**Проект №8 — Дополнительные инструменты для тестирования.**

Привет, участник Школы 21! В этом проекте мы поговорим о том, как отслеживать трафик и логи приложений и для чего это нам может понадобиться, как пользоваться терминалом, изучим базовые команды. Узнаем, что такое непрерывная интеграция и непрерывное развёртывание, и как устроен релизный цикл.

**Instructions**

Напоминаем, что все отчёты по результатам выполнения заданий тебе нужно оформлять в файлах с расширением .md. Если они уже созданы, то пересоздавать или удалять их не нужно (просто отредактируй этот файл). Все созданные отчёты и файлы тебе нужно будет загрузить в папку src/ в корне проекта (обязательно в ветку *develop*).

**Contents**

1. [Chapter I](https://edu.21-school.ru/project/62671/task#chapter-i)  
   1.1. [Общая инструкция](https://edu.21-school.ru/project/62671/task#%D0%BE%D0%B1%D1%89%D0%B0%D1%8F-%D0%B8%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D1%8F)
2. [Chapter II](https://edu.21-school.ru/project/62671/task#chapter-ii)  
   2.1. [Снифферы](https://edu.21-school.ru/project/62671/task#%D1%81%D0%BD%D0%B8%D1%84%D1%84%D0%B5%D1%80%D1%8B)  
   2.2. [DevTools](https://edu.21-school.ru/project/62671/task#devtools)  
   2.3. [Задание №1](https://edu.21-school.ru/project/62671/task#%D0%B7%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-1-%D0%B2%D0%BE%D0%B7%D0%BC%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D0%B8-devtools)  
   2.4. [Задание №2](https://edu.21-school.ru/project/62671/task#%D0%B7%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-2-devtools-%D0%B8-fiddler)
3. [Chapter III](https://edu.21-school.ru/project/62671/task#chapter-iii)  
   3.1. [Логи](https://edu.21-school.ru/project/62671/task#%D0%BB%D0%BE%D0%B3%D0%B8)  
   3.2. [Задание №3](https://edu.21-school.ru/project/62671/task#%D0%B7%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-3-%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B8-%D0%B8-%D1%82%D0%B8%D0%BF%D1%8B-%D0%BB%D0%BE%D0%B3%D0%BE%D0%B2)  
   3.3. [Иструменты](https://edu.21-school.ru/project/62671/task#%D0%B8%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D1%8B)  
   3.4. [Журнал событий](https://edu.21-school.ru/project/62671/task#%D0%B6%D1%83%D1%80%D0%BD%D0%B0%D0%BB-%D1%81%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9)  
   3.5. [Задание №4](https://edu.21-school.ru/project/62671/task#%D0%B7%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-4-%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0-%D1%81-%D0%B6%D1%83%D1%80%D0%BD%D0%B0%D0%BB%D0%BE%D0%BC-%D1%81%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9)
4. [Chapter IV](https://edu.21-school.ru/project/62671/task#chapter-iv)  
   4.1. [Задание №5](https://edu.21-school.ru/project/62671/task#%D0%B7%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-5-powershell)  
   4.2. [Задание №6](https://edu.21-school.ru/project/62671/task#%D0%B7%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-6-webminal)
5. [Chapter V](https://edu.21-school.ru/project/62671/task#chapter-v)  
   5.1. [Релизный цикл, CI, CD](https://edu.21-school.ru/project/62671/task#%D1%80%D0%B5%D0%BB%D0%B8%D0%B7%D0%BD%D1%8B%D0%B9-%D1%86%D0%B8%D0%BA%D0%BB-ci-cd)  
   5.2. [GitLab CI](https://edu.21-school.ru/project/62671/task#gitlab-ci)  
   5.3. [Задание №7](https://edu.21-school.ru/project/62671/task#%D0%B7%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-7-docker-%D0%B8-kubernetes)

**Chapter I**

**Общая инструкция**

Методология Школы 21 может быть не похожа на тот образовательный опыт, который случался с тобой ранее. Её отличает высокий уровень автономии: у тебя есть задача, ты должен её выполнить. По большей части тебе нужно будет самому добывать знания для её решения. Второй важный момент — это peer-to-peer обучение. В образовательном процессе нет менторов и экспертов, перед которыми ты защищаешь свой результат. Ты это делаешь перед таким же учащимися, как и ты сам. У них есть чек-лист, который поможет им качественно выполнить приемку вашей работы.

Роль Школы 21 заключается в том, чтобы обеспечить через последовательность заданий и оптимальный уровень поддержки такую траекторию обучения, при которой ты не только освоишь hard skills, но и научишься самообучаться.

