Санкт-Петербургский национальный исследовательский университет

информационных технологий, механики и оптики

**Лабораторная работа №1**

**Дисциплина «Тестирование программного обеспечения»**

**Вариант 19**

**Выполнил:**

Съестов Дмитрий Вячеславович

Группа P3317

**Преподаватель:**

Исаев Илья Владимирович

Санкт-Петербург

2019

**Задание**

Разработка программного обеспечения для автоматизированного тестирования веб-приложения с помощью Selenium WebDriver.

Написать десктопный программный продукт для автоматизированного тестирования сайта (по вариантам):

* Написать тестовый сценарий, который будет покрывать наибольшее количество функций данного сайта (80 - 90%);
* По тестовому сценарию написать программу, использующую инструмент автоматизированного тестирования (в качестве инструмента Selenium, язык программирования Java/Kotlin, предпочтительный браузер – Chrome);
* Программа должна предусматривать такие функции, как - поиск элементов на странице, переход по элементам (например, нашли кнопку, кликнули на неё, проверили, что на странице есть все нужные элементы);
* Открывать диалоговые окна, и проверять информацию в них.
* Обязательно проверить форму регистрации (кроме 1 варианта).

Проверить функционал веб-приложения Google презентации. Ограничиться только меню.

**Selenium**

Selenium - это инструмент для автоматизированного управления браузерами.Наиболее популярной областью применения Selenium является автоматизация тестирования веб-приложений. Selenium является центральным компонентом целого ряда других инструментов и фреймворков автоматизации.

Selenium поддерживает десктопные и мобильные браузеры и позволяет разрабатывать сценарии автоматизации практически на любом языке программирования.

**Тестовые сценарии**

При помощи Selenium тестируются следующие пункты меню:

* Файл
  + Создать
  + Создать копию
  + Скачать как
  + Информация о документе
  + Язык
  + Настройки страницы
* Вид
  + Анимация
  + Привязка
  + В виде сетки
  + Направляющие
  + Показать линейку
  + Масштаб
* Вставка
  + Анимация
  + Оставить комментарий
  + Номера слайдов
  + Диаграмма
  + Таблица
  + Текстовое поле
  + WordArt
* Слайд
  + Новый слайд
  + Дублировать слайд
  + Удалить слайд
  + Пропустить слайд
  + Изменить фон
  + Сменить тему
  + Переход
* Инструменты
  + Настройки
  + Настройки специальных возможностей
  + Расширенный поиск
  + Проверка правописания
  + Словарь
* Панели инструментов
  + Изображение
  + Линия
  + Масштаб
  + Форма
  + Тема слайда
  + Переход слайда
  + Новый слайд
  + Дублировать слайд
  + Удалить слайд
  + Цвет текста
  + Цвет заливки
  + Шрифт
  + Размер шрифта
  + Цвет линии
  + Пунктир
  + Толщина линии
* Правый клик
  + Изменить фон
  + Оставить комментарий
  + Выбрать макет
  + Сменить тему
  + Переход
  + Новый слайд
  + Дублировать слайд
  + Удалить слайд

**Найденные баги**

Поскольку над приложением Google Презентации работали высококвалифицированные разработчики и тестировщики, багов не найдено.

**Код**

Для запуска тестов используется класс TestRunner, который инициализирует драйвер и веб-страницу и запускает сами тесты:

package tpo1  
  
import org.openqa.selenium.firefox.FirefoxDriver  
import org.openqa.selenium.support.ui.WebDriverWait  
import tpo1.tests.BarTests  
import tpo1.tests.HiddenBarTests  
import tpo1.tests.RightClickTests  
import tpo1.tests.menu.\*  
import java.util.concurrent.TimeUnit  
  
  
class TestRunner {  
 private val driver = initDriver()  
 private val wait = WebDriverWait(driver, 10)  
 private var page: PresentationPage  
  
 init {  
 try {  
 driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.SECONDS)  
 driver.manage().timeouts().setScriptTimeout(10, TimeUnit.SECONDS)  
 driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS)  
 driver.manage().window().maximize()  
  
 page = PresentationPage(driver, wait)  
 } catch (ex: Exception) {  
 driver.quit()  
 throw ex  
 }  
 }  
  
