

**Министерство цифрового развития, связи и массовых  
коммуникаций Российской Федерации  
Ордена трудового Красного Знамени  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский технический университет связи и информатики»  
Кафедра математической кибернетики и информационных технологий**

**Отчет по курсовой работе  
на тему «Akka HTTP REST сервис на языке программирования  
Scala»  
по дисциплине «Введение в ИТ»**

Выполнила: студентка группы  
БВТ1901 Московская  
Елизавета Дмитриевна  
Проверила: Мосева Марина  
Сергеевна

Москва, 2021

## **Оглавление**

1 Цель работы .....	3
2 Выполнение .....	4
3 Вывод.....	16

## 1 Цель работы

**Целью** данной работы является запуск и тестировании HTTP-приложения Akka, получении предварительного обзора того, как маршруты упрощают обмен данными по HTTP.

Приложение должно быть реализовано в следующих четырех исходных файлах:

- QuickstartApp.scala - содержит основной метод начальной загрузки приложения.
- UserRoutes.scala - HTTP-маршруты Akka, определяющие открытые эндпоинты.
- UserRegistry.scala - актор, обрабатывающий запросы на регистрацию.
- JsonFormats.scala - преобразует данные JSON из запросов в типы Scala и из типов Scala в ответы JSON.

## 2 Выполнение

```
1 lazy val akkaHttpVersion = "10.2.7"
2 lazy val akkaVersion      = "2.6.17"
3
4 lazy val root = (project in file(".")).
5 settings(
6   inThisBuild(List(
7     organization := "com.example",
8     scalaVersion := "2.13.4"
9   )),
10  name := "akka-http-quickstart-scala",
11  libraryDependencies ++= Seq(
12    "com.typesafe.akka" %% "akka-http"           % akkaHttpVersion,
13    "com.typesafe.akka" %% "akka-http-spray-json" % akkaHttpVersion,
14    "com.typesafe.akka" %% "akka-actor-typed"      % akkaVersion,
15    "com.typesafe.akka" %% "akka-stream"          % akkaVersion,
16    "ch.qos.logback"    % "logback-classic"      % "1.2.7",
17
18    "com.typesafe.akka" %% "akka-http-testkit"    % akkaHttpVersion % Test,
19    "com.typesafe.akka" %% "akka-actor-testkit-typed" % akkaVersion % Test,
20    "org.scalatest"     %% "scalatest"           % "3.2.9" % Test
21  )
22 )
```

Рисунок 1 – Содержимое файла build.sbt

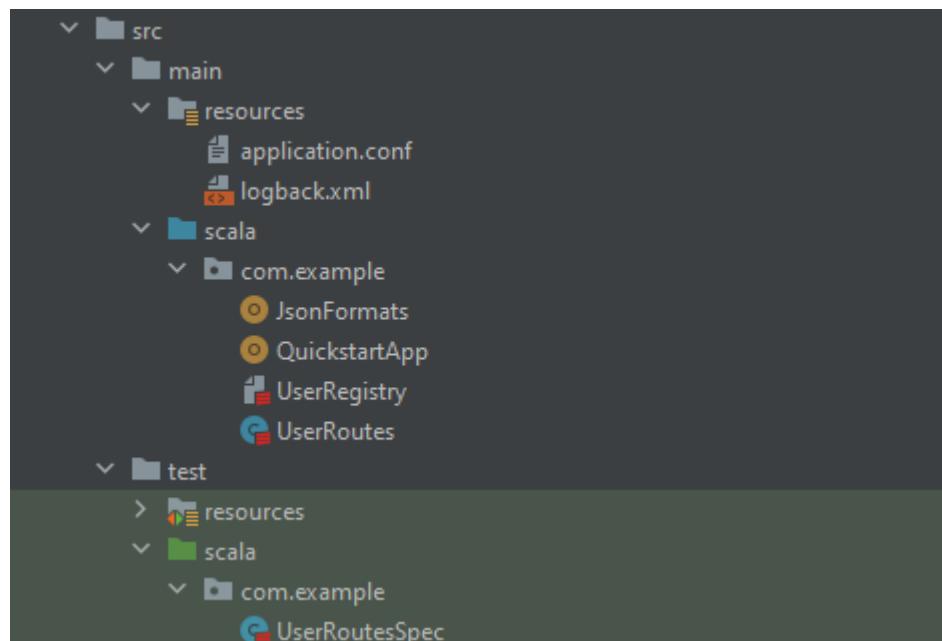


Рисунок 2 – Структура готового приложения

```

1  package com.example
2
3  import akka.actor.typed.ActorSystem
4  import akka.actor.typed.scaladsl.Behaviors
5  import akka.http.scaladsl.Http
6  import akka.http.scaladsl.server.Route
7  import scala.util.Failure
8  import scala.util.Success
9
10 // Основной класс
11 object QuickstartApp {
12   // Запуск HTTP сервера
13   private def startHttpServer(routes: Route)(implicit system: ActorSystem[_]): Unit = {
14     // Akka HTTP нуждается в классическом ActorSystem для запуска
15     import system.executionContext
16
17     val futureBinding = Http().newServerAt(interface = "localhost", port = 8080).bind(routes)
18     futureBinding.onComplete {
19       case Success(binding) =>
20         val address = binding.localAddress
21         system.log.info("Server online at http://{}/:{}/", address.getHostString, address.getPort)
22       case Failure(ex) =>
23         system.log.error("Failed to bind HTTP endpoint, terminating system", ex)
24         system.terminate()
25     }
26   }
27   def main(args: Array[String]): Unit = {
28     val rootBehavior = Behaviors.setup[Nothing] { context =>
29       val userRegistryActor = context.spawn(UserRegistry(), name = "UserRegistryActor")
30       context.watch(userRegistryActor)
31
32       val routes = new UserRoutes(userRegistryActor)(context.system)
33       startHttpServer(routes.userRoutes)(context.system)
34
35       Behaviors.empty
36     }
37     val system = ActorSystem[Nothing](rootBehavior, name = "HelloAkkaHttpServer")
38   }
39 }

