Narzędzia w testowaniu

Radosław
Kowal
02.08.2020

Agenda

1) Wprowadzenie do narzędzi
2) Zarządzanie testami i incydentami
3) Narzędzia pomocnicze (robienie screenshotów, generatory)
4) Automatyzacja
5) Systemy kontroli wersji
6) Continuous Integration
7) Testy wydajnościowe
8) Testy webservice'ów
9) Testy bezpieczeństwa

Czym są narzędzia testowe i po co nam one?

Są one wykorzystywane do czynności testowych przez zautomatyzowanie powtarzających się zadań lub wsparcie dla czynności testowych wykonywanych ręcznie takich jak: planowanie testów, projektowanie testów, raportowanie i monitorowanie testów
Automatyzacja czynności które zajmuje dużo czasu ręcznie (analiza statyczna)
Automatyzować czynności, które nie mogą być wykonane ręcznie (np. testy aplikacji klient-serwer na wielką skalę)
Poprawić „niezawodność testów” (np. przez automatyzację porównywanie dużej ilości danych lub symulacje)

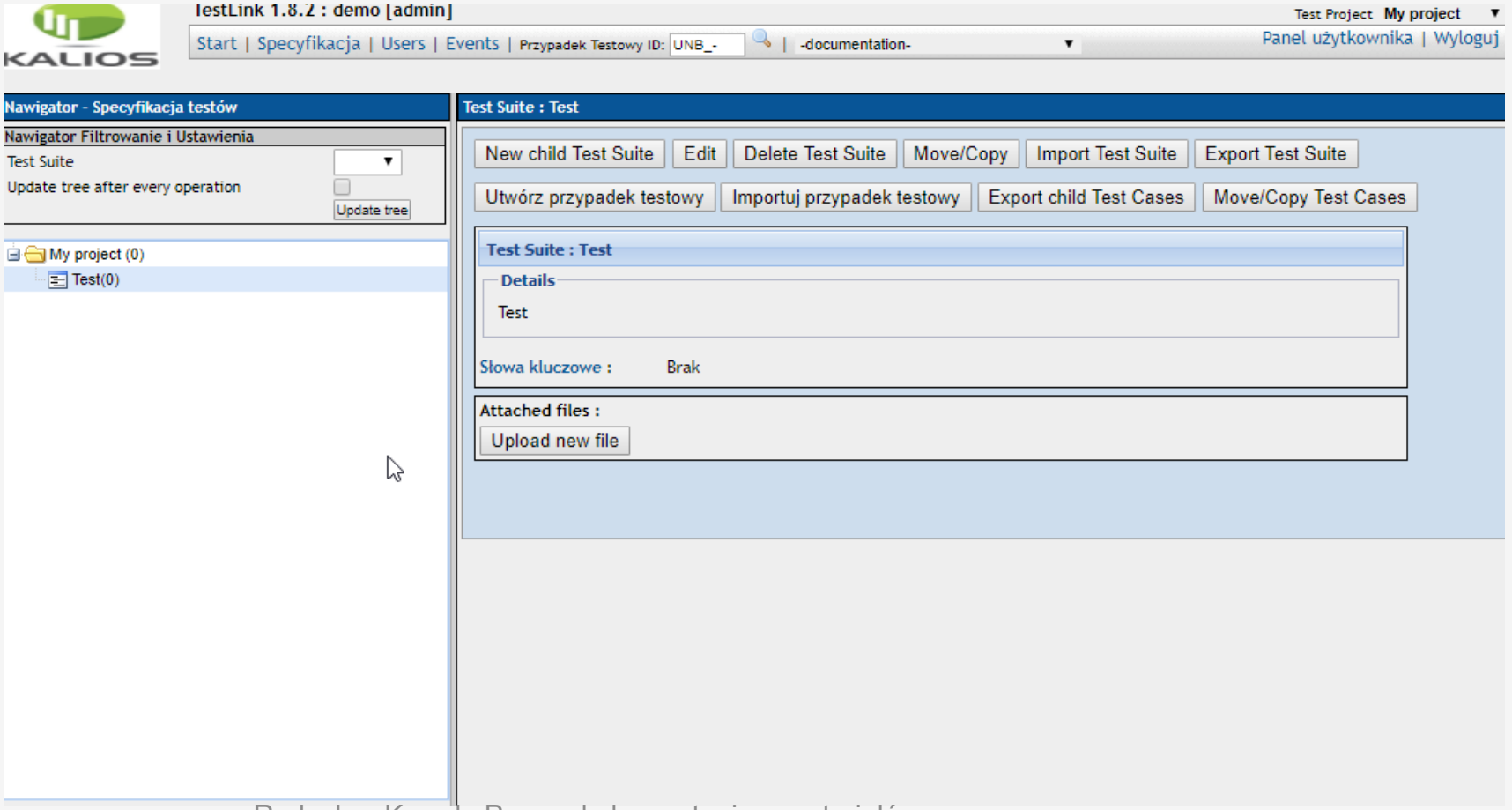
Przykładowe narzędzia

Zarządzanie błędami i testami
Tworzenie screenshotów i nagrywanie ekranu
Generatory
Wtyczki i konsole przeglądarkowe
Narzędzia do automatyzacji
Systemy kontroli wersji
Narzędzia do Continuous Integration