 private fun initDriver(): FirefoxDriver {  
 System.setProperty("webdriver.gecko.driver", "@DRIVER\_LOCATION@")  
 System.setProperty(FirefoxDriver.SystemProperty.BROWSER\_LOGFILE, "/dev/null")  
  
 return FirefoxDriver()  
 }  
  
 fun runTests() {  
 FileMenuTests(page).run()  
 page.reload()  
 ViewMenuTests(page).run()  
 page.reload()  
 InsertMenuTests(page).run()  
 page.reload()  
 SlideMenuTests(page).run()  
 page.reload()  
 ToolsMenuTests(page).run()  
 page.reload()  
  
 BarTests(page).run()  
 HiddenBarTests(page).run()  
 RightClickTests(page).run()  
 }  
  
 fun cleanup() {  
 page.deletePresentation()  
 driver.quit()  
 }  
}  
  
fun main(args: Array<String>) {  
 var runner: TestRunner? = null  
 try {  
 runner = TestRunner()  
 runner.runTests()  
 } finally {  
 runner?.cleanup()  
 }  
}

Веб-страница описывается абстрактным классом AbstractWebPage, который содержит несколько вспомогательных методов:

package tpo1  
  
import org.openqa.selenium.\*  
import org.openqa.selenium.interactions.Actions  
import org.openqa.selenium.support.ui.WebDriverWait  
  
abstract class AbstractWebPage(val driver: WebDriver, val wait: WebDriverWait) {  
 val actions = Actions(driver)  
  
 fun find(locator: By): WebElement? {  
 return try {  
 val element = driver.findElement(locator)  
 wait.until **{** element.isDisplayed **}** element  
 } catch (e: NoSuchElementException) {  
 null  
 }  
 }  
  
 fun findById(id: String): WebElement? {  
 return find(By.id(id))  
 }  
  
 fun findByName(name: String): WebElement? {  
 return find(By.name(name))  
 }  
  
 fun findByCss(selector: String): WebElement? {  
 return find(By.cssSelector(selector))  
 }  
  
 fun findByXpath(xpath: String): WebElement? {  
 return find(By.xpath(xpath))  
 }  
  
 fun findByTag(tag: String): WebElement? {  
 return find(By.tagName(tag))  
 }  
  
 fun findByClass(klass: String): WebElement? {  
 return find(By.className(klass))  
 }  
  
 fun reload() {  
 driver.navigate().refresh()  
 wait.until **{** webDriver **->** (webDriver as JavascriptExecutor).executeScript(“return document.readyState”) == “complete”  
 **}** }  
  
 protected open fun onLoad () { }  
}

Страница презентации наследует этот класс:

package tpo1  
  
import org.openqa.selenium.By  
import org.openqa.selenium.Keys  
import org.openqa.selenium.WebDriver  
import org.openqa.selenium.WebElement  
import org.openqa.selenium.support.FindBy  
import org.openqa.selenium.support.PageFactory  
import org.openqa.selenium.support.ui.WebDriverWait  
  
class PresentationPage(driver: WebDriver, wait: WebDriverWait) : AbstractWebPage(driver, wait) {  
 companion object {  
 const val URL = """https://docs.google.com/presentation/"""  
 const val LOGIN = "@LOGIN@"  
 const val PASSWORD = "@PASSWORD@"  
 const val IMAGE\_URL = """https://ssl.gstatic.com/docs/templates/thumbnails/slides-blank-googlecolors.png"""  
 const val TITLE = "Лабораторная работа №1"  
 }  
  
 init {  
 driver.get(URL)  
 login()  
 createPresentation()  
 do try {  
 PageFactory.initElements(driver, this)  
 onLoad()  
 enterPresentationTitle()  
 break  
 } catch (e: Exception) {  
 reload()  
 } while (true)  
 }  
  
