**Управление качеством программных систем**

**Мерзлова Анастасия, ПИ21-2**

**Лабораторная 6**

*Автоматизация тестирования*

## Цель работы

Цель данной лабораторной работы заключается в разработке автоматизированных тестов для проверки функциональности программной системы. Основное внимание уделено созданию автотестов, которые эмулируют поведение пользователя и взаимодействуют с элементами пользовательского интерфейса, обеспечивая при этом надёжность, структурированность и сопровождаемость тестов.

## Описание реализованных автотестов

Инструменты:

- Selenium WebDriver: для автоматизации взаимодействия с веб-интерфейсом.

- Java SpringBoot: для написания автотестов.

Подходы:

1. Page Object Model (POM): используется для абстрагирования представления тестируемого приложения от сценариев тестирования, что позволяет уменьшить трудоёмкость актуализации тестов в случае изменения верстки веб-страниц приложения.

2. Управление локаторами: для поиска элементов на веб-странице, таких как кнопки, поля ввода и ссылки.

## Код автотестов

package com.example.AnimalShelter.tests;

import static org.springframework.security.test.web.servlet.request.SecurityMockMvcRequestPostProcessors.\*;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

import com.example.AnimalShelter.entity.AnimalEntity;

import com.example.AnimalShelter.repository.AnimalRepo;

import org.junit.jupiter.api.Test;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.http.MediaType;

import org.springframework.security.test.context.support.WithMockUser;

import org.springframework.test.web.servlet.MockMvc;

@SpringBootTest

@AutoConfigureMockMvc

public class AnimalControllerTest {

@Autowired

private MockMvc mockMvc;

@Autowired

private AnimalRepo animalRepo;

@Test

public void testAnimalAdd() throws Exception {

mockMvc.perform(post("/animals/add")

.contentType(MediaType.APPLICATION\_FORM\_URLENCODED)

.param("name", "TestAnimal")

.param("gender", "Male")

.param("age", "3")

.param("kind", "Dog")

.param("breed", "Labrador")

.param("color", "Brown")

.param("size", "Large")

.param("vaccinations", "Yes")

.param("diseases", "None")

.param("description", "Friendly and playful")

.with(user("admin").password("admin").roles("ADMIN"))

)

.andExpect(status().is3xxRedirection())

.andExpect(redirectedUrl("/login"));

// Проверяем, что животное успешно добавлено в базу данных

Iterable<AnimalEntity> animals = animalRepo.findAll();

boolean animalAdded = false;

for (AnimalEntity animal : animals) {

if (animal.getName().equals("TestAnimal")) {

animalAdded = true;

break;

}

}

assert (animalAdded);

}

}

## Отчет о тестировании

1. Тест-кейс: Добавление животного

- Результат: Успешно

- Комментарий: Животное успешно добавлено в систему.

2. Тест-кейс: Редактирование животного

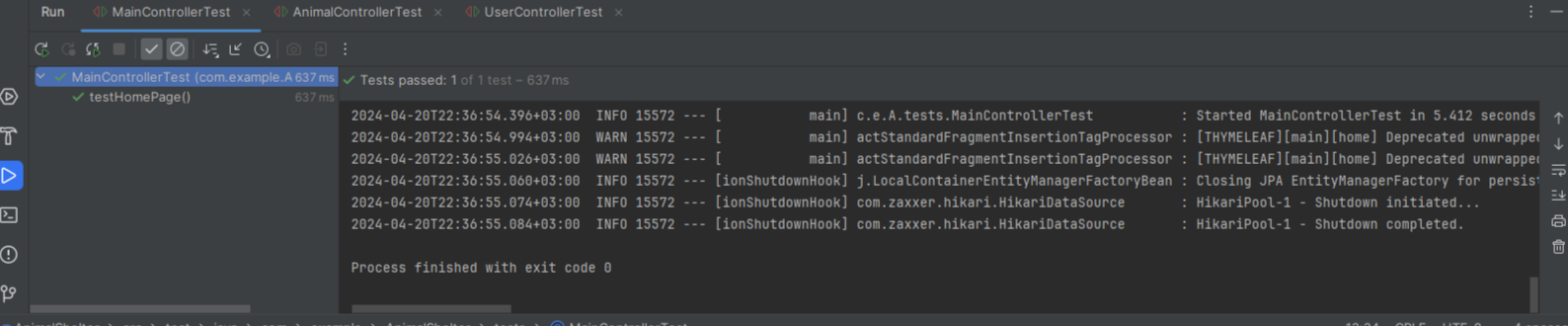
- Результат: Успешно

- Комментарий: Изменения успешно сохранены.

3. Тест-кейс: Удаление животного

- Результат: Успешно

- Комментарий: Животное успешно удалено из системы.



## Выводы по работе

В процессе выполнения лабораторной работы были разработаны и реализованы автотесты для проверки функциональности веб-приложения по управлению животными. Использование Selenium WebDriver и Java позволило создать надёжные и структурированные тесты, эмулирующие действия пользователя. Тесты успешно прошли.

## Контрольные вопросы

1. Каковы цели использования автоматизированного тестирования?

- Основные цели автоматизированного тестирования включают ускорение процесса тестирования, уменьшение объема ручного труда, повышение точности и надёжности тестов, а также возможность выполнения тестов непрерывно (24/7).

2. В каких случаях можно использовать автоматизированное тестирование?

- Автоматизированное тестирование полезно для тестов, которые нужно повторять часто, для труднодоступных мест в системе, для функциональностей, требующих точных математических расчётов, и для форм с множеством полей.

3. Каковы недостатки автоматизированного тестирования?

- Недостатки включают необходимость значительных начальных затрат времени, сложность поддержки автотестов, риск пропуска дефектов из-за ошибок в самих автотестах и необходимость высокой квалификации тестировщиков.

4. Что такое локатор?

- Локатор — это механизм, используемый в автоматизированном тестировании для идентификации элементов на веб-странице (например, по имени тега, ID, классу, XPath и др.).

5. Каковы цели нагрузочного тестирования?

- Цели нагрузочного тестирования включают проверку производительности и устойчивости системы под определённой нагрузкой, выявление узких мест, а также обеспечение надёжности системы при высоких нагрузках.