**УДК 004.054**

# РАЗРАБОТКА БИБЛИОТЕКИ ДЛЯ АВТОМАТИЗАЦИИ ТЕСТИРОВАНИЯ ПРОИЗВОДИТЕЛЬНОСТИ КЛИЕНТСКИХ ВЕБ-ПРИЛОЖЕНИЙ

**Савин Г.Е.** (Университет ИТМО, Санкт-Петербург)
**Научный руководитель – ассистент, Яркеев А.С.**
(Университет ИТМО, Санкт-Петербург)

**Аннотация.**
В работе рассматривается подход к автоматизации тестирования производительности пользовательских интерфейсов веб-приложений, основная часть работы которых связана с выполнением JavaScript.

**Введение.**
Современные пользовательские компьютеры обладают относительно большой производительностью, что позволяет не задумываться о том, как быстро работает интерфейс при его разработке. Возникающие проблемы почти всегда хорошо решаются при помощи встроенного в браузер профилировщика производительности. Существуют решения, которые хорошо подходят для автоматизации тестирования классических веб-приложений, основная часть работы которых связана с отрисовкой DOM и его изменениями.

Однако, существует класс приложений, для которых также крайне важна хорошая производительность, и ее необходимо постоянно отслеживать, при этом, основная часть работы этих приложений связана с исполнением JavaScript. Разработчикам таких приложений приходится придумывать свои способы автоматизации тестирования производительности.

**Основная часть.**
Перед реализацией были проанализированы существующие решения для автоматизации тестирования производительности, однако все они опираются на метрики Lighthouse, которые не очень показательны в случае, когда существенно большую часть работы приложения составляет постоянное исполнение JavaScript.

За основу реализованного решения выбрана библиотека Puppeteer – она предоставляет API для управления браузером Google Chrome поверх протокола DevTools. Предоставляемое API позволяет производить профилирование во время выполнения в браузере каких-либо действий. Сохраняемый после профилирования отчет не позволяет получить такие характеристики как среднее количество кадров в секунду и процент от количества времени, затраченного на выполнение JavaScript, поэтому необходимо выполнить предварительную обработку данных из отчета. С использованием полученных метрик можно составить тестовые сценарии, например, при помощи библиотеки jest. По результатам выполнения тестов формируется отчет в формате JSON. Также было реализовано веб-приложение для просмотра и анализа результатов полученных отчетов.

При написании тестовых сценариев важно учитывать производительность той вычислительной машины, на которых они будут выполняться. Также возможно оформление тестовых сценариев в виде отдельных URL на каждый сценарий вместе с реализацией сценариев внутри приложения отдельно от библиотеки – в таком случае библиотеке нужно только перейти по URL, запустить профилирование, собрать метрики, проанализировать их и сверить с заданными предельными значениями, если необходимо. Сформированные отчеты можно использовать для анализа динамики производительности, однако важно понимать, что существует некоторый разброс в значениях между запусками тестов на одном и том же окружении.

**Выводы.**

Данная библиотека была применена для автоматизации тестирования веб-интерфейса приложения для построения графиков, основная часть работы которой связана с отрисовкой на канвасе. Запуск тестов был добавлен в систему непрерывной интеграции teamcity и проходил внутри Docker-контейнера. Отчеты сохранялись в общую историю результатов тестирования, на их основании можно оценить динамику производительности приложения.