 @FindBy(css="input.docs-title-input")  
 lateinit var nameInput: WebElement  
  
 private fun login() {  
 val loginField = findById("identifierId")!!  
 val loginNextButton = findById("identifierNext")!!  
 loginField.sendKeys(LOGIN)  
 loginNextButton.click()  
  
 val passwordField = findByName("password")!!  
 val passwordNextButton = findById("passwordNext")!!  
 passwordField.sendKeys(PASSWORD)  
 passwordNextButton.click()  
 }  
  
 private fun createPresentation() {  
 val newPresentation = findByCss("div > img[src=\"$IMAGE\_URL\"]")!!  
 wait.until **{** newPresentation.isEnabled **}** newPresentation.click()  
 }  
  
 override fun onLoad() {  
 val locator = By.xpath("//div[@class='punch-theme-sidebar-header']//div[@title='Закрыть']/div")  
 find(locator)?.click()  
 }  
  
 private fun enterPresentationTitle() {  
 wait.until **{** nameInput.isEnabled **}** Thread.sleep(250)  
 nameInput.clear()  
 Thread.sleep(250)  
 nameInput.sendKeys(TITLE, Keys.ENTER)  
 }  
  
 fun deletePresentation() {  
 driver.get("https://docs.google.com/presentation/u/0/")  
 findByXpath("//div[@class='docs-homescreen-grid-item-metadata-container'][contains(., '$TITLE')]")?.let **{** val button = it.findElement(By.cssSelector("div[aria-label=\"Ещё (раскрывающееся меню)\"]"))  
 button.click()  
  
 val deleteButton = findByXpath("//div[@class='goog-menuitem'][contains(., 'Удалить')]")!!  
 deleteButton.click()  
 **}** }  
}

Для запуска тестов используется класс AbstractTestSet. С помощью рефлексии он находит методы, отмеченные аннотацией @Test, запускает их и выводит результаты при вызове метода run():

package tpo1.tests  
  
import org.openqa.selenium.ElementNotInteractableException  
import org.openqa.selenium.Keys  
import org.openqa.selenium.WebElement  
import org.openqa.selenium.support.PageFactory  
import org.openqa.selenium.support.ui.ExpectedConditions  
import tpo1.PresentationPage  
import kotlin.reflect.full.memberFunctions  
  
abstract class AbstractTestSet(private val name: String, protected val page: PresentationPage) {  
 init {  
 PageFactory.initElements(page.driver, this)  
 }  
  
 @Target(AnnotationTarget.FUNCTION)  
 protected annotation class Test(val spec: String)  
  
 protected sealed class TestResult {  
 object Success : TestResult()  
 data class Failure(val message: String?) : TestResult()  
 }  
  
 abstract fun initializeTest()  
  
 protected fun basicWindowTest(item: WebElement, expectedXpath: String) {  
 initializeTest()  
 item.click()  
 page.findByXpath(expectedXpath)?.click() ?: fail("Element with xpath \"$expectedXpath\" not found.")  
 }  
  
 protected fun basicEscapeTest(item: WebElement, expectedXpath: String) {  
 initializeTest()  
 item.click()  
 assertExists(expectedXpath)  
 page.actions.sendKeys(Keys.ESCAPE).perform()  
 }  
  
  
  
 protected fun fail(msg: String) {  
 throw AssertionError(msg)  
 }  
  
 protected fun assertExists(xpath: String) {  
 val item = page.findByXpath(xpath)  
 page.wait.until **{** ExpectedConditions.visibilityOf(item) **}** if (item == null) fail("Element with xpath \"$xpath\" not found.")  
 }  
  
 private fun runTest(test: () -> Unit): TestResult {  
 try {  
 while (true) try {  
 test()  
 Thread.sleep(200)  
 return TestResult.Success  
 } catch(e: ElementNotInteractableException) { }  
 } catch (e: Exception) {  
 val cause = e.cause ?: e  
 page.reload()  
 return TestResult.Failure(when (cause) {  
 is AssertionError -> cause.message  
 else -> "${cause::class.simpleName}: ${cause.localizedMessage.substringBefore('\n')}"  
 })  
 }  
 }  
  
 open fun run() {  
 val tests = this::class.memberFunctions.filter **{** func **->** func.annotations.any **{ it** is Test **}** && func.parameters.count() == 1  
 **}**.map **{** testFunc **->** val annotation = testFunc.annotations.filterIsInstance(Test::class.java).single()  
 Pair(annotation.spec, testFunc)  
 **}**.toMap().toSortedMap()  
  
 println("---------- ${this.name.toUpperCase()} ----------")  
 var successfulTests = 0  
 tests.forEach **{** spec, test **->** val result = runTest **{** test.call(this) **}** if (result is TestResult.Success) successfulTests++  
 val resultText = when (result) {  
 is TestResult.Failure -> "✗ (${result.message ?: "no message"})"  
 else -> "✔"  
 }  
 println("$spec: $resultText")  
 **}** println("$successfulTests/${tests.count()} succeeded\n")  
 }  
}