```

Рисунок 3 – Содержимое файла QuickstartApp.scala

```
UserRoutes.scala
1 package com.example
2
3 import akka.http.scaladsl.server.Directives._
4 import akka.http.scaladsl.model.StatusCodes
5 import akka.http.scaladsl.server.Route
6
7 import scala.concurrent.Future
8 import com.example.UserRegistry._
9 import akka.actor.typed.ActorRef
10 import akka.actor.typed.ActorSystem
11 import akka.actor.typed.scaladsl.AskPattern._
12 import akka.util.Timeout
13
14 class UserRoutes(userRegistry: ActorRef[UserRegistry.Command])(implicit val system: ActorSystem[_]) {
15
16   import akka.http.scaladsl.marshallers.sprayjson.SprayJsonSupport._
17   import JsonFormats._
18
19   private implicit val timeout: Timeout = Timeout.create(system.settings.config.getDuration(path = "my-app.routes.ask-timeout"))
20
21   def getUsers(): Future[Users] =
22     userRegistry.ask(GetUsers)
23   def getUser(name: String): Future[GetUserResponse] =
24     userRegistry.ask(GetUser(name, _))
25   def createUser(user: User): Future[ActionPerformed] =
26     userRegistry.ask(CreateUser(user, _))
27   def deleteUser(name: String): Future[ActionPerformed] =
28     userRegistry.ask(DeleteUser(name, _))
29 }
```

Рисунок 4 – Содержимое файла UserRoutes.scala (часть 1)

```
UserRoutes.scala x
30  val userRoutes: Route =
31    pathPrefix( pm = "users") {
32      concat(
33        pathEnd {
34          concat(
35            get {
36              complete(getUsers())
37            },
38            post {
39              entity(as[User]) { user =>
40                onSuccess(createUser(user)) { performed =>
41                  complete((StatusCodes.Created, performed))
42                }
43              }
44            })
45          },
46          path(Segment) { name =>
47            concat(
48              get {
49                rejectEmptyResponse {
50                  onSuccess(getUser(name)) { response =>
51                    complete(response.maybeUser)
52                  }
53                }
54              },
55              delete {
56                onSuccess(deleteUser(name)) { performed =>
57                  complete((StatusCodes.OK, performed))
58                }
59              })
60          })
61      }
62    }
```

Рисунок 5 – Содержимое файла UserRoutes.scala (часть 2)

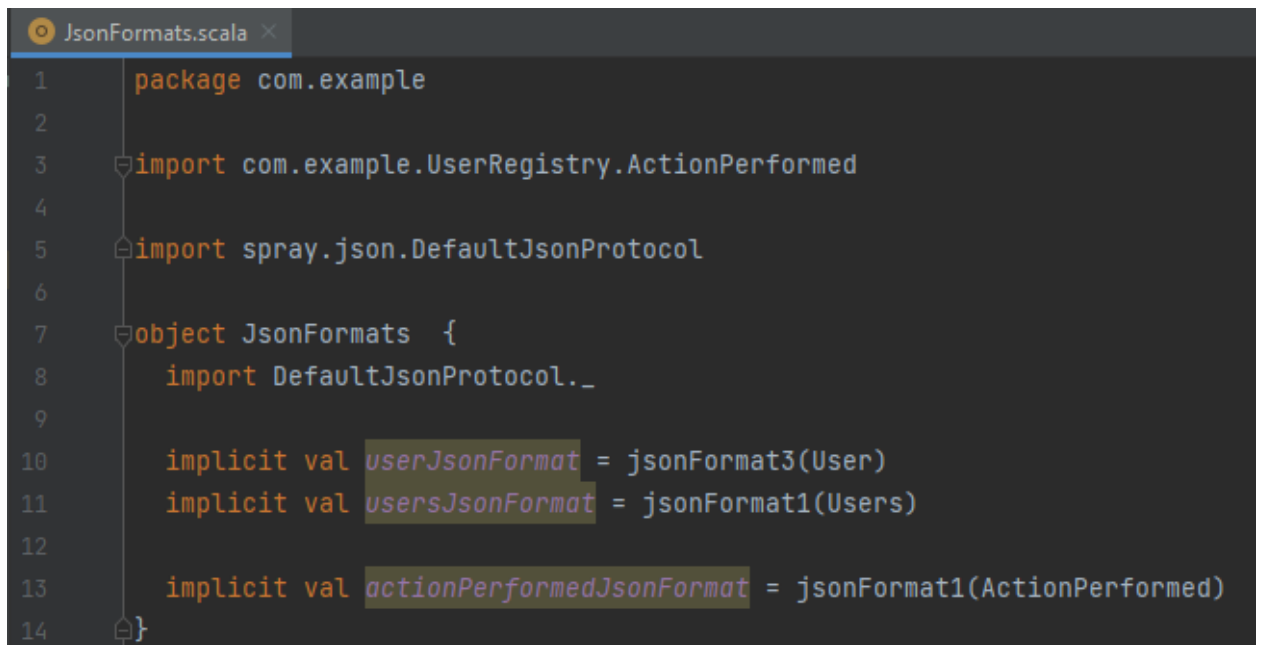
```

1 package com.example
2
3 import akka.actor.typed.ActorRef
4 import akka.actor.typed.Behavior
5 import akka.actor.typed.scaladsl.Behaviors
6 import scala.collection.immutable
7
8 final case class User(name: String, age: Int, countryOfResidence: String)
9 final case class Users(users: immutable.Seq[User])
10
11 object UserRegistry {
12   sealed trait Command
13   final case class GetUsers(replyTo: ActorRef[Users]) extends Command
14   final case class CreateUser(user: User, replyTo: ActorRef[ActionPerformed]) extends Command
15   final case class GetUser(name: String, replyTo: ActorRef[GetUserResponse]) extends Command
16   final case class DeleteUser(name: String, replyTo: ActorRef[ActionPerformed]) extends Command
17
18   final case class GetUserResponse(maybeUser: Option[User])
19   final case class ActionPerformed(description: String)
20
21   def apply(): Behavior[Command] = registry(Set.empty)
22
23   private def registry(users: Set[User]): Behavior[Command] =
24     Behaviors.receiveMessage {
25       case GetUsers(replyTo) =>
26         replyTo ! Users(users.toSeq)
27         Behaviors.same
28       case CreateUser(user, replyTo) =>
29         replyTo ! ActionPerformed(s"User ${user.name} created.")
30         registry(users + user)
31       case GetUser(name, replyTo) =>
32         replyTo ! GetUserResponse(users.find(_.name == name))
33         Behaviors.same
34       case DeleteUser(name, replyTo) =>
35         replyTo ! ActionPerformed(s"User $name deleted.")
36         registry(users.filterNot(_.name == name))
37     }
38 }