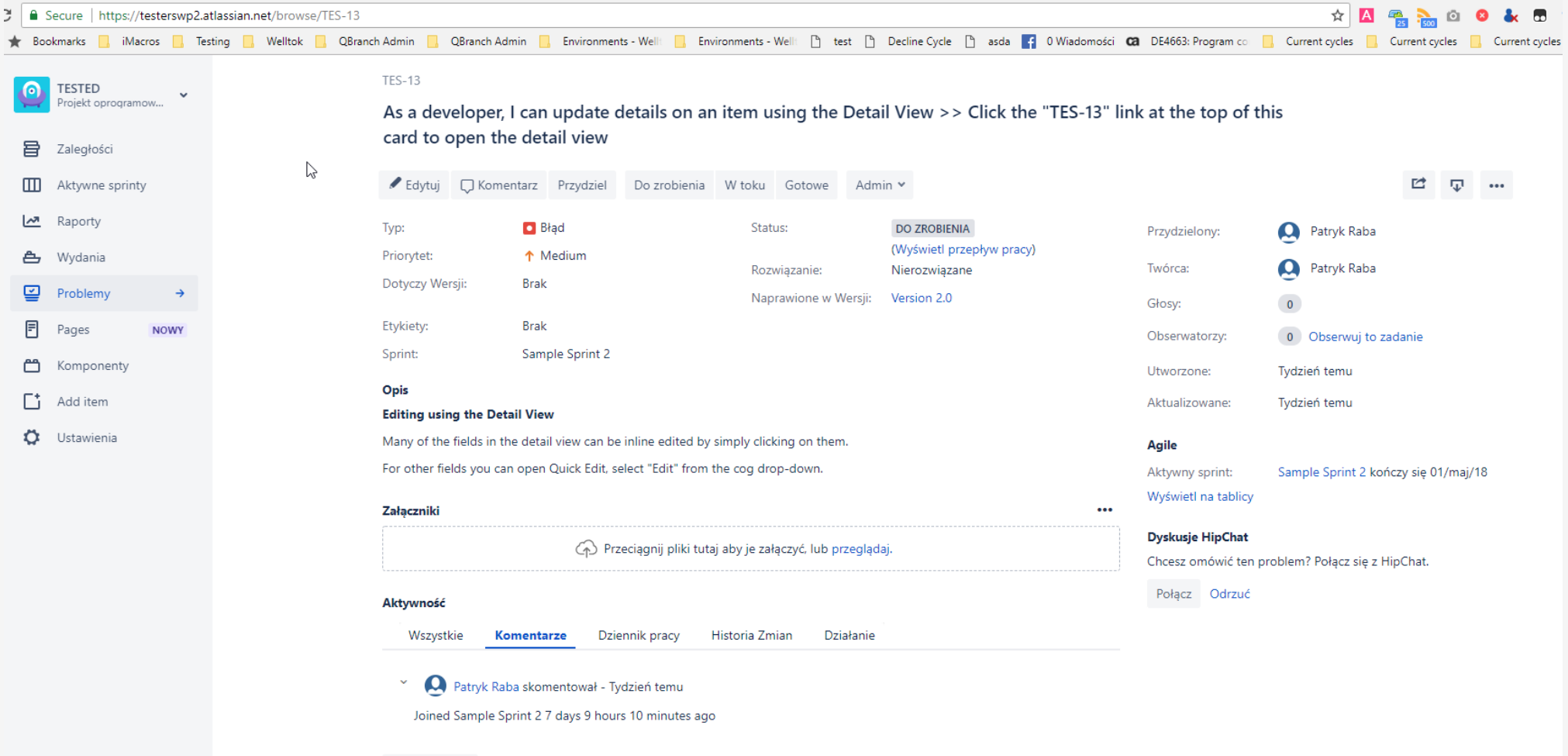
Zarządzanie incydentami/testami

TestLink

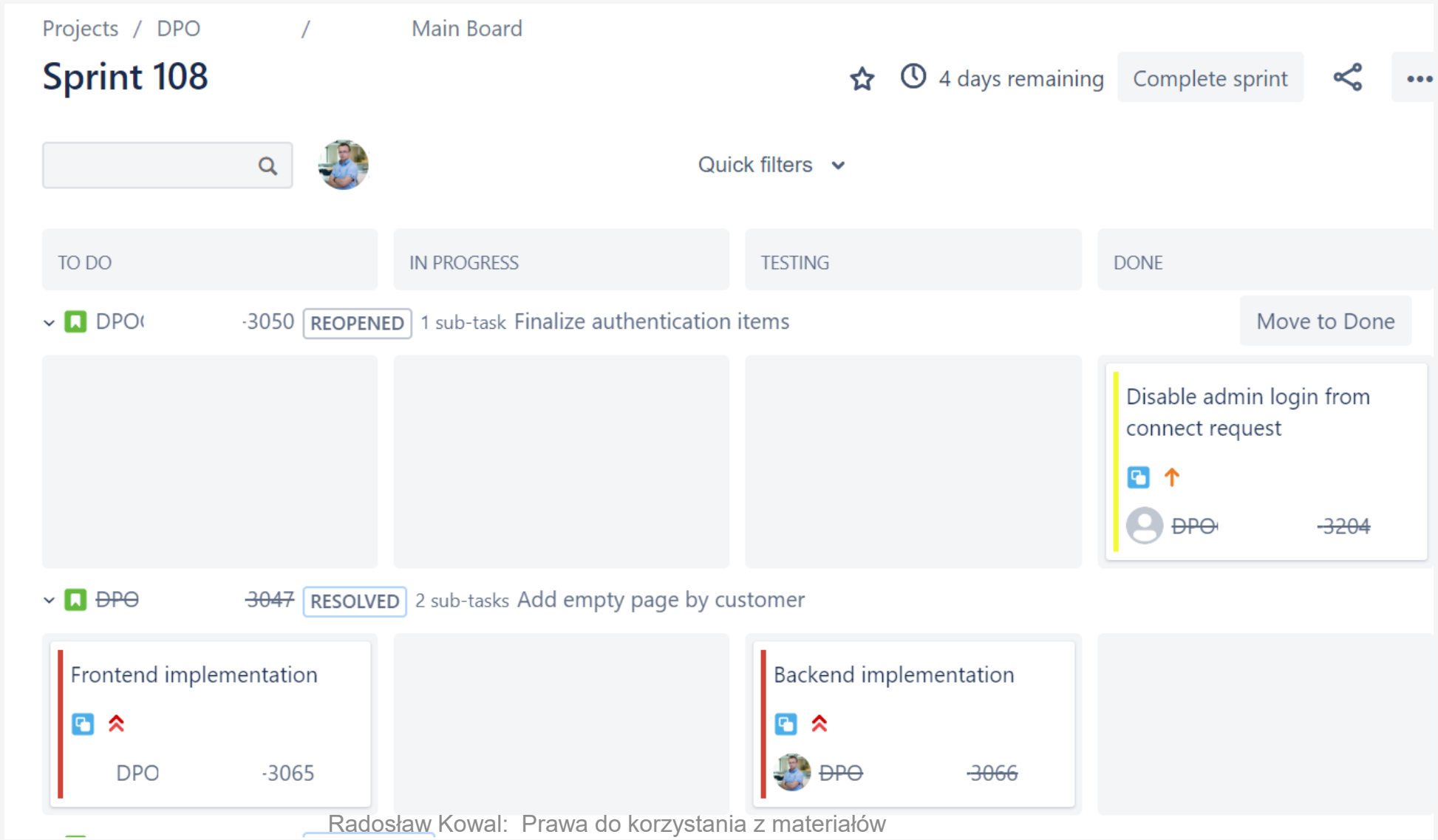
Bitnami Testlink (instalacja na własnym komputerze): http://127.0.0.1/testlink



JIRA



JIRA



BugZilla

Before reporting a bug, please read the bug writing guidelines, please look at the list of most frequently reported bugs, and please search for the bug.

Show Advanced Fields (* = Required Field)

* Product: OpenDemo.ORG Reporter: odoun54568

* Component: bugzilla-4.2.1

Component Description
bugzilla-4.2.1

* Version: unspecified

Severity: enhancement

Hardware: PC

OS: Windows

We've made a guess at your operating system and platform. Please check them and make any corrections if necessary.

* Summary:

Description:

Attachment: Add an attachment

Submit Bug

Trac

Create New Ticket

Properties

Summary:

Description:

You may use WikiFormatting here.

Type: defect

Milestone:

Version:

Cc:

Priority: major

Component: component1

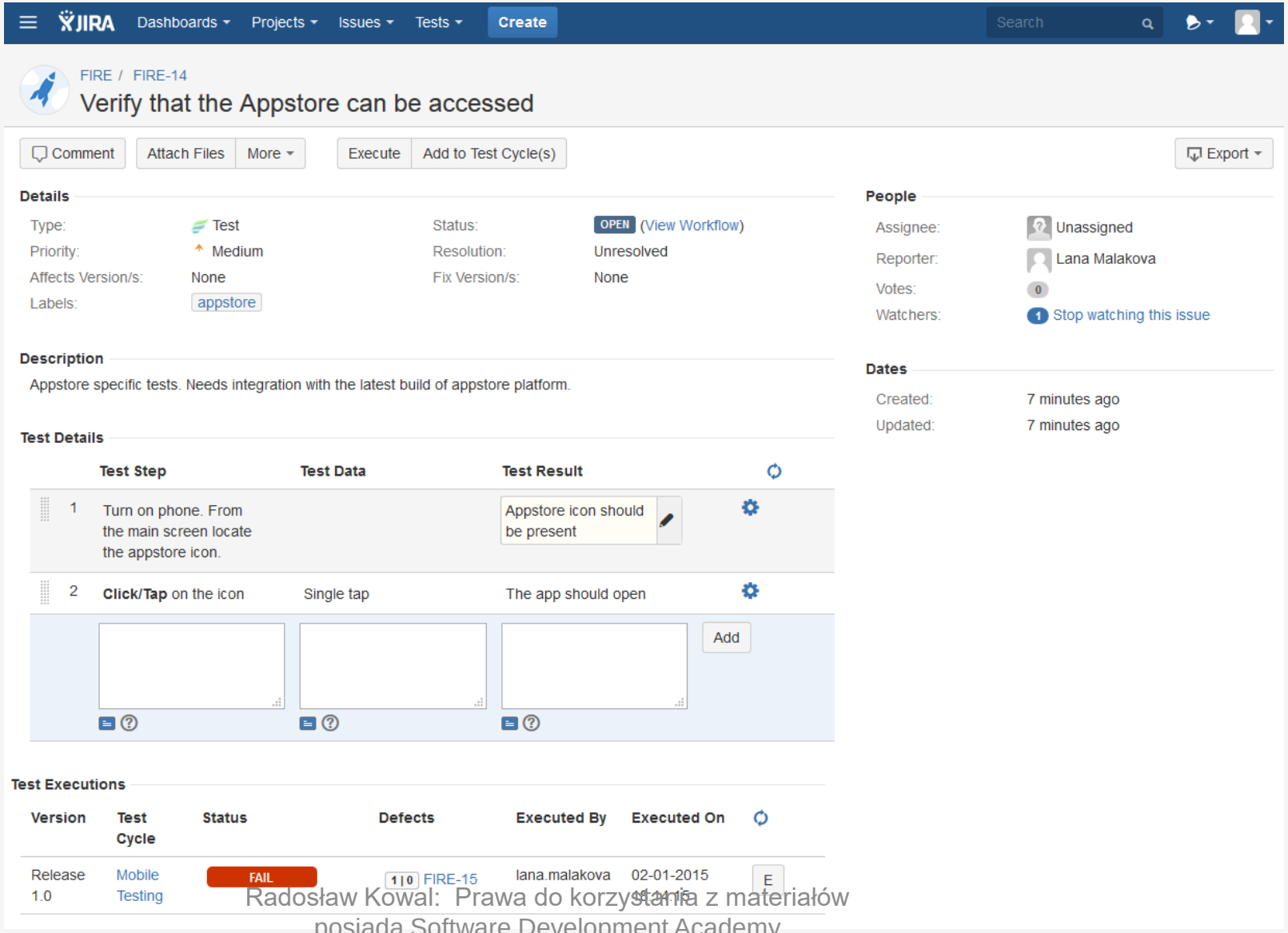
Keywords:

Owner: < default >

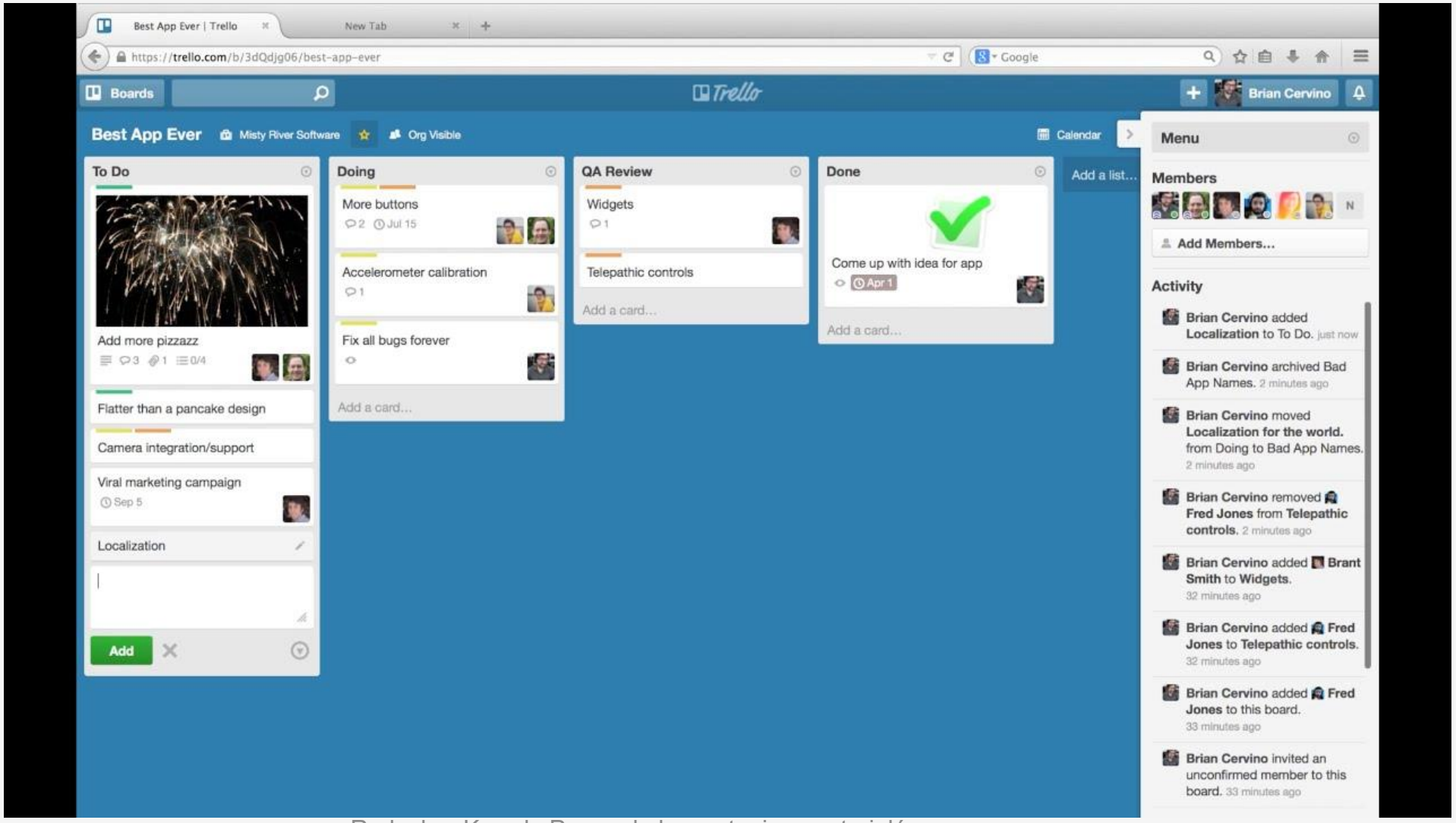
I have files to attach to this ticket

Preview Create ticket

Zephyr



Trello



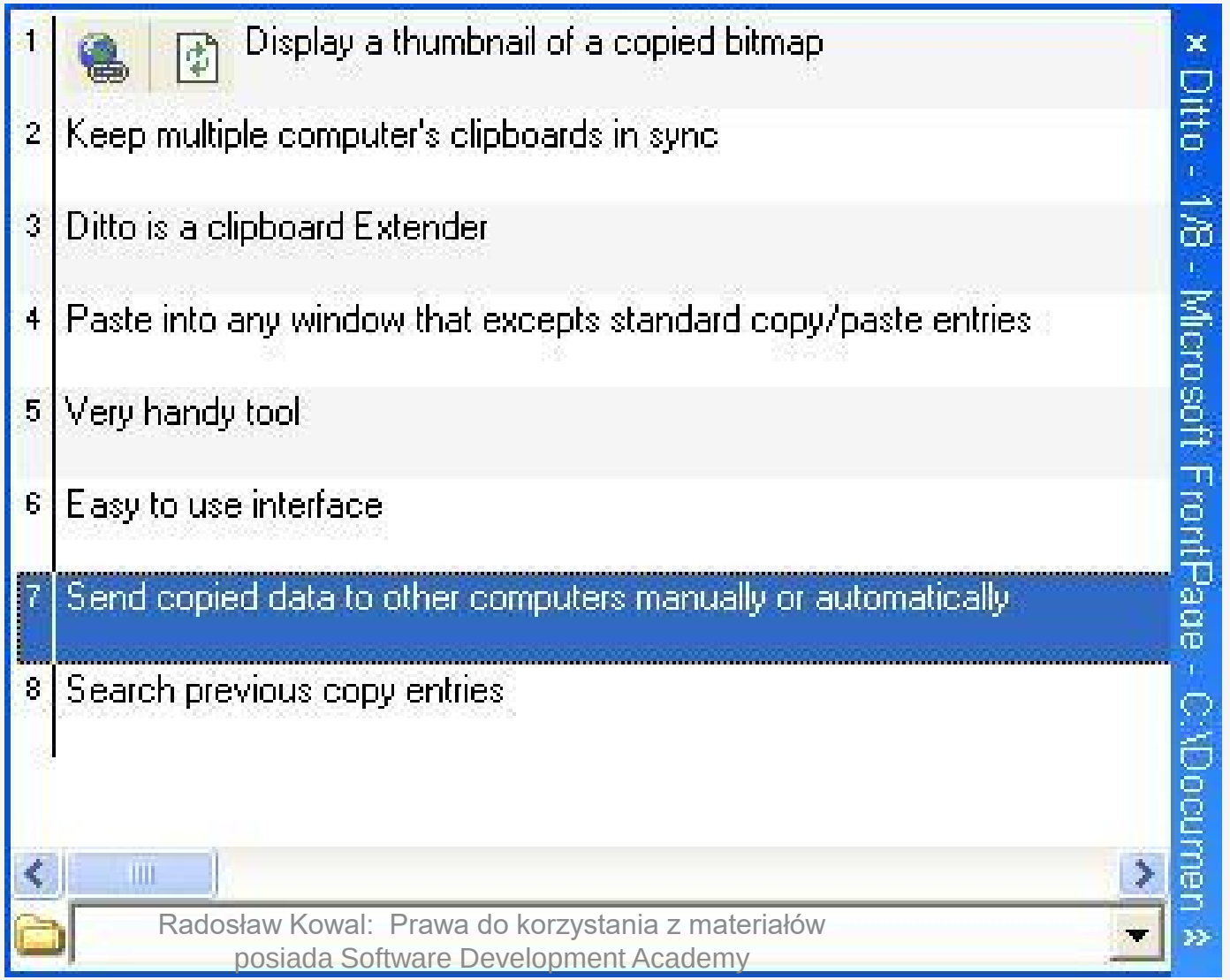
Praca domowa

Na stronie http://www.opendemo.org/open-source-demos podaj swój adres mailowy w sekcji Issue Tracking, dostaniesz na niego link generujący Bugzillę, Mantisa i Traca.