Для тестирования пунктов меню используется его расширение MenuTestSet:

package tpo1.tests.menu  
  
import org.openqa.selenium.WebElement  
import org.openqa.selenium.support.ui.ExpectedConditions  
import tpo1.PresentationPage  
import tpo1.tests.AbstractTestSet  
  
abstract class AbstractMenuTestSet(name: String, page: PresentationPage) : AbstractTestSet(name, page) {  
 protected abstract val menu: WebElement  
  
 protected fun basicDropdownMenuTest(item: WebElement, expectedXpath: String) {  
 menu.click()  
 page.wait.until **{** ExpectedConditions.visibilityOf(item) **}** item.click()  
 assertExists(expectedXpath)  
  
 menu.click()  
 }  
  
 override fun initializeTest() {  
 menu.click()  
 }  
  
 override fun run() {  
 Thread.sleep(2500)  
 super.run()  
 }  
}

Тесты меню «Файл»:

package tpo1.tests.menu  
  
import org.openqa.selenium.WebElement  
import org.openqa.selenium.support.FindBy  
import tpo1.PresentationPage  
  
class FileMenuTests(page: PresentationPage) : AbstractMenuTestSet("File menu", page) {  
 @FindBy(id = "docs-file-menu")  
 override lateinit var menu: WebElement  
  
 @FindBy(xpath = "//span[contains(@aria-label,'Создать n')]")  
 lateinit var itemNew: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Создать копию c']")  
 lateinit var itemCreateCopy: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Скачать d']")  
 lateinit var itemDownloadAs: WebElement  
  
 @FindBy(xpath = "//span[contains(@aria-label,'Информация о файле b')]")  
 lateinit var itemInfo: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Язык l']")  
 lateinit var itemLanguage: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Настройки страницы g']")  
 lateinit var itemSettings: WebElement  
  
 @Test("New") fun newTest() {  
 basicEscapeTest(itemNew, "//span[contains(.,'Создать документ по шаблону')]")  
 }  
  
 @Test("Copy") fun copyTest() {  
 basicEscapeTest(itemCreateCopy, "//span[contains(.,'Копировать документ')]")  
 }  
  
 @Test("Download") fun downloadTest() {  
 basicDropdownMenuTest(itemDownloadAs, "//span[contains(.,'Microsoft PowerPoint (.pptx)')]")  
 }  
  
 @Test("Document info") fun docInfoTest() {  
 basicEscapeTest(itemInfo, "//td[contains(.,'Местоположение')]")  
 }  
  
 @Test("Language") fun langTest() {  
 basicDropdownMenuTest(itemLanguage, "//span[@lang='de']")  
 }  
  
 @Test("Settings") fun settingsTest() {  
 basicEscapeTest(itemSettings, "//div[contains(.,'Широкоэкранный (16:9)')]")  
 }  
}

Тесты меню «Вставка»:

package tpo1.tests.menu  
  
import org.openqa.selenium.WebElement  
import org.openqa.selenium.support.FindBy  
import tpo1.PresentationPage  
  
class InsertMenuTests(page: PresentationPage) : AbstractMenuTestSet("Insert menu", page) {  
 @FindBy(id = "docs-insert-menu")  
 override lateinit var menu: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Анимация a']")  
 lateinit var itemAnimation: WebElement  
  
 @FindBy(xpath = "//span[contains(@aria-label,'Оставить комментарий m')]")  
 lateinit var itemComment: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Номера слайдов e']")  
 lateinit var itemEnum: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Диаграмма h']")  
 lateinit var itemDiagram: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Таблица b']")  
 lateinit var itemTable: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Текстовое поле t']")  
 lateinit var itemTextField: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Word Art w']")  
 lateinit var itemWordArt: WebElement  
  
 @Test("Animation") fun animationTest() {  
 basicWindowTest(itemAnimation, "//div[@class='punch-animation-sidebar-header']//div[@title='Закрыть']/div")  
 }  
  
