Факультет программной инженерии и компьютерной техники

Тестирование программного обеспечения

Лабораторная работа №3

Вариант №23195: Ulmarts.ru. Продажа компьютеров и комплектующих, бытовой техники. - http://www.ulmarts.ru

Преподаватель: Яркеев Александр Сергеевич

Выполнил: Васькин Алексей Андреевич P33112

Санкт-Петербург
2021

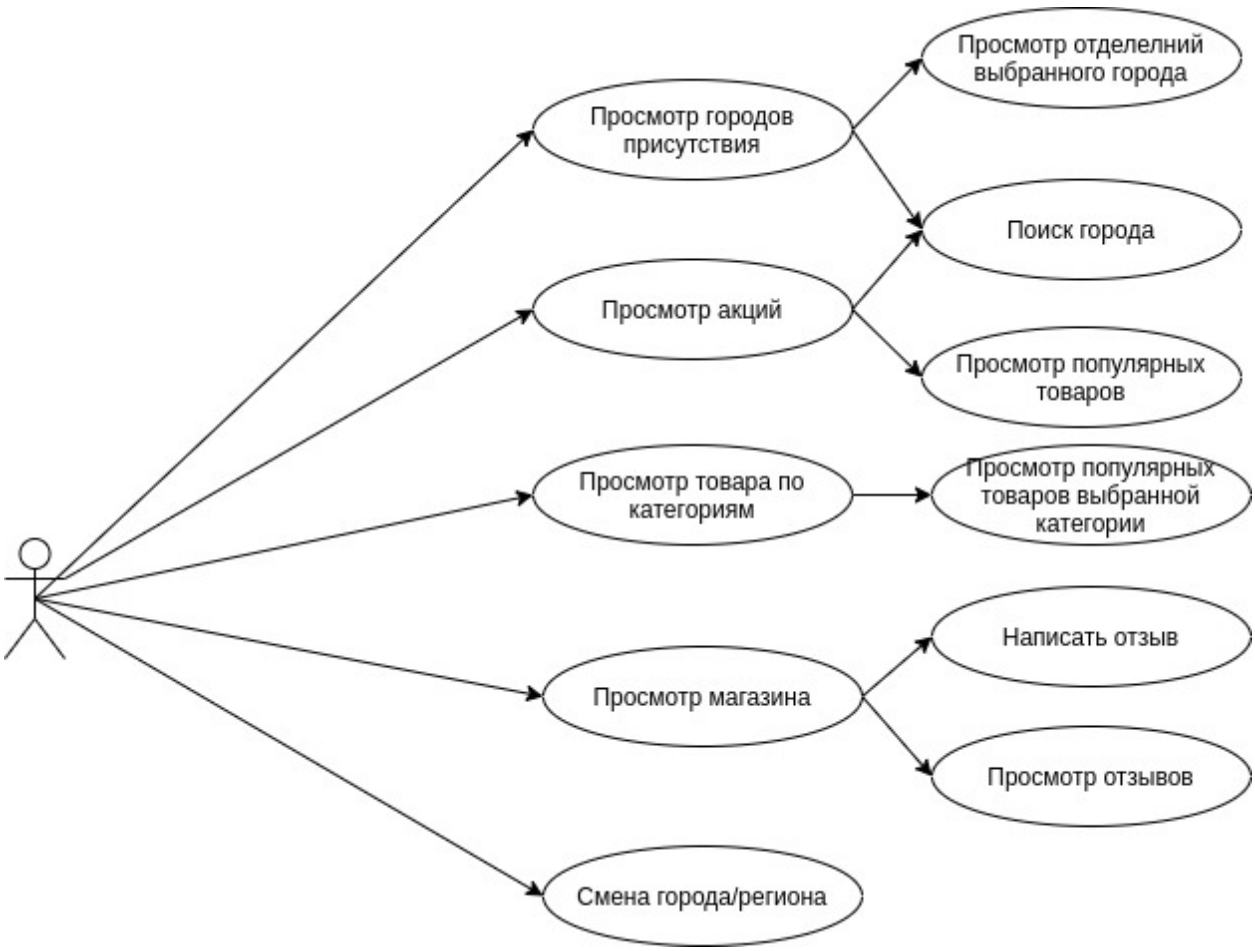
Задание

Сформировать варианты использования, разработать на их основе тестовое покрытие покрытие и провести функциональное тестирование интерфейса сайта (в соответствии с вариантом).