```

Рисунок 6 – Содержимое файла UserRegistry.scala



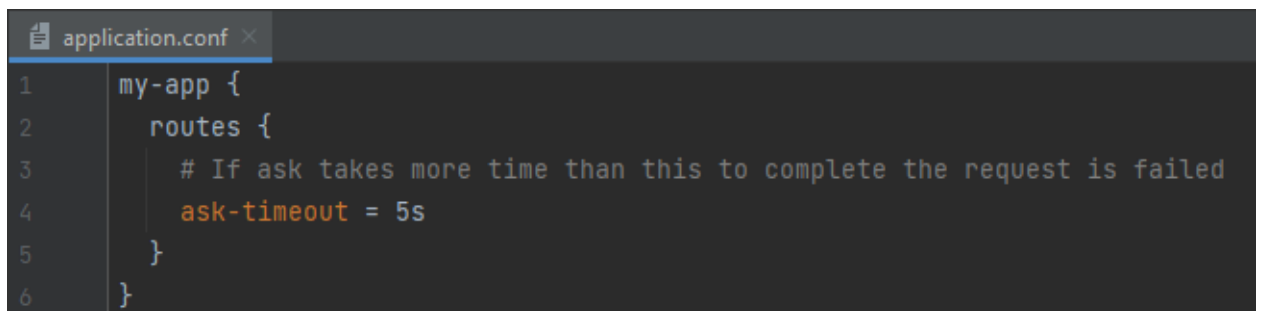


```

1 package com.example
2
3 import com.example.UserRegistry.ActionPerformed
4
5 import spray.json.DefaultJsonProtocol
6
7 object JsonFormats {
8   import DefaultJsonProtocol._
9
10  implicit val userJsonFormat = jsonFormat3(User)
11  implicit val usersJsonFormat = jsonFormat1(Users)
12
13  implicit val actionPerformedJsonFormat = jsonFormat1(ActionPerformed)
14 }

```

Рисунок 7 – Содержимое файла JsonFormats.scala

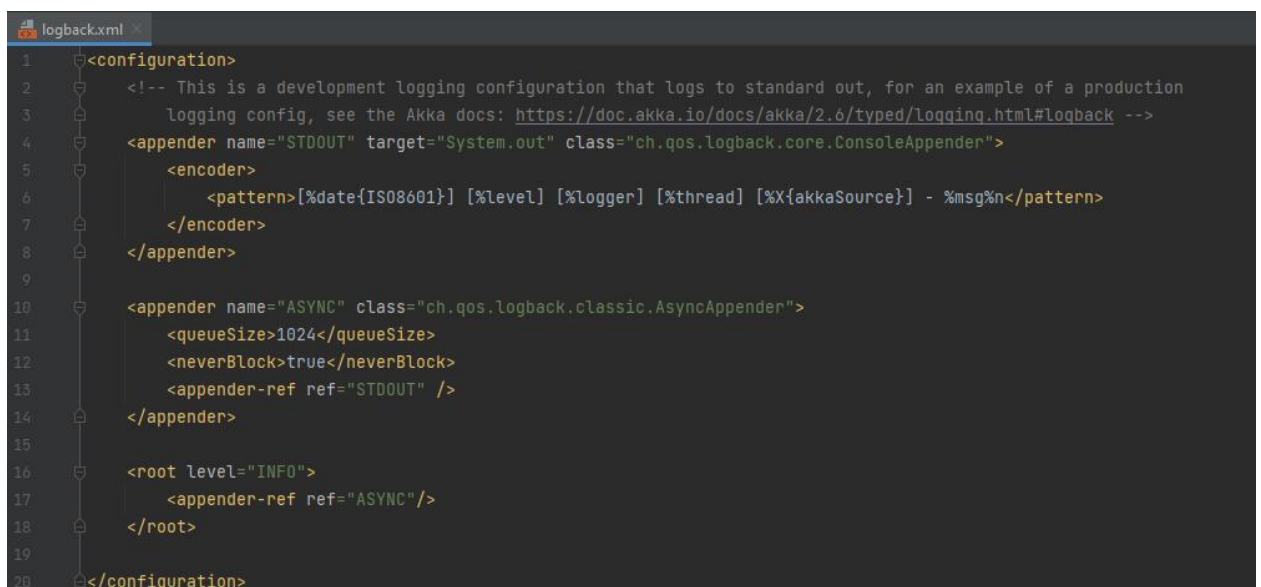


```

1 my-app {
2   routes {
3     # If ask takes more time than this to complete the request is failed
4     ask-timeout = 5s
5   }
6 }

```

Рисунок 8 – Содержимое файла application.conf



```

1 <configuration>
2   <!-- This is a development logging configuration that logs to standard out, for an example of a production
3        logging config, see the Akka docs: https://doc.akka.io/docs/akka/2.6/typed/logging.html#logback -->
4   <appender name="STDOUT" target="System.out" class="ch.qos.logback.core.ConsoleAppender">
5     <encoder>
6       <pattern>[%date{ISO8601}] [%level] [%logger] [%thread] [%X{akkaSource}] - %msg%n</pattern>
7     </encoder>
8   </appender>
9
10  <appender name="ASYNC" class="ch.qos.logback.classic.AsyncAppender">
11    <queueSize>1024</queueSize>
12    <neverBlock>true</neverBlock>
13    <appender-ref ref="STDOUT" />
14  </appender>
15
16  <root level="INFO">
17    <appender-ref ref="ASYNC"/>
18  </root>
19
20 </configuration>