 @Test("Comment") fun commentTest() {  
 basicWindowTest(itemComment, "//div[@data-tooltip='Отмена отправки комментария']")  
 }  
  
 @Test("Enum") fun enumTest() {  
 basicEscapeTest(itemEnum, "//span[@class='modal-dialog-title-text'][contains(.,'Номера слайдов')]")  
 }  
  
 @Test("Diagram") fun diagramTest() {  
 basicEscapeTest(itemDiagram, "//span[text()='Линейчатая']")  
 }  
  
 @Test("Table") fun tableTest() {  
 basicEscapeTest(itemTable, "//div[text()='\u202A1 x 1\u202C']")  
 }  
  
 @Test("Text field") fun textFieldTest() {  
 menu.click()  
 itemTextField.click()  
 page.findById("textboxButton")?.let **{** button **->** if (button.getAttribute("aria-pressed") != "true") fail("Button doesn't press first time")  
 menu.click()  
 itemTextField.click()  
 if (button.getAttribute("aria-pressed") != "false") fail("Button doesn't press second time")  
 **}** ?: fail ("Doesn't open")  
 }  
  
 @Test("WordArt") fun wordArtTest() {  
 basicEscapeTest(itemWordArt, "//textarea[@class='sketchy-bubble-textarea']")  
 }  
}

Тесты меню «Слайд»:

package tpo1.tests.menu  
  
import org.openqa.selenium.By  
import org.openqa.selenium.WebElement  
import org.openqa.selenium.support.FindBy  
import tpo1.PresentationPage  
  
class SlideMenuTests(page: PresentationPage) : AbstractMenuTestSet("Slide menu", page) {  
 @FindBy(id = "punch-slide-menu")  
 override lateinit var menu: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Новый слайд n']")  
 lateinit var itemNewSlide: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Дублировать слайд d']")  
 lateinit var itemDuplicateSlide: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Удалить слайд e']")  
 lateinit var itemDeleteSlide: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Пропустить слайд s']")  
 lateinit var itemSkipSlide: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Изменить фон b']")  
 lateinit var itemBackgroundMenu: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Изменить тему t']")  
 lateinit var itemTheme: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Изменить переход r']")  
 lateinit var itemTransition: WebElement  
  
 @Test("Skip slide") fun slideSkipTest() {  
 menu.click()  
 itemSkipSlide.click()  
 page.driver.findElement(By.tagName("image")) ?: fail("Image not found")  
 menu.click()  
 itemSkipSlide.click()  
 }  
  
 @Test("Create and delete") fun createDeleteTest() {  
 menu.click()  
 itemNewSlide.click()  
  
 menu.click()  
 itemDuplicateSlide.click()  
  
 val slideCount = page.driver.findElements(By.className("punch-filmstrip-thumbnail")).count()  
 if (slideCount != 3) fail("Incorrect slide count")  
  
 for (i in 1..2) {  
 menu.click()  
 itemDeleteSlide.click()  
 }  
 }  
  
 @Test("Background menu") fun backgroundTest() {  
 basicWindowTest(itemBackgroundMenu, "//button[contains(@name,'punch-bg-close')]")  
 }  
  
 @Test("Theme menu") fun themeTest() {  
 basicWindowTest(itemTheme, "//div[@class='punch-theme-sidebar-header']//div[@title='Закрыть']/div")  
 }  
  
 @Test("Transition menu") fun transitionTest() {  
 basicWindowTest(itemTransition, "//div[@class='punch-animation-sidebar-header']//div[@title='Закрыть']/div")  
 }  
  
}

Тесты меню «Инструменты»:

package tpo1.tests.menu  
  
import org.openqa.selenium.WebElement  
import org.openqa.selenium.support.FindBy  
import tpo1.PresentationPage  
  
class ToolsMenuTests(page: PresentationPage) : AbstractMenuTestSet("Tools menu", page) {  
 @FindBy(id = "docs-tools-menu")  
 override lateinit var menu: WebElement  
  
