Лабораторная работа 5.

Приложение akka streams, предназначенное для нагрузочного тестирования .

1. Задача

Требуется разработать приложение использующее технологию akka streams и позволяющее с помощью http запроса несколько одинаковых GET запросов и померять среднее время отклика.

Для запроса к серверу используем библиотеку async Http Client.

https://github.com/AsyncHttpClient/async-http-client

Создаем специальный actor для кеширования результатов тестирования.

Пример запроса :

http://localhost:8080/?testUrl=http://rambler.ru&count=20

3. Подсказки.

а. Инициализация http сервера в akka

```
public static void main(String[] args) throws IOException {
    System.out.println("start!");
    ActorSystem system = ActorSystem.create("routes");
    final Http http = Http.get(system);
    final ActorMaterializer materializer =
ActorMaterializer.create(system);
    final Flow<HttpRequest, HttpResponse, NotUsed> routeFlow = <вызов
метода которому передаем Http, ActorSystem и ActorMaterializer>;
    final CompletionStage<ServerBinding> binding = http.bindAndHandle(
            routeFlow,
            ConnectHttp.toHost("localhost", 8080),
            materializer
    );
    System.out.println("Server online at http://localhost:8080/\nPress
RETURN to stop...");
    System.in.read();
    binding
            .thenCompose(ServerBinding::unbind)
            .thenAccept(unbound -> system.terminate()); // and shutdown
when done
}
```

б. создаем в actorSystem – актор который принимает две команды – поиск уже готового результата тестирования и результат тестрования.

б. Общая логика требуемого flow

HttpRequest (этот запрос пришел снаружи) преобразуется в HttpResponse

```
Flow.of(HttpRequest.class)
```

→ map в Pair<url сайта из query параметра, Integer количество запросов>

→ mapAsync,

С помощью Patterns.ask посылаем запрос в кеширующий актор — есть ли результат. Обрабатываем ответ с помощью метода thenCompose

если результат уже посчитан, то возвращаем его как completedFuture

если нет, то создаем на лету flow из данных запроса, выполняем его и возвращаем CompletionStage<Long> :

```
Source.from(Collections.singletonList(r))
.toMat(testSink, Keep.right()).run(materializer);
```

→ map в HttpResponse с результатом а также посылка результата в кеширующий актор.

в. Общая логика создания внутреннего sink - testSink

С помощью метода create создаем Flow

Flow.<Pair<String, Integer>>create()

→ mapConcat размножаем сообщения до нужного количества копий

→ mapAsync – засекаем время, вызываем async http client и с помощью метода thenCompose вычисляем время и возвращаем future с временем выполнения запроса

→ завершаем flow : .toMat(fold, Keep.right()) ;

в данном случае fold – это агрегатор который подсчитывает сумму всех времен созаем его с помощью Sink.fold