```

Рисунок 9 – Содержимое файла logback.xml

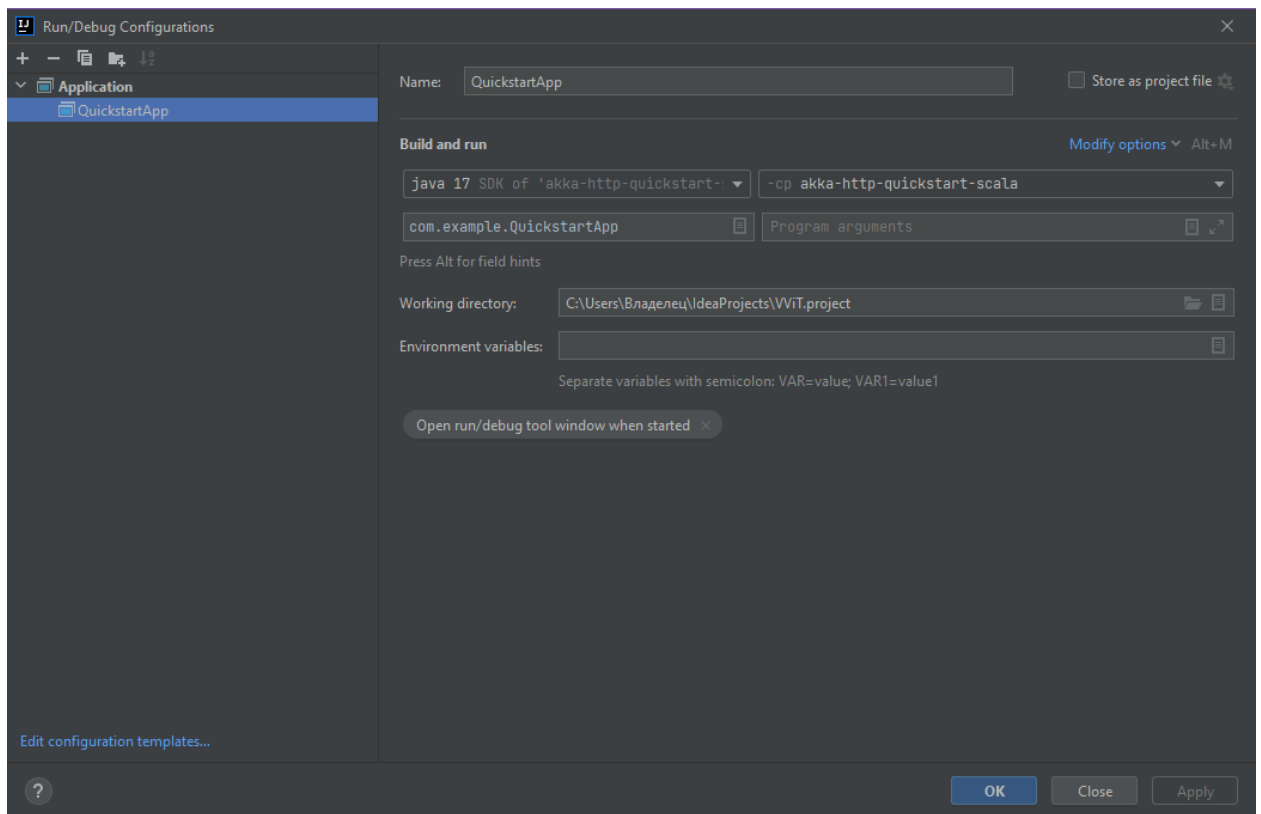


Рисунок 10 – Настройка конфигулятора для запуска

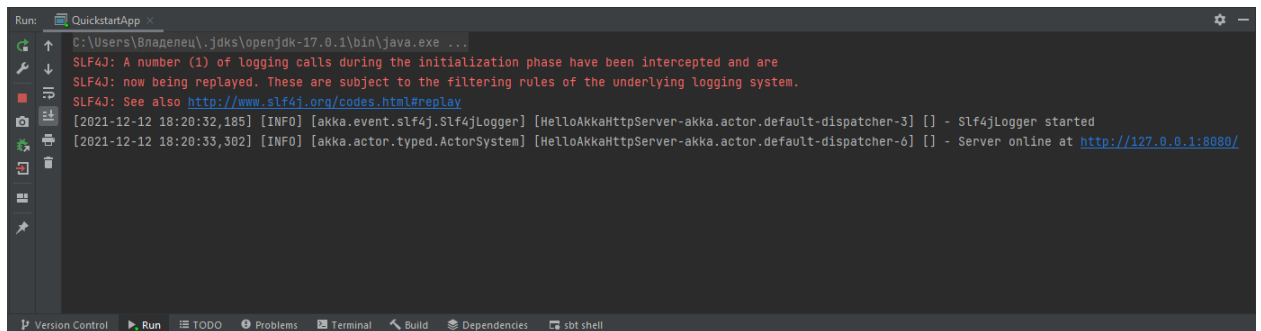