 @FindBy(xpath = "//span[contains(@aria-label,'Настройки специальных возможностей c')]")  
 lateinit var itemAccessibility: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Настройки p']")  
 lateinit var itemSettings: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Расширенный поиск r']")  
 lateinit var itemSearch: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Проверка правописания s']")  
 lateinit var itemSpelling: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Словарь d']")  
 lateinit var itemDictionary: WebElement  
  
 @Test("Accessibility menu") fun accessibilityTest() {  
 basicWindowTest(itemAccessibility, "//div[@class='docs-material-gm-dialog-buttons']//span[text()='Отмена']")  
 }  
  
 @Test("Settings") fun settingsTest() {  
 basicWindowTest(itemSettings, "//span[@data-tooltip='Закрыть']")  
 }  
  
 @Test("Search") fun searchTest() {  
 basicWindowTest(itemSearch, "//div[@class='docs-explore-sidebar-title']//div[@aria-label='Закрыть']/div")  
 }  
  
 @Test("Spelling") fun spellingTest() {  
 basicEscapeTest(itemSpelling, "//span[@aria-label='Подчеркивать ошибки h']")  
 }  
  
 @Test("Dictionary") fun dictTest() {  
 basicWindowTest(itemDictionary, "//div[@class='docs-dictionary-sidebar docs-material']//div[@aria-label='Закрыть словарь']")  
 }  
}

Тесты меню «Вид»:

package tpo1.tests.menu  
  
import org.openqa.selenium.WebElement  
import org.openqa.selenium.support.FindBy  
import tpo1.PresentationPage  
  
class ViewMenuTests(page: PresentationPage) : AbstractMenuTestSet("View menu", page) {  
 @FindBy(id = "docs-view-menu")  
 override lateinit var menu: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Анимация a']")  
 lateinit var itemAnimation: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Привязка x']")  
 lateinit var itemBind: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='В виде сетки g']")  
 lateinit var itemGrid: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Направляющие d']")  
 lateinit var itemDirections: WebElement  
  
 @FindBy(xpath = "//span[@aria-label='Показать линейку r']")  
 lateinit var itemShowRuler: WebElement  
  
 @FindBy(xpath = "//span[contains(@aria-label,'Масштаб z')]")  
 lateinit var itemScale: WebElement  
  
 @Test("Animation") fun animationTest() {  
 basicWindowTest(itemAnimation, "//div[@class='punch-animation-sidebar-header']//div[@title='Закрыть']/div")  
 }  
  
 @Test("Bind") fun bindTest() {  
 basicEscapeTest(itemBind, "//span[text()='Сетка']")  
 }  
  
 @Test("Directions") fun directionsTest() {  
 basicEscapeTest(itemDirections, "//span[text()='Показать направляющие']")  
 }  
  
 @Test("Grid") fun gridTest() {  
 menu.click()  
 itemGrid.click()  
 val grid = page.findById("grid-view-tab")  
 if (grid == null || grid.getAttribute("aria-selected") != "true") {  
 fail("Doesn't work")  
 }  
  
 menu.click()  
 itemGrid.click()  
 }  
  
 @Test("Ruler") fun rulerTest() {  
 val ruler = page.findById("sketchy-horizontal-ruler")!!  
  
 menu.click()  
 itemShowRuler.click()  
 if (ruler.isDisplayed) fail("Ruler doesn't hide")  
  
 menu.click()  
 itemShowRuler.click()  
 if (!ruler.isDisplayed) fail("Ruler doesn't show")  
 }  
  
 @Test("Scale") fun scaleTest() {  
 basicEscapeTest(itemScale, "//span[text()='Увеличить']")  
 }  
}

Тесты панели инструментов:

package tpo1.tests  
  
import org.openqa.selenium.WebElement  
import org.openqa.selenium.support.FindBy  
import tpo1.PresentationPage  
  
class BarTests(page: PresentationPage) : AbstractTestSet("Bar", page) {  
 @FindBy(id = "insertImageMenuButton")  
 lateinit var insertImageMenuButton: WebElement  
  
 @FindBy(id = "lineMenuButton")  
 lateinit var lineMenuButton: WebElement  
  
 @FindBy(id = "zoomButtonDropdown")  
 lateinit var zoomButtonDropdown: WebElement  
  