* Не доверяй слухам и предположениям о том, как должно быть оформлено ваше решение. Этот документ является единственным источником, к которому стоит обращаться по большинству вопросов;
* твое решение будет оцениваться другими учащимися;
* подлежат оцениванию только те файлы, которые ты выложил в GIT (ветка develop, папка src);
* в твоей папке не должно быть лишних файлов — только те, что были указаны в задании;
* не забывай, что у вас есть доступ к интернету и поисковым системам;
* обсуждение заданий можно вести и в Rocket.Chat;
* будь внимателен к примерам, указанным в этом документе — они могут иметь важные детали, которые не были оговорены другим способом;
* и да пребудет с тобой Сила!

**Chapter II**

**Снифферы**

**Сниффер** (от английского sniff — нюхать) представляет из себя программное обеспечения для анализа входящего и исходящего трафика, своего и чужого, в рамках сети.

Снифферы бывают:

* *Коммерческие* (применяются для тестирования, обнаружения неполадок)
* *Вредоносные* (используются для нанесения ущерба, кражи данных)
* *Кратковременные* (применяются единожды для какой-либо задачи)
* *Постоянные* (используются с целью постоянного мониторинга)

Одним из самых популярных снифферов можно назвать WireShark. Однако для тестирования веб-приложений часто используют Fiddler, который позволяет:

* Анализировать трафик;
* Осуществлять сбор данных;
* Подменять HTTP-пакеты;
* Подменять POST-данные;

**DevTools**

**DevTools** являются одним из ключевых инструментов тестирования графического интерфейса, а также активно используются при проверке интеграции клиента с сервером. Давай рассмотрим их функционал подробнее!😇

**Инструменты разработчика** (от англ. "development tools" или сокращённо "DevTools") ─ это программы, позволяющие создавать, тестировать и отлаживать (debug) программное обеспечение.

Современные браузеры имеют встроенные инструменты разработчика, позволяющие просмотреть исходный код сайта. Отдельно устанавливать их не требуется. С их помощью можно просматривать и изменять HTML-, CSS- и Javascript-код. Также можно проверить сетевой трафик, потребляемый сайтом, его быстродействие и множество других параметров. DevTools позволяет редактировать страницы на лету и быстро диагностировать проблемы.

**Задание №1. Возможности DevTools**

В этом задании вам потребуется хорошо изучить *Chrome DevTools*! Создайте файл exercise1.md, в котором:

* укажите горячие клавиши, которыми можно открыть Chrome DevTools;
* **подробно** опишите все возможности, которые предоставляет Chrome DevTools.

**Задание №2. DevTools и Fiddler**

Скачайте и установите себе на компьютеры [Firefox](https://www.mozilla.org/ru/firefox/new/). Перейдите на официальный сайт [СберМегаМаркета](https://sbermegamarket.ru/). При помощи вкладки "Console" (Консоль) в DevTools отловите ошибки, которые сможете найти в веб-приложении. Найденные проблемы (ошибки, которые появляются в консоли, баги интерфеса, баги в логике работы приложения, если они будут) опишите в виде баг-репортов в файле exercise2.reports.md. Описывать одни и те же баги, которые повторяются с некоторой периодичностью, не нужно :)

С помощью вкладки "Network" (Сеть) в DevTools в файле exercise2.requests.md укажите:

* Какой запрос(-ы) отправляется на сервер для получения списка типов товаров в шторке "Каталог"?
* Какой запрос(-ы) отправляется на сервер при использовании поиска товаров в каталоге?
* Какой запрос(-ы) отправляется на сервер при поиске региона или города в модалке выбора вашего региона?

Скачайте и установите Fiddler Everywhere или Fiddler Classic. Настройте фильтрацию трафика по хосту СберМегаМаркет. Сделайте что-нибудь в веб-приложении и сделайте скрин отловленных запросов, назовите его exercise2.png.

**Chapter III**

Если представить тестируемую систему как живой организм, то дефект — это болезнь, а логи — это визуализация событий, которые произошли с этим организмом перед появлением болезни. Если логи хорошо организованы, то в них еще и будут результаты "анализов" организма, в котором произошел сбой. Зная всю эту информацию, тестировщики, как врачи, смогут поставить правильный диагноз.

**Логи**

Если в работе сервера, компьютера или программного обеспечения возникает неизвестная ошибка, то в первую очередь смотрят логи. **Лог** — текстовый файл с информацией о действиях программного обеспечения или пользователей, который хранится на компьютере или сервере. Это хронология событий и их источников, ошибок и причин, по которым они произошли. Читать и анализировать логи можно с помощью специального ПО. Соответственно, **логированием** называют запись логов.