Рисунок 11 – Логирование только что запущенного приложения

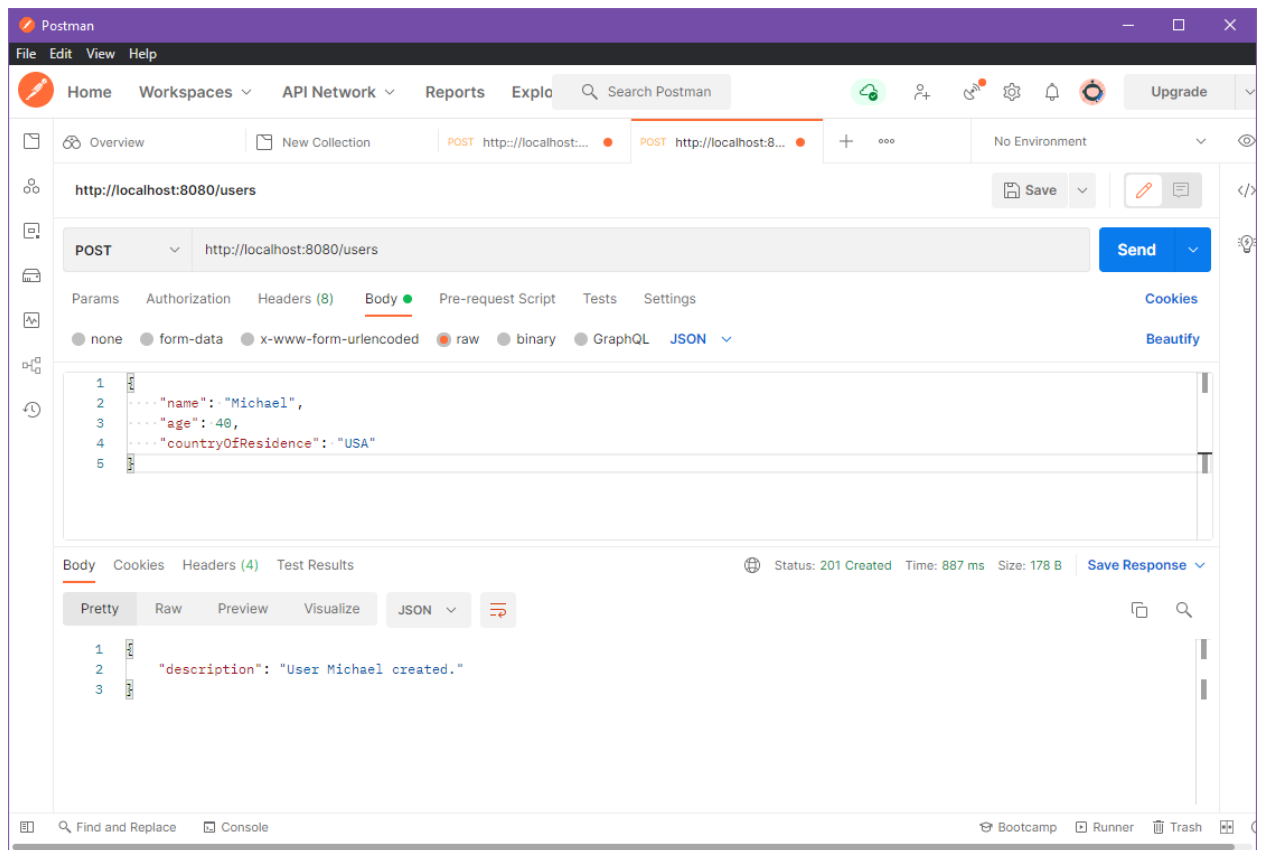


Рисунок 12 – Получение ответа на запрос, отправленный на порт 8080

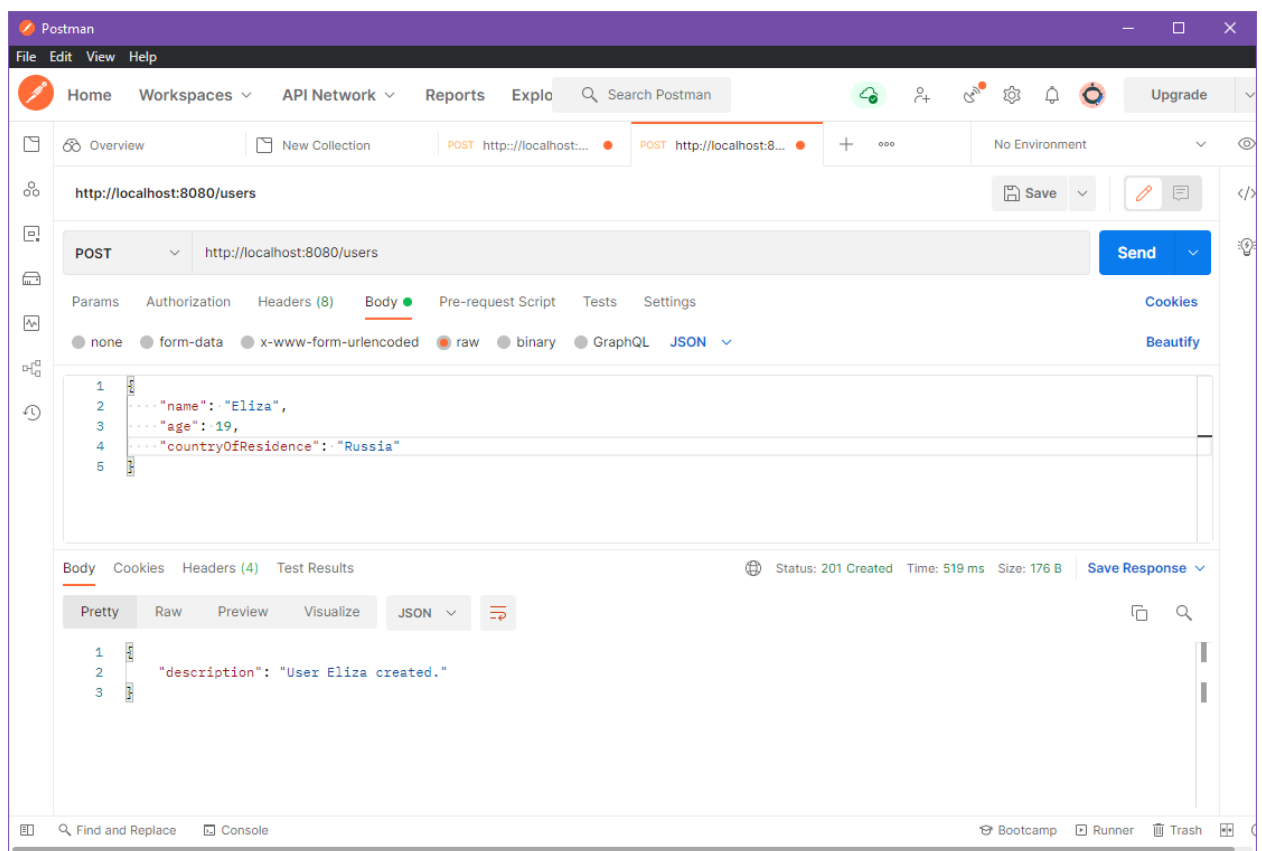


Рисунок 13 – Добавление пользователя