Pobierz aplikację Mr Buggy ze strony http://mrbuggy.pl/mrbuggy1/data/MrBuggy.exe
W aplikacjach z punktu 1 zgłoś kilka błędów. Dla ułatwienia (w końcu uczymy się obsługi samych narzędzi) znajdziesz je pod tym linkiem:
https://docs.google.com/spreadsheets/d/1dt2_xVu8AXMIGKdRxuThljb0Ldx2RJQ4Q4a9rpanOBo/pub?output=html

Powodzenia :)

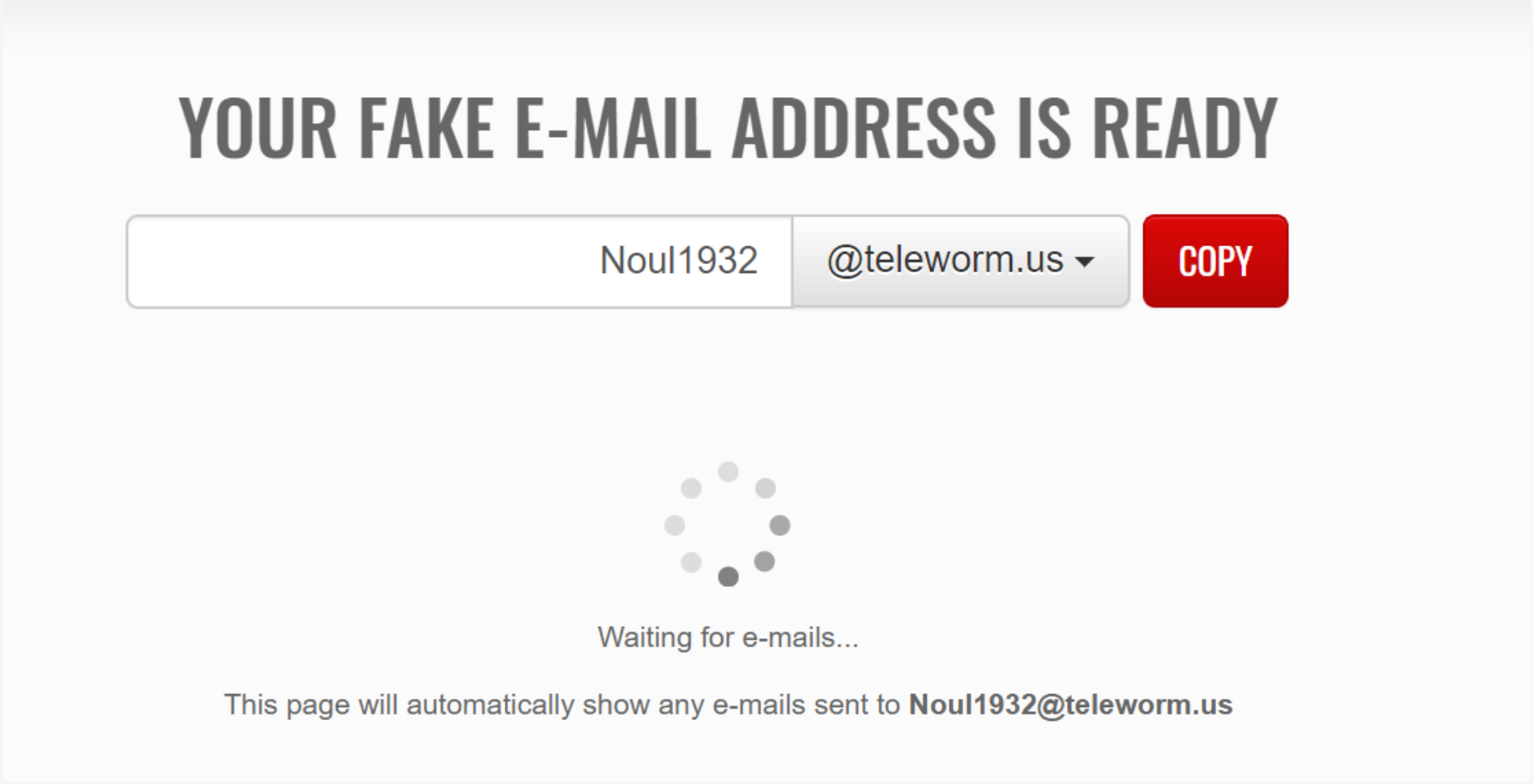
Narzędzia do robienia screenshotów i nagrywania ekranu (Screenpresso)

