#include <axis2\_util.h>

#include <axiom\_soap.h>

#include <axis2\_client.h>

void show(const axutil\_env\_t \*env, adb\_structure\_t\* structure);

int main(int argc, char\*\* argv)

{

axis2\_stub\_t \*stub = NULL;

axis2\_status\_t status = AXIS2\_FAILURE;

const axutil\_env\_t \*env = NULL;

const axis2\_char\_t \*address = NULL;

const axis2\_char\_t \*client\_home = NULL;

adb\_generateTopicEventRequest\_t\* generateTopicEventRequest;

adb\_generateUnsubscribeEventRequest\_t\* generateUnsubscribeEventRequest;

adb\_generateUnsubscribeEventResponse\_t\* generateUnsubscribeEventResponse;

adb\_subscriberRequest\_t\* subscriberRequest;

int operation = 0;

char nameOfTopic[255];

char message[255];

/\* Set up the environment \*/

env = axutil\_env\_create\_all("subscribers.log", AXIS2\_LOG\_LEVEL\_TRACE);

/\* Set up deploy folder. It is from the deploy folder, the configuration is picked up

\* using the axis2.xml file. You need to set the AXIS2C\_HOME variable to the axis2/c

\* installed dir.

\*/

client\_home = AXIS2\_GETENV("AXIS2C\_HOME");

if (!client\_home || !strcmp(client\_home, ""))

client\_home = "../..";

/\* Set end point reference of Subscribers service \*/

address = "http://localhost:9091/axis2/services/Subscribers";

stub = axis2\_stub\_create\_Subscribers(env, client\_home, address);

if (stub)

{

printf("Subscribed to : %s\n", address);

}

else

{

printf("You didn't subscribe to : %s\n", address);

}

generateTopicEventRequest = adb\_generateTopicEventRequest\_create(env);

subscriberRequest = adb\_subscriberRequest\_create(env);

do

{

printf("-------------------------------------------------------------------------------\n");

printf("1 - GetStructure\n");

printf("2 - GenerateTopicEvent\n");

printf("3 - GenerateUnsubscribeEvent\n");

printf("Please, input number of operation: ");

scanf("%d", &operation);

switch(operation)

{

case 1://GetStructure

adb\_subscriberRequest\_set\_number(subscriberRequest, env, 0);

adb\_structure\_t\* structure = axis2\_stub\_op\_Subscribers\_GetStructure(stub, env, subscriberRequest);

if (structure == NULL)

printf("subscribers is null!");

show(env, structure);

break;

case 2://GenerateTopicEvent

printf("Name: ");

gets(nameOfTopic);

printf("Message: ");

gets(message);

adb\_generateTopicEventRequest\_set\_nameOfTopic(generateTopicEventRequest, env, &nameOfTopic);

adb\_generateTopicEventRequest\_set\_message(generateTopicEventRequest, env, &message);

axis2\_stub\_op\_Subscribers\_GenerateTopicEvent(stub, env, generateTopicEventRequest);

break;

}

}while(operation != 3);

generateUnsubscribeEventRequest = adb\_generateUnsubscribeEventRequest\_create(env);

adb\_generateUnsubscribeEventRequest\_set\_number(generateUnsubscribeEventRequest, env, 0);

generateUnsubscribeEventResponse = axis2\_stub\_op\_Subscribers\_GenerateUnsubscribeEvent(stub, env, generateUnsubscribeEventRequest);

printf("%s\n", adb\_generateUnsubscribeEventResponse\_get\_message(generateUnsubscribeEventResponse, env));

return status;

}

void show(const axutil\_env\_t \*env, adb\_structure\_t\* structure)

{

printf("IN SHOW\n");

if (structure == NULL)

return;

int i = 0;

adb\_subscriber\_t\*\* subscribers = adb\_structure\_get\_subscribers(structure, env);

for(i = 0; i < 2; i++)

{

printf("%d", adb\_subscriber\_get\_id(subscribers, env));

}

}