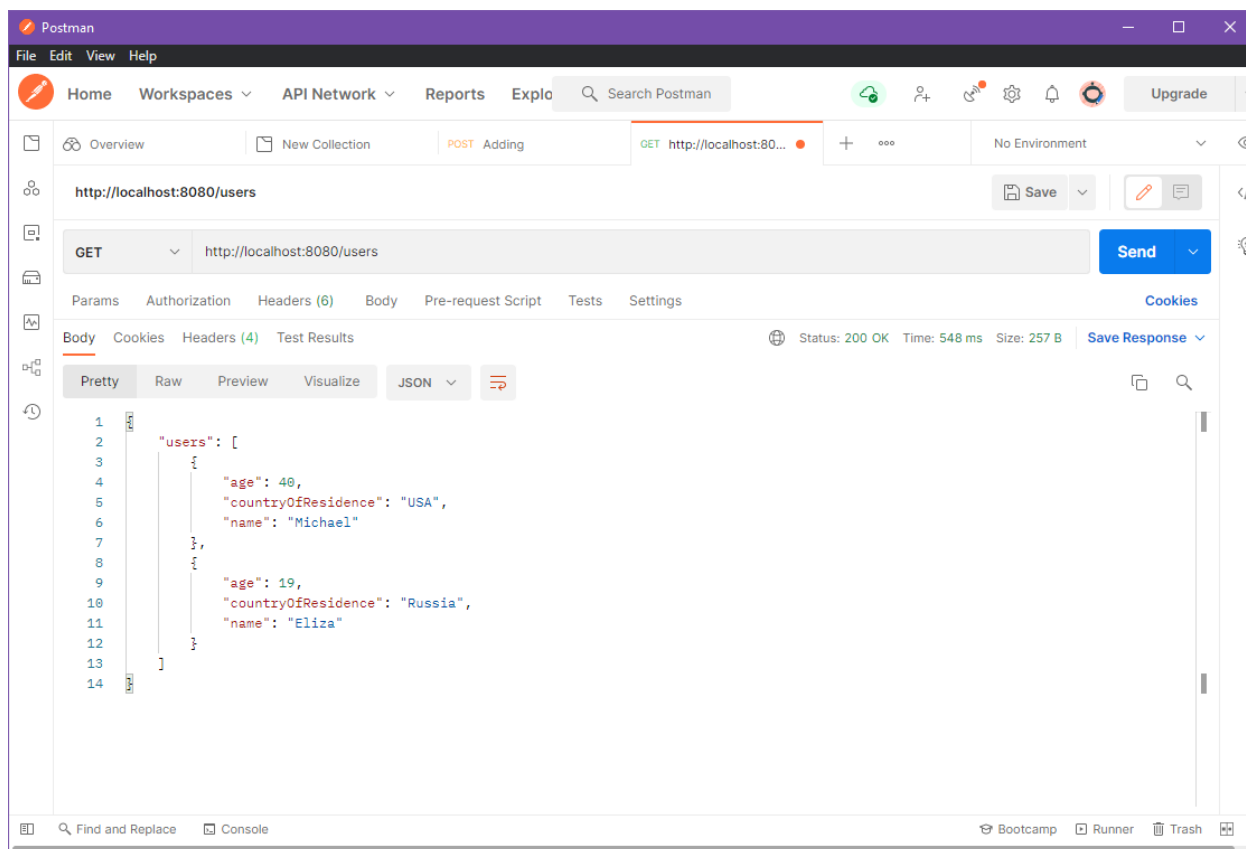


Рисунок 14 – Вывод списка всех пользователей

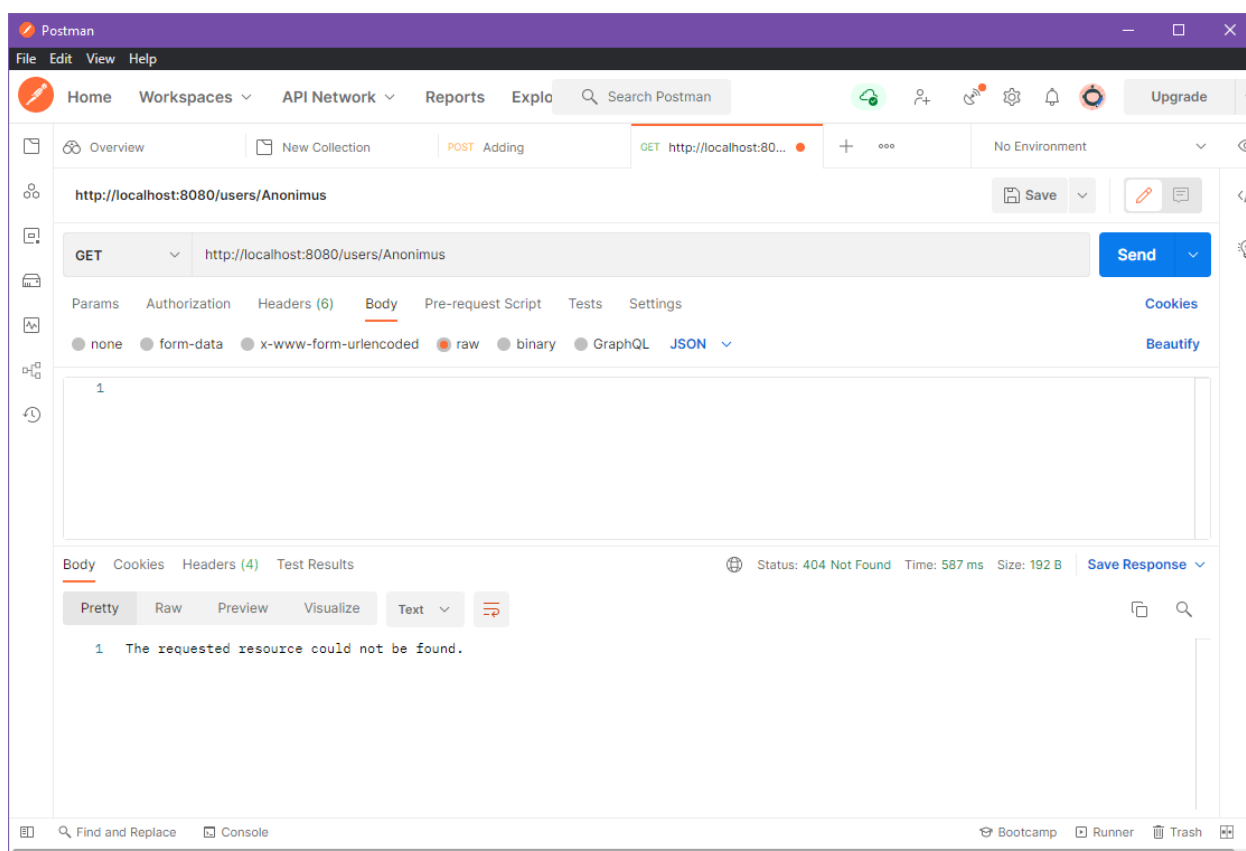


Рисунок 15 – Ответ на запрос «Запрошенный ресурс не может быть найден»