 @FindBy(id = "shapeButton")  
 lateinit var shapeButton: WebElement  
  
 @FindBy(id = "slideThemeButton")  
 lateinit var slideThemeButton: WebElement  
  
 @FindBy(id = "slideTransitionButton")  
 lateinit var slideTransitionButton: WebElement  
  
  
 override fun initializeTest() { }  
  
  
 @Test("Insert image") fun insertImagesest() {  
 basicEscapeTest(insertImageMenuButton, "//span[text()='Загрузить с компьютера']")  
 }  
  
 @Test("Line") fun lineTest() {  
 basicEscapeTest(lineMenuButton, "//span[text()='Линия']")  
 }  
  
 @Test("Zoom dropdown") fun zoomButtonDropdownTest() {  
 basicEscapeTest(zoomButtonDropdown, "//span[@aria-label='50% 5']")  
 }  
  
 @Test("Shape") fun shapeTest() {  
 basicEscapeTest(shapeButton, "//span[text()='Фигуры']")  
 }  
  
 @Test("Theme") fun themeTest() {  
 basicWindowTest(slideThemeButton, "//div[@class='punch-theme-sidebar-header']//div[@title='Закрыть']/div")  
 }  
  
 @Test("Transition") fun transotionTest() {  
 basicWindowTest(slideTransitionButton, "//div[@class='punch-animation-sidebar-header']//div[@title='Закрыть']/div")  
 }  
}

Тесты «скрытой» панели инструментов, которая появляется при выборе элемента слайда:

package tpo1.tests  
  
import org.openqa.selenium.WebElement  
import org.openqa.selenium.support.FindBy  
import tpo1.PresentationPage  
  
class HiddenBarTests(page: PresentationPage) : AbstractTestSet("Hidden bar", page) {  
 @FindBy(id = "canvas")  
 lateinit var canvas: WebElement  
  
 @FindBy(id = "textColorButton")  
 lateinit var textColorButton: WebElement  
  
 @FindBy(id = "fillColorMenuButton")  
 lateinit var fillColorMenuButton: WebElement  
  
 @FindBy(id = "docs-font-family")  
 lateinit var docsFontFamilyButton: WebElement  
  
 @FindBy(id = "fontSizeSelect")  
 lateinit var fontSizeSelect: WebElement  
  
 @FindBy(id = "lineColorMenuButton")  
 lateinit var lineColorMenuButton: WebElement  
  
 @FindBy(id = "lineDashingMenuButton")  
 lateinit var lineDashingMenuButton: WebElement  
  
 @FindBy(id = "lineWidthMenuButton")  
 lateinit var lineWidthMenuButton: WebElement  
  
  
  
 private fun showHiddenBar() {  
 page.actions.moveToElement(canvas).click().perform()  
 }  
  
 override fun initializeTest() {  
 showHiddenBar()  
 }  
  
  
  
 @Test("Text color") fun textColorTest() {  
 basicEscapeTest(textColorButton, "//div[@style='user-select: none; visibility: visible; left: 817.6px; top: 97.3667px;']")  
 }  
  
 @Test("Fill color") fun fillColorTest() {  
 basicEscapeTest(fillColorMenuButton, "//div[text()='Один цвет']")  
 }  
  
 @Test("Docs font family") fun docsFontFamilyTest() {  
 basicEscapeTest(docsFontFamilyButton, "//div[text()='Другие шрифты']")  
 }  
  
 @Test("Font size selection") fun fontSizeTest() {  
 basicEscapeTest(fontSizeSelect, "//div[text()='96']")  
 }  
  
 @Test("Line color") fun lineColorTest() {  
 basicEscapeTest(lineColorMenuButton, "//div[@style='user-select: none; visibility: visible; left: 450.4px; top: 97.3667px;']")  
 }  
  
 @Test("Line dashing") fun lineDashingTest() {  
 basicEscapeTest(lineDashingMenuButton, "//div[@aria-label='Пунктир: Сплошная']")  
 }  
  
 @Test("Line width") fun lineWidthTest() {  
 basicEscapeTest(lineWidthMenuButton, "//div[text()='1 пикс.']")  
 }  
}

Тесты контекстного меню:

package tpo1.tests  
  
import org.openqa.selenium.WebElement  
import org.openqa.selenium.support.FindBy  
import tpo1.PresentationPage  
  
class RightClickTests(page: PresentationPage) : AbstractTestSet("Right click", page) {  
 @FindBy(id = "filmstrip")  
 lateinit var filmstrip: WebElement  
  
