SCRIPTS DE TESTING

Actividad 4

ID BANNER
100097921 Jose Gregorio Vellojin

de Hoyos

```
package controllers.operation.user;

import com.fasterxml.jackson.annotation.JsonInclude;
import com.fasterxml.jackson.databind.ObjectMapper;
import config.MongoTestAppContext;
import config.TestAppContext;
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.Calendar;
/**
 * Agregar una descripcion de la clase
 *
 * @version 1.0.0
 */
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(classes
      = {
         MongoTestAppContext.class,
         TestAppContext.class
      })
@WebAppConfiguration
@FixMethodOrder(MethodSorters.NAME_ASCENDING)
public class User {

   @Autowired
   private WebApplicationContext webApplicationContext;
   private MockMvc mockMvc;
   private String token;
   @Autowired
   private DemographicService userService;
   @Autowired
   private SpecialistDao specialistDao;

   @Before
   public void setUp() throws Exception {
      MockitoAnnotations.initMocks(this);
      mockMvc = MockMvcBuilders
            .webAppContextSetup(webApplicationContext)
            .build();
      AuthorizedUser user = new AuthorizedUser();
      user.setUserName("tests");
      user.setLastName("Pruebas");
      user.setName("Proyect");
      user.setId(1);
      user.setBranch(1);
      token                                    =                                    JWT.generate(user,
Integer.parseInt(Tools.jsonObject(TestTools.getResponseString(mockMvc.perform(get("/api/confi
guration/tokenexpiration"))), Configuration.class).getValue()));
   }
```

```
@Test
public void test_01_create() throws Exception {

   //Crea el demographico de la historia
   Demographic demographic1 = new Demographic();
   demographic1.setName("CIUDAD");
   demographic1.setOrigin("H");
   demographic1.setEncoded(false);
   demographic1.setObligatory(Short.parseShort("0"));
   demographic1.setOrdering(Short.parseShort("8"));
   demographic1.setStatistics(false);
   demographic1.setLastOrder(false);
   demographic1.getUser().setId(1);
   demographic1.setModify(true);

   ResultActions result = mockMvc.perform(post("/api/demographics")
         .header("Authorization", token)
         .contentType(TestTools.APPLICATION_JSON_UTF8)
         .content(Tools.jsonObject(demographic1)));
   demographic1 = Tools.jsonObject(TestTools.getResponseString(result), Demographic.class);

   //Obtiene los demograficos
   List<Demographic> demographics = userService.list(true);
   Calendar calendar = Calendar.getInstance();
   Patient patient = new Patient();
   patient.setPatientId("123456");
   patient.setName1("NOMBRE 1");
   patient.setName2("NOMBRE 2");
   patient.setLastName("APELLIDO 1");
   patient.setSurName("APELLIDO 2");
   calendar.set(1987, 0, 19);
   patient.setBirthday(calendar.getTime());
   patient.setEmail("informacion@ibero.edu");
   patient.setSize(new BigDecimal(180));
   patient.setWeight(new BigDecimal(87.5));
   patient.getDocumentType().setId(1);
   patient.getRace().setId(1);
   patient.getSex().setId(9);
   List<DemographicValue> demographicList = new ArrayList<>();
   DemographicValue demoValue = new DemographicValue();
   demoValue.setIdDemographic(demographic1.getId());
   demographicList.add(demoValue);

   if (!demographics.isEmpty())
   {
      DemographicValue value = null;
      for (Demographic demographic : demographics)
      {
         if (demographic.getOrigin().equals("H"))
         {
            value = new DemographicValue();
```

```
            value.setIdDemographic(demographic.getId());
            if (demographic.isEncoded())
            {
               value.setCodifiedId(3);
            } else
            {
               value.setNotCodifiedValue("123456");
            }
            patient.getDemographics().add(value);
         }
      }
   }

   String response = TestTools.getResponseString(mockMvc.perform(post("/api/users")
         .header("Authorization", token)
         .contentType(TestTools.APPLICATION_JSON_UTF8)
         .content(Tools.jsonObject(patient))));

   patient = Tools.jsonObject(response, Patient.class);
   Order order = new Order();
   order.setCreatedDateShort(20200521);
   OrderType type = new OrderType();
   type.setId(1);
   type.setCode("V");
   order.setType(type);
   order.setPatient(patient);
   order.setHomebound(false);
   order.setMiles(0);
   order.setActive(true);
   order.setExternalId("123456");

   AuthorizedUser user = new AuthorizedUser();
   user.setId(1);

   rate = Tools.jsonObject(response, Rate.class);
   order.setRate(rate);
   order.setOrderNumber(202005210001L);

   ObjectMapper mapper = new ObjectMapper();
   mapper.setSerializationInclusion(JsonInclude.Include.NON_NULL);
   String jsonContent = mapper.writeValueAsString(order);

   TestScript.deleteData("lab24");
   TestScript.deleteData("lab39");
   TestScript.deleteData("lab57");
   //Usuario
   TestScript.execTestUpdateScript("INSERT INTO user VALUES (1, 'Prueba', 1, 3, 1, 'Test', 10,
'2017-09-11 13:48:21.775', 1, 1, 1, '1', '1,2,3', 1)");


   mockMvc.perform(post("/api/users")
```

```
            .header("Authorization", token)
            .contentType(TestTools.APPLICATION_JSON_UTF8)
            .content(jsonContent))
            .andExpect(status().isOk());
    }

    @Test
    public void test_02_priceOfTest() throws Exception {
        TestScript.createInitialOrders();
        TestScript.execTestUpdateScript("DELETE FROM user");
        TestScript.execTestUpdateScript("INSERT INTO user VALUES (1, 1, 100)");

        mockMvc.perform(get("/api/user/idtest/1/rate/1")
            .header("Authorization", token)
            .contentType(TestTools.APPLICATION_JSON_UTF8))
            .andExpect(status().isOk())
            .andExpect(content().contentType(TestTools.APPLICATION_JSON_UTF8))
            .andExpect(content().string("100"));
    }

    @Test
    public void test_07_getRecalledOrder() throws Exception {
        TestScript.execTestUpdateScript("INSERT INTO user (dateI, dateF) VALUES (201709080001,
201709080002); ");
        mockMvc.perform(get("/api/user/filter/recalled/order/" + 201709080001L)
            .header("Authorization", token)
            .contentType(TestTools.APPLICATION_JSON_UTF8))
            .andExpect(status().isOk())
            .andExpect(content().contentType(TestTools.APPLICATION_JSON_UTF8))
            .andExpect(jsonPath("$.daughterOrder[0]", is(201709080002L)));
    }

    @Test
    public void test_14_getDemographics() throws Exception
    {
        mockMvc.perform(get("/api/orders/demographics/H")
            .header(token, "Authorization"))
            .andExpect(status().isNoContent());
    }

    @Test
    public void test_16_getLastOrderNoContent() throws Exception {

        TestScript.createInitialOrders();
        mockMvc.perform(get("/api/user/last/user/1")
            .header("Authorization", token)
            .contentType(TestTools.APPLICATION_JSON_UTF8))
            .andExpect(status().isNoContent());
    }
}
```