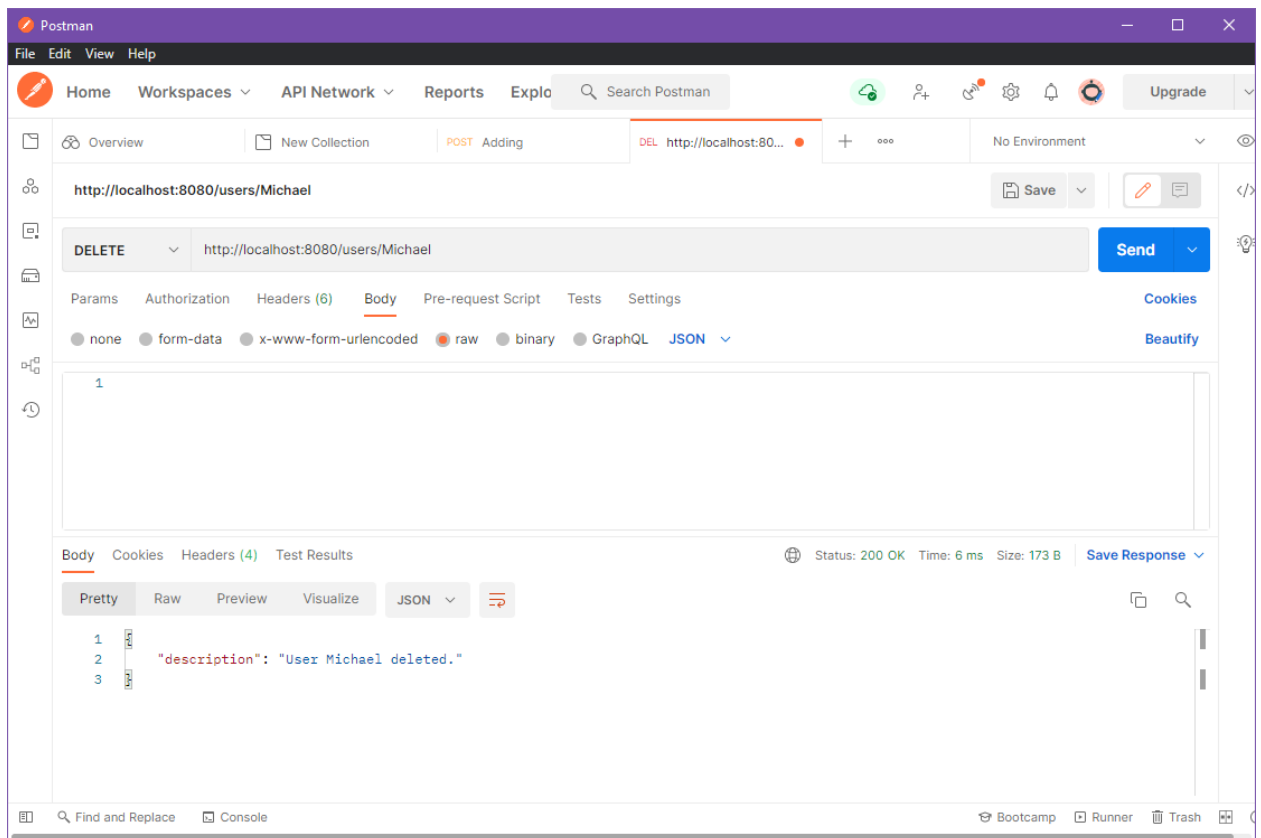


Рисунок 16 – Удаление пользователя

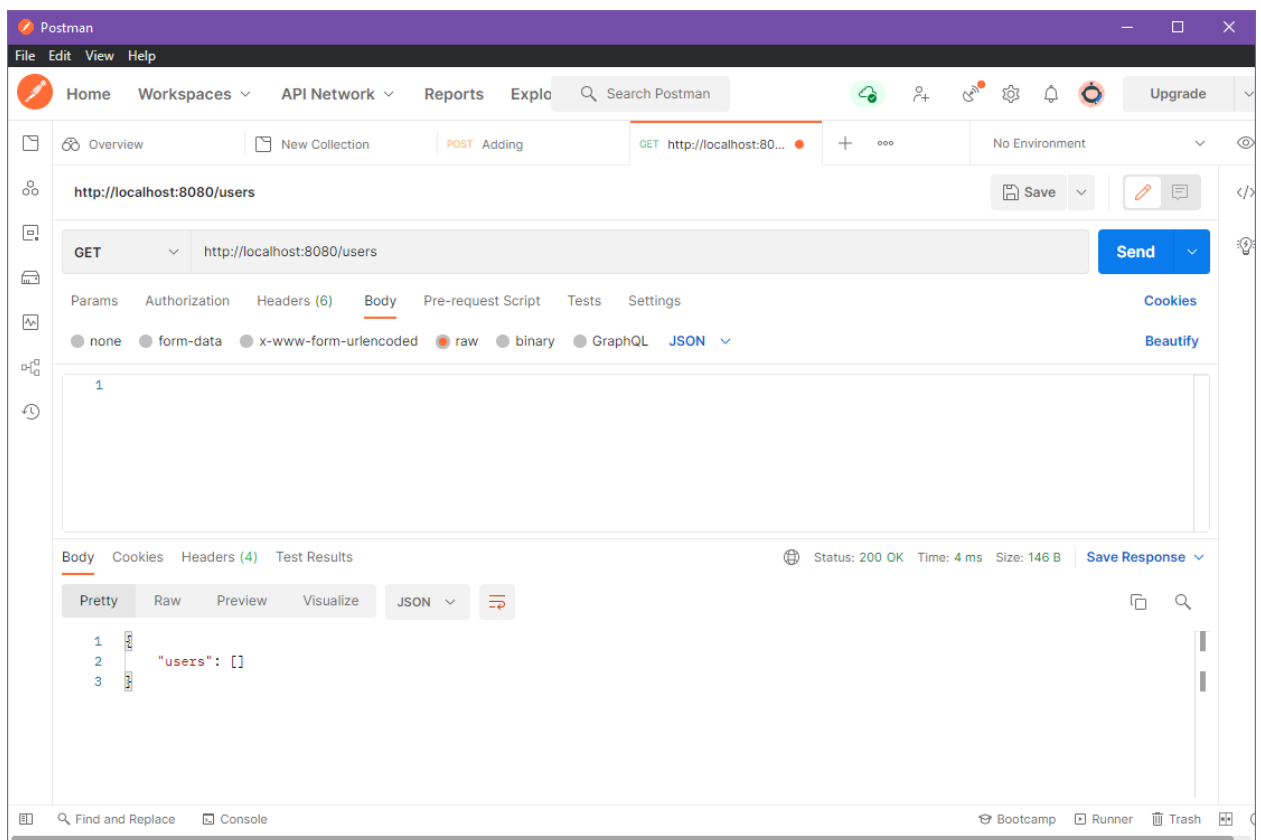


Рисунок 17 – Получение пустого списка на запрос получения списка удаленных пользователей

```

1 package com.example
2
3 import akka.actor.testkit.typed.scaladsl.ActorTestKit
4 import akka.http.scaladsl.marshalling.Marshal
5 import akka.http.scaladsl.model._
6 import akka.http.scaladsl.testkit.ScalatestRouteTest
7 import org.scalatest.concurrent.ScalaFutures
8 import org.scalatest.matchers.should.Matchers
9 import org.scalatest.wordspec.AnyWordSpec
10
11 class UserRoutesSpec extends AnyWordSpec with Matchers with ScalaFutures with ScalatestRouteTest {
12
13   lazy val testKit = ActorTestKit()
14   implicit def typedSystem = testKit.system
15   override def createActorSystem(): akka.actor.ActorSystem =
16     testKit.system.classicSystem
17
18   val userRegistry = testKit.spawn(UserRegistry())
19   lazy val routes = new UserRoutes(userRegistry).userRoutes
20
21   import akka.http.scaladsl.marshallers.sprayjson.SprayJsonSupport._
22   import JsonFormats._
23
24   "UserRoutes" should {
25     "return no users if no present (GET /users)" in {
26       val request = HttpRequest(uri = "/users")
27       request ~> routes ~> check {
28         status should ===(StatusCodes.OK)