 @FindBy(xpath = "//span[text()='Изменить фон']")  
 lateinit var itemChangeBackground: WebElement  
  
 @FindBy(xpath = "//span[text()='Выбрать макет']")  
 lateinit var itemLayout: WebElement  
  
 @FindBy(xpath = "//span[text()='Изменить тему']")  
 lateinit var itemTheme: WebElement  
  
 @FindBy(xpath = "//span[text()='Изменить переход']")  
 lateinit var itemTransition: WebElement  
  
 @FindBy(id = "newSlideButton")  
 lateinit var itemNewSlide: WebElement  
  
  
  
 private fun openContextMenu() {  
 page.actions.contextClick(filmstrip).perform()  
 }  
  
 override fun initializeTest() {  
 openContextMenu()  
 }  
  
  
  
 @Test("Background") fun backgroundTest() {  
 basicWindowTest(itemChangeBackground, "//span[@aria-label='Закрыть']")  
 }  
  
 @Test("Layout") fun layoutTest() {  
 basicEscapeTest(itemLayout, "//div[text()='Заголовок и текст']")  
 }  
  
 @Test("Theme") fun themeTest() {  
 basicEscapeTest(itemTheme, "//div[@class='punch-theme-sidebar-header']//div[@title='Закрыть']/div")  
 }  
  
 @Test("Transition") fun transitionTest() {  
 basicWindowTest(itemTransition, "//div[@class='punch-animation-sidebar-header']//div[@title='Закрыть']/div")  
 }  
  
 @Test("New slide delete") fun newSlideDeleteTest() {  
 val firstSlide = page.findByXpath("//\*[@id='filmstrip']//\*[contains(@class, 'punch-filmstrip-thumbnail')][1]")  
 page.actions.contextClick(firstSlide).perform()  
 page.findByXpath("//span[text()='Новый слайд']")?.click() ?: fail("Failed to create new slide")  
 if (filmstrip.text != "12") fail("Incorrect filmstrip text (expected: \"12\", actual: \"${filmstrip.text}\"")  
  
 val secondSlide = page.findByXpath("//\*[@id='filmstrip']//\*[contains(@class, 'punch-filmstrip-thumbnail')][2]")  
 page.actions.contextClick(secondSlide).perform()  
 page.findByXpath("//span[text()='Удалить']")?.click() ?: fail("Failed to delete slide")  
 if (filmstrip.text != "1") fail("Incorrect filmstrip text (expected: \"1\", actual: \"${filmstrip.text}\"")  
 }  
  
 @Test("Duplicated slide delete") fun duplicatedSlideDeleteTest() {  
 val firstSlide = page.findByXpath("//\*[@id='filmstrip']//\*[contains(@class, 'punch-filmstrip-thumbnail')][1]")  
 page.actions.contextClick(firstSlide).perform()  
 page.findByXpath("//span[text()='Дублировать слайд']")?.click() ?: fail("Failed to duplicate slide")  
 if (filmstrip.text != "12") fail("Incorrect filmstrip text (expected: \"12\", actual: \"${filmstrip.text}\"")  
  
 val secondSlide = page.findByXpath("//\*[@id='filmstrip']//\*[contains(@class, 'punch-filmstrip-thumbnail')][2]")  
 page.actions.contextClick(secondSlide).perform()  
 page.findByXpath("//span[text()='Удалить']")?.click() ?: fail("Failed to delete slide")  
 if (filmstrip.text != "1") fail("Incorrect filmstrip text (expected: \"1\", actual: \"${filmstrip.text}\"")  
 }  
}

**Выводы**

В ходе выполнения этой работы было проведено тестирование веб-сайта при помощи Selenium. Для этого был использован драйвер браузера Firefox и программа на языке Kotlin, открывающая меню и проверяющая наличие необходимых элементов при нажатии на его соответствующие пункты.

Также для удобства был разработан простейший тестовый фреймворк с использованием аннотаций и рефлексии. Он позволяет запускать все методы, отмеченные аннотацией Test, и выводит в консоль результаты наборов тестов (класс AbstractTestSet) и каждого отдельного теста в них.