**Задание №3. Уровни и типы логов**

Создайте файл exercise3.md и опишите в нём:

* Какие существуют типы и уровни логов?
* Какое расширение имеет файл с логами?
* Какие программы используются для просмотра логов?

**Инструменты**

Сбор, хранение и анализ логов выполняется с помощью специальных инструментов. Очень часто совместно используют Logstash, Elasticsearch и Kibana.

* **Logstash** — приложение для работы с большими объемами данных, собирает информацию из разных источников и переводит ее в удобный формат.
* **Elasticsearch** — система для поиска информации. Помогает быстро найти нужные строки в файлах хранения.
* **Kibana** — плагин визуализации данных и аналитики в Elasticsearch. Помогает обрабатывать информацию, находить в ней закономерности и слабые места.

**Журнал событий**

**Журнал событий (Event Log)** — в Windows стандартный способ записи и централизованного хранения логов для приложений и операционной системы. Нам он может понадобиться для того, чтобы понимать, что происходит с системой, найти причину сбоев, определив некорректно работающие софт или устройства. Выглядит он как небольшая подборка логов, которые система записывает в ходе работы компьютера. Эти журналы являются обычными текстовыми файлами, записанными в формате XML.

**Linux** — семейство Unix-подобных операционных систем на базе ядра Linux. Все типы сообщений, которые создаются при работе ОС Linux и её приложений, регистрируются в различных файлах журналов. В Linux используются специальное программное обеспечение, файлы и директории для хранения лог файлов. Для локализации и устранения ошибок необходимо знать, в каких файлах находятся логи, и какими запросами доставать информацию из этих логов.

При запуске определенной программы на компьютере он отправляет сообщение в журнал событий. Каждый осуществляемый доступ, изменения конфигурации безопасности, сбой операционной системы, аппаратных компонентов или драйверов оставляет след в том или ином логе событий.

**Задание №4. Работа с журналом событий**

Создайте файл exercise4.md, в котором укажите ОС, на которой вы выполняете задание. Если у вас *Windows*, то выполняйте следующие пункты:

* Войдите в журнал событий. Если по какой-то причине запись журналов событий на вашем ПК отключена, то найдите в интернете инструкцию как включить записи событий.
* В файле exercise4.md опишите инструкцию для открытия журнала событий Windows.
* Добавьте в файл exercise4.md описание следующих журналов: Система, Приложение, Безопасность. В этом же файле укажите, какую информацию в этих журналах можно найти.
* Какая команда в cmd отображает текущую таблицу ARP для всех интерфейсов, присутcтвующих на компьютере? Сделайте запуск этой команды в cmd и скрин, который назовите exercise4.1.png.
* Какая команда в cmd поможет вывести список установленных драйверов устройств? Сделайте запуск этой команды в cmd и скрин, назовите скрин exercise4.2.png.

Если же у вас *Unix-подобная ОС*, то выполняйте следующие пункты:

* В файл exercise4.md внесите информацию о том, что такое логи (журнал сервера) в Linux.
* В том же документе создайте таблицу из двух столбцов. В левом — вопрос, в правом — ответ. Внесите в таблицу следующие пары вопросов и ответов:
  + В каком файле смотреть логи неудачных попыток авторизации?
  + Что делает команда ls /var/log?
  + Какой командой посмотреть логи журнала сообщений от ядра в реальном времени?
  + Какая команда покажет, кто из пользователей сейчас залогинен в системе и когда он зашел? Сделайте запуск этой команды в терминале и скрин, который назовите exercise4.1.png.
  + Какая команда дает понять, когда пользователь заходил в систему и сколько времени в ней находился? Сделайте запуск этой команды в терминале и скрин, который назовите exercise4.2.png.

**Chapter IV**

В этой главе вам предстоит поработать с терминалами Windows и Linux.

**Задание №5. PowerShell**

* Создайте файл exercise5.md и внесите в него ответы на следующие вопросы:
  + Что такое PowerShell?
  + Какие возможности имеет PowerShell?
  + Как открыть журнал событий в PowerShell?
  + Чем cmd отличается от PowerShell?
  + Какой командой в PowerShell можно просмотреть список доступных журналов событий? Запустите эту команду в PowerShell и сделайте скрин, который назовите exercise5.1.png.
  + Какой командой можно узнать текущую версию PowerShell? Запустите эту команду в PowerShell и сделайте скрин, который назовите exercise5.2.png.
* Создайте файл ps.txt (файл должен быть именно в формате .txt) в C:\Users\<ваш логин в системе> с содержимым: This is PS file. Запустите команду из PowerShell для чтения содержимого этого файла. Сделайте скрин запуска команды, который назовите exercise5.3.png.

