Министерство Просвещения, Культуры и Исследования Республики Молдова

Технический Университет Молдовы

Факультет Вычислительной Техники, Информатики и Микроэлектроники

Департамент Инженерии Программного Обеспечения и Автоматики

Отчёт

по лабораторной работе №1

по дисциплине «Программирование распределенных приложений»

Тема: «Агент обмена сообщениями - *Message*

*Broker*»

Подготовил:cт. гр. TI-144 Куликов И.

Проверил*:* асист преп. Гавришко А.

Кишинёв 2017

# 1. Цель

Интегрирование, основанное на агентах обмена сообщениями, которое позволяет асинхронное общение между распределёнными компонентами системы.

# 2. Условия задания

1. Изучение агентов сообщений;
2. Разработка протокола связи агента обмена сообщениями;
3. Одновременное обращение с сообщениями;
4. Выбор транспортного протокола (в зависимости от цели / среды агента обмена сообщениями);
5. Выбор и разработка стратегии хранения сообщений;

**2.1 Оценочная шкала**

1. Реализация очереди сообщений (оценка 5);
2. Реализация механизма хранения сообщений (оценка 6);
3. Реализация механизма маршрутизации сообщений (оценка 7);
4. Внедрение шаблона издателя-подписчика (оценка 8);
5. Реализация расширенной маршрутизации сообщений (оценка 9);
6. Реализация механизма «last will and testament» (оценка 10).

# 3. Выполнение задания

Для реализации данной лабораторной работы был выбран язык программирования Python. В качестве брокера был запущен таск, который подключается к сокету и слушает его, при люблом евенте с данным сокетом вызвается метод обработчик.

task = asyncio.start\_server(handle\_message, hostname, port, loop=loop)

server = loop.run\_until\_complete(task)

print('Serving on ', server.sockets[0].getsockname())

try:

loop.run\_forever()

except KeyboardInterrupt:

pass

Данный обработчик определяет тип сообщения и в зависимости от того, чтение или запись посылает сообщение в очередь, или возвращает его запрашиваюшему .

if type == 'send':

if topic == '':

yield from \_MESSAGE\_QUEUE.put(payload)

elif topic in static\_topics.keys():

yield from static\_topics[topic].put(payload)

elif topic in dynamic\_topics.keys():

yield from dynamic\_topics[topic].put(payload)

else:

dynamic\_topics[topic] = asyncio.Queue(loop=asyncio.get\_event\_loop())

yield from dynamic\_topics[topic].put(payload)

msg = 'OK'

elif type == 'read':

if topic == '':

yield from \_MESSAGE\_QUEUE.get()

elif topic in static\_topics.keys():

msg = yield from static\_topics[topic].get()

elif topic in dynamic\_topics.keys():

msg = yield from dynamic\_topics[topic].get()

else:

print("Unhandled error")

msg = "Error!"

# Вывод

В результате проведённой лабораторной работы был изучен механизм под названием Агент Сообщений, который служит связывающей цепочкой между отправителем и получателем сообщений.

В данной лабораторной работе был изучен подход для создания централизованной системы по отправке сообщений. Были изучены шаблоны по проектированию модели отправки сообщения и некоторые из них были использованы на практике. Были применены уже имеющиеся знания по сериализации и десериализации данных, по использованию TCP соединений.

Данное задание было интересным для выполнения, так как отчётливо можно провести связь с реальными приложениями, которые построены при помощи похожей архитектуры и возможно используют похожие паттерн для отправки сообщений.

https://github.com/CulicovIgor/PAD\_Labs