29         contentType should ===(ContentTypes.`application/json`)
30         entityAs[String] should ===(right = """{"users":[]}""")
31       }
32     }
33     "be able to add users (POST /users)" in {
34       val user = User("Kapi", 42, "jp")
35       val userEntity = Marshal(user).to[MessageEntity].futureValue
36       val request = Post("/users").withEntity(userEntity)

```

Рисунок 16 – Содержание файла модульного теста (часть 1)

```

37
38     request ~> routes ~> check {
39       status should ===(StatusCodes.Created)
40       contentType should ===(ContentTypes.`application/json`)
41       entityAs[String] should ===(right = """{"description":"User Kapi created."}""")
42     }
43   }
44   "be able to remove users (DELETE /users)" in {
45     val request = Delete(uri = "/users/Kapi")
46     request ~> routes ~> check {
47       status should ===(StatusCodes.OK)
48
49       contentType should ===(ContentTypes.`application/json`)
50
51       entityAs[String] should ===(right = """{"description":"User Kapi deleted."}""")
52     }
53   }
54 }
55

```

Рисунок 17 – Содержание файла модульного теста (часть 2)

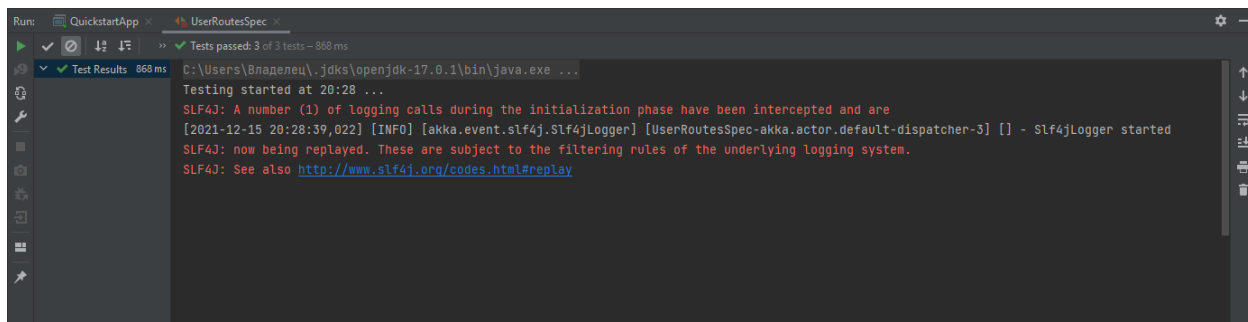


Рисунок 18 – Компиляция файла модульного теста

### **3 Вывод**

В ходе выполнения данной лабораторной работы были получены базовые навыки работы с Akka.