Требования к выполнению работы:

1. Тестовое покрытие должно быть сформировано на основании набора прецедентов использования сайта.
2. Тестирование должно осуществляться автоматически - с помощью системы автоматизированного тестирования Selenium.
3. Шаблоны тестов должны формироваться при помощи Selenium IDE и исполняться при помощи Selenium RC в браузерах Firefox и Chrome.
4. Предполагается, что тестируемый сайт использует динамическую генерацию элементов на странице, т.е. выбор элемента в DOM должен осуществляться не на основании его ID, а с помощью XPath.

UseCases



Описание тестового покрытия

Имя прецедента:	Зайти в раздел «О проекте»
Начальные условия:	Открыта любая страница сайта
Ход выполнения:	1. Нажать на текст с названием раздела «О проекте» в заголовке страницы
Критерий прохождения:	Произошёл переход на страницу «О проекте», в хлебных крошках написан этот раздел
Критерий непрохождения:	Не произошёл переход на страницу «О проекте»

Имя прецедента:	Смена региона/города
Начальные условия:	Открыта любая страница сайта
Ход выполнения:	1 Нажать на кнопку с названием текущего региона вверху страницы 2. Нажать на кнопку с нужным регионом/городом
Критерий прохождения:	Открылась страница региона/города, в хлебных крошках написан этот город
Критерий непрохождения:	Не произошёл переход на страницу выбранного региона/города

Имя прецедента:	Зайти в раздел «Телефоны» и выбрать один из популярных
Начальные условия:	Открыта любая страница сайта
Ход выполнения:	1. Нажать на кнопку «Каталог товаров» 2. В появившемся дропдауне нажать на ссылку на раздел «Телефоны» 3. Прокрутить страницу до «Популярные товары из категории Телефоны» 4. Нажать на ссылку под фото одного из популярных телефонов
Критерий прохождения:	Произошёл переход на страницу выбранного телефона
Критерий непрохождения:	Не произошёл переход на страницу выбранного телефона

Имя прецедента:	Найти скидки и акции в городе «Москва»
Начальные условия:	Открыта любая страница сайта
Ход выполнения:	1. Перейти в раздел «Скидки и акции» 2. Ввести «мо» в строку поиска 3. Выбрать «Москва» из сократившегося списка городов
Критерий прохождения:	При введении «мо» поиске остаются только города «Москва», «Ломоносов», на страницу «Москва» можно перейти после поиска
Критерий непрохождения:	Список не сокращается после ввода, на страницу «Москва» нельзя перейти

Код

https://github.com/reddist/tpo_lab3

Чек-лист тестирования

Главная страница

1) проверка title страницы

2) пункты меню на главной

3) проверка смены города (3 штуки)

Телефоны

4) проверять, что выдаёт популярные телефоны, можно перейти на страницу какого-то из них

Скидки и акции

5) скидки и акции -> поиск города

Вывод

Выполнение данной лабораторной было сопряжено с многими трудностями. Во-первых, фреймворк Selenium произвёл на меня в преимущественно неприятное впечатление из-за отсутствия достаточно очевидных и нужных вещей, таких как метод задания атрибутов элементов, фабрики генерации драйверов, а также механизма для выполнения тестов во всех браузерах сразу: все эти вещи пришлось писать самому, чего не ожидаешь от готового решения для тестирования. Во-вторых, не с лучшей стороны показал себя в данном случае JUnit. В JUnit4 была возможность создавать параметризованные тестовые классы, но в JUnit5 эта фича отсутствует, и вместо неё предлагает использовать параметры на уровне методов, что в большинстве случаев действительно может быть более полезно, но не в текущем сценарии, когда драйвер для браузера хочется иметь на

уровне класса, ввиду его переиспользования. Приходится либо откатывать к JUnit4, либо пользоваться пакетом Vintage, что чревато ошибками из-за миксования аннотаций 4 и 5 версий, которые несовместимы между собой. Из-за вышеописанных причин, желания уменьшить дублирование кода, и при всём этом применять паттерн PageObject, рекомендуемый в такого рода тестах, пришлось разрабатывать целую архитектуру для тестирования, что не очень-то и вяжется с высокой простотой и скоростью написания тестов, и уж точно дольше ручного тестирования. А если тесты будет писать отдельная команда, не принимающая участие в разработке страницы, то также много времени уйдёт на изучение атрибутов элементов и js-кода для получения корректного и устойчивого x-path. При использовании же современных реактивных фреймворков, где любая визуальный элемент при разработке часто абстрагирован от DOM и генерируется фреймворком, написание этих тестов становится сущим адом с огромным количеством переменных, классов и скриптов, которые могут сломать тесты в два счёта. Я считаю, что написание подобных тестов может быть полезно, легко и быстро только в случаях, когда непосредственно разработчик страницы создаёт их.