# **SELENIDE**

1. Написать тесты, покрывающие функциональность логина <https://the-internet.herokuapp.com/login> (минимум 4 теста)

Обязательное использование Selenide, все тесты внутри одного класса, для каждого теста должен открываться новый браузер, использование Page Object, DDT, Before/After методы

1. Расширяем проект, добавляем тесты для выпадающих списков - <https://the-internet.herokuapp.com/dropdown>. Первый тест, выбираем опцию по тексту Option 1, после выбора проверяем, что выбрано значение “1” (НЕ ТЕКСТ!!!).

Второй тест, выбираем опцию по значению “2”, после проверяем, что выбрана опция с текстом “Option 2”.

1. Написать тест с наведение мыши для <https://the-internet.herokuapp.com/hovers>. Один и тот же тест написать дважды: первая реализация через цикл с использованием HashMap, вторая как DDT.
2. \*\*Начинаем скачивать файлы, пишем тест (DDT/no DDT) для <https://the-internet.herokuapp.com/download>. Задача – скачать каждый из файлов и проверить, что он существует. ☺
3. \*\*Написать тесты для работы с алертами <https://the-internet.herokuapp.com/javascript_alerts>. Ожидаю 5 тестов – нажал, что-то ввёл – увидел сообщение на странице. ☺
4. \*\*\*Написать тесты для сортировки <https://the-internet.herokuapp.com/tables>. Проверяем сортировку только по Last Name, First Name. Первый тест для примера 1, можно использовать класс, который указывает на изменяющуюся сортировку. Второй тест для примера 2 и необходимо посредственно проверять каждую строчку. ☺

# **REPORTING**

1. Визуализируем результаты тестирования с помощью ReportNG.
2. Визуализируем результаты тестирования с помощью Allure reporting framework. (Обязательно использование Features, Stories, TestCaseID).