**Задание №6. Webminal**

**Webminal** — онлайновый Linux-терминал для обучения Linux-командам для начинающих. Вам потребуется поработать с Linux через данный терминал.

Создайте аккаунт в <https://www.webminal.org/register/>. Вам нужно будет пройти 5 уроков (Lesson1 – Lesson5) и в каждом уроке нужно будет сделать скрин после запуска определенной команды. Эти команды будут указаны в каждом пункте задания.

* Авторизуйтесь в терминале (логин и пароль такие же, как на платформе).
* *Lesson1*. Сделайте скрин результата команды ls -R, назовите скрин exercise6.1.png.
* *Lesson2*. Cделайте скрин запуска последних двух команд урока: stat hello.txt и stat dir1, назовите скрин exercise6.2.png.
* *Lesson3*. После запуска последней команды урока rmdir dir50 сделайте запуск команды ls и скрин результата, назовите скрин exercise6.3.png.
* *Lesson4*. Сделайте скрин запуска последней команды урока time ls -l, назовите скрин exercise6.4.png.
* *Lesson5*. Сделайте скрин запуска последней команды урока diff3 hello new.txt linux.txt, назовите скрин exercise6.5.png.

Создайте файл exercise6.md и опишите в нём, что делают команды grep и diff.

*P.S.: все уроки должен выполнить каждый участник команды. Важно не пропускать ни одной команды и запускать их в указанной последовательности, все уроки требуется выполнять по порядку. После выполнения всех уроков сравните полученные результаты. Если у всех всё совпадает - загрузите выполненное задание.*

**Chapter V**

**Релизный цикл, CI, CD**

От момента принятия решения о необходимости дополнения/изменения функционала до непосредственного релиза должен произойти ряд процессов. Этот путь от идеи до релиза и называется **релизным циклом**. Мы не будет подробно останавливаться на его шагах, так как они сильно зависят от каждой отдельной компании и даже от проектов внутри этих компаний, но все же упомянем пару основных моментов.

Предположим, разработчики написали код. Остается доставить его до пользователя в работающем состоянии. Загвоздка в том, что сделать это надо с наименьшими рисками и потерями — чтобы в процессе это можно было мониторить, оповещать кого следует и в случае чего откатиться к последней работающей версии. Процесс, который придумали для решения этой задачи сводится по сути к двум аббревиатурам: CI и CD.

**CI (Continuous Integration)** — непрерывная интеграция — автоматическая проверка кода при внесении постоянных небольших изменений;

**CD (Continuous Delivery)** — непрерывная доставка — автоматическое развертывание этого кода.

**GitLab CI**

**GitLab** — это, как ты уже можешь знать, популярный веб-сервис для совместной разработки и поддержки программного обеспечения.

**GitLab CI** — инструмент, встроенный в GitLab для автоматизации рутинных задач, возникающих в процессе разработки программного обеспечения. Конфигурация для GitLab CI описывается в отдельном файле .gitlab-ci.yml. **GitLab-runner** — это программа — агент, которая выполняет инструкции, указанные в файлах .gitlab-ci.yml. Проще говоря — агент, который выполняет инструкции твоего пайплайна. Более подробную информацию, необходимую для установки и конфигурации раннеров ты можешь найти [здесь](https://docs.gitlab.com/runner/).😉

Познакомиться с возможностями GitLab CI и использовать его для автоматизации сборки и тестирования проекта ты можешь самостоятельно, так как эта тема выходит за рамки ручного тестирования и относится уже к его автоматизации.

**Задание №7. Docker и Kubernetes**

Создайте файл exercise7.md, в который добавьте ответы на следующие вопросы:

* Что такое Docker? Для чего он используется?
* Чем Docker отличается от виртуальной машины?
* Что такое Docker Compose?
* Что такое Kubernetes? Для чего он используется?

**Double-check**

Перед загрузкой выполненного проекта в репозиторий перепроверьте наличие всех необходимых файлов, которые требовалось создать во время его выполнения:

exercise1.md

exercise2.png

exercise2.reports.md

exercise2.requests.md

exercise3.md

exercise4.1.png

exercise4.2.png

exercise4.md

exercise5.1.png

exercise5.2.png

exercise5.3.png

exercise5.md

exercise6.1.png

exercise6.2.png

exercise6.3.png

exercise6.4.png

exercise6.5.png

exercise6.md

exercise7.md

💡 [Нажми здесь](https://forms.gle/TUkAA914bJ9TQZHH7) **чтобы отправить обратную связь по проекту**.