Multischowek (Ditto)



Tymczasowe skrzynki pocztowe

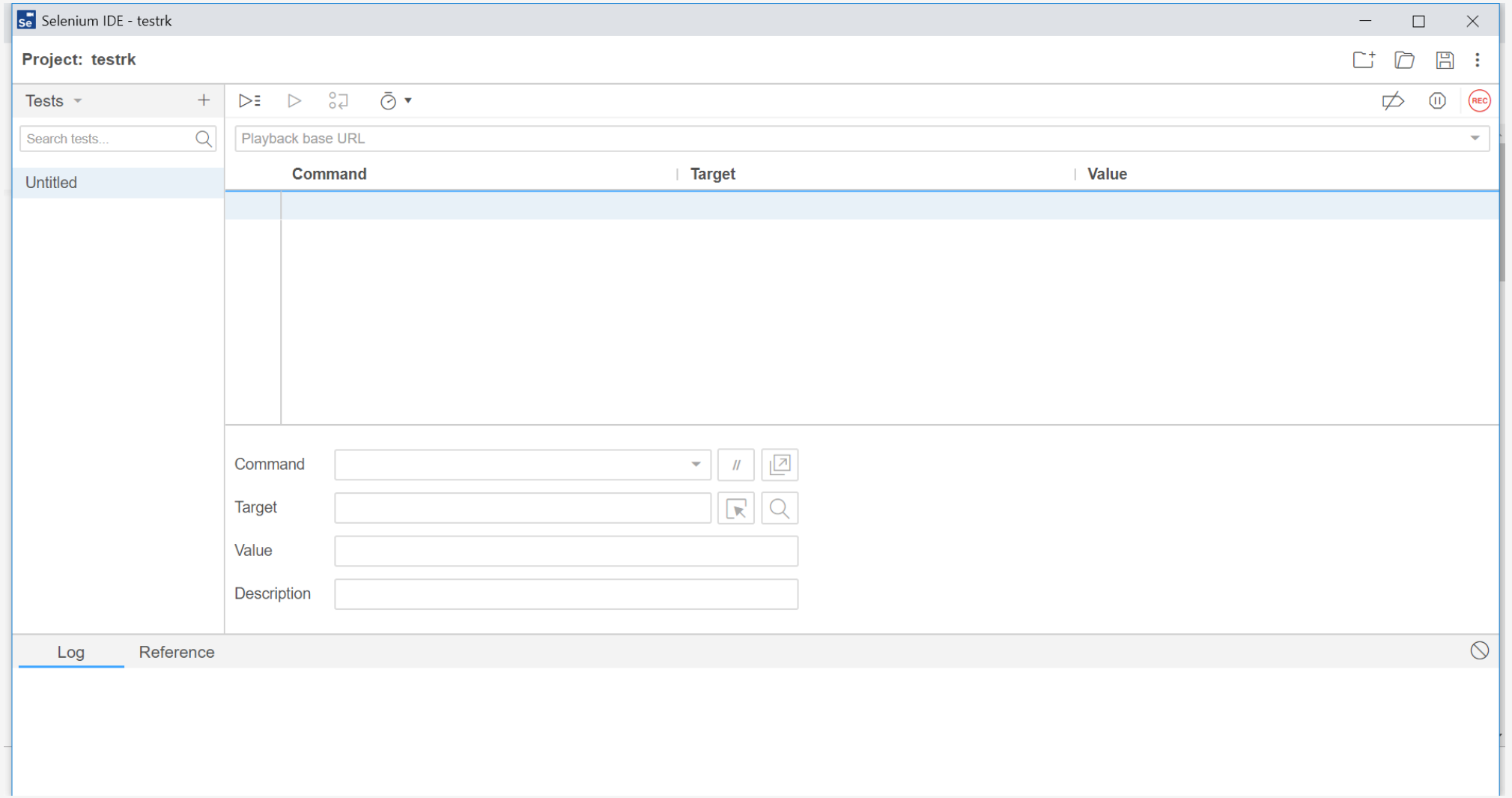
http://www.fakemailgenerator.com



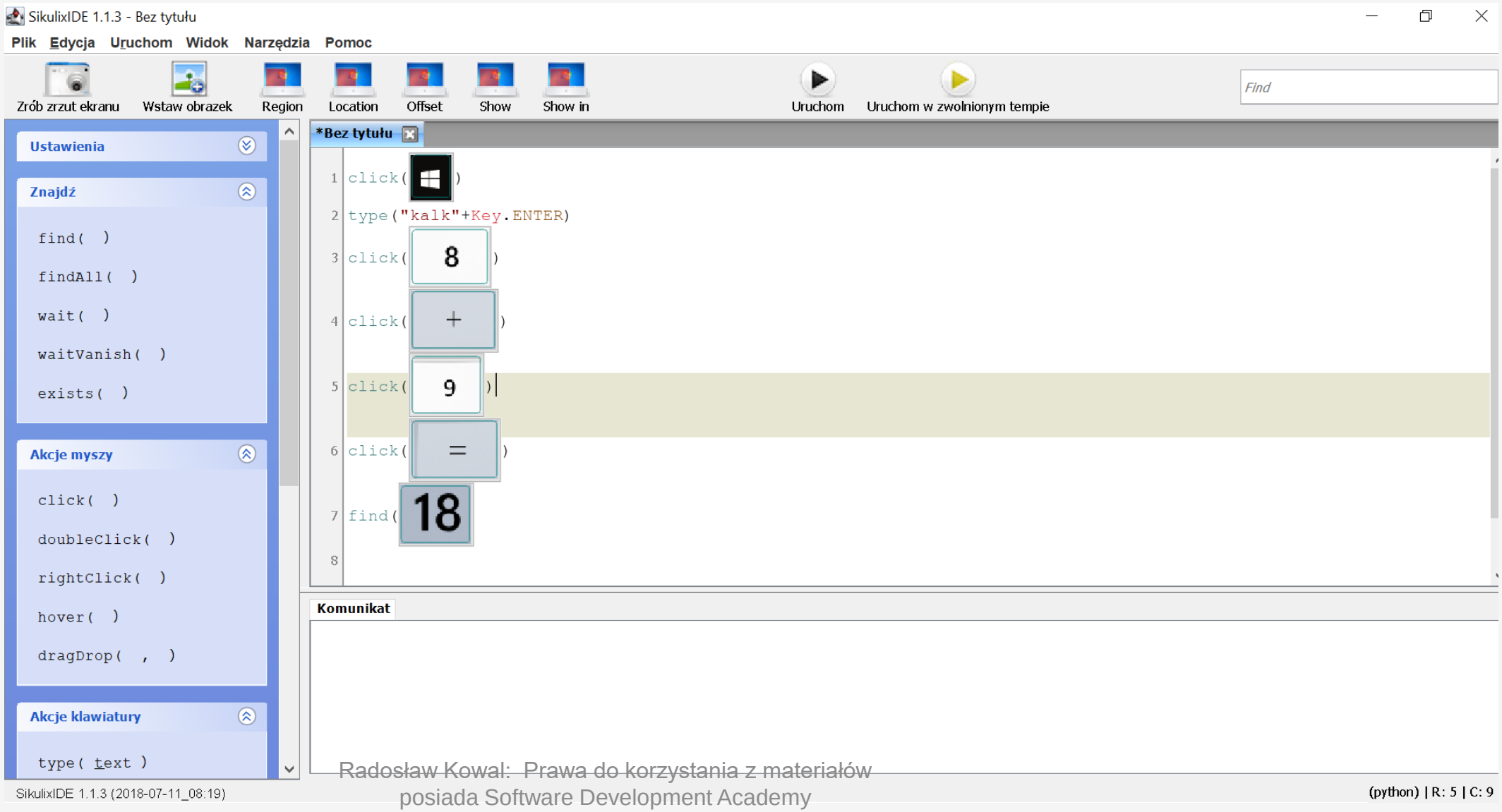
Generatory haseł

https://generator.blulink.pl/

Automatyzacja (Selenium IDE)



Automatyzacja (SikuliX)



Praca domowa

1. Przy użyciu narzędzia Selenium IDE przeprowadź test logowania do swojej skrzynki pocztowej.
2. Przy życiu narzędzia SikuliX sprawdź, czy 2+9-5*8=48, a następnie czy 2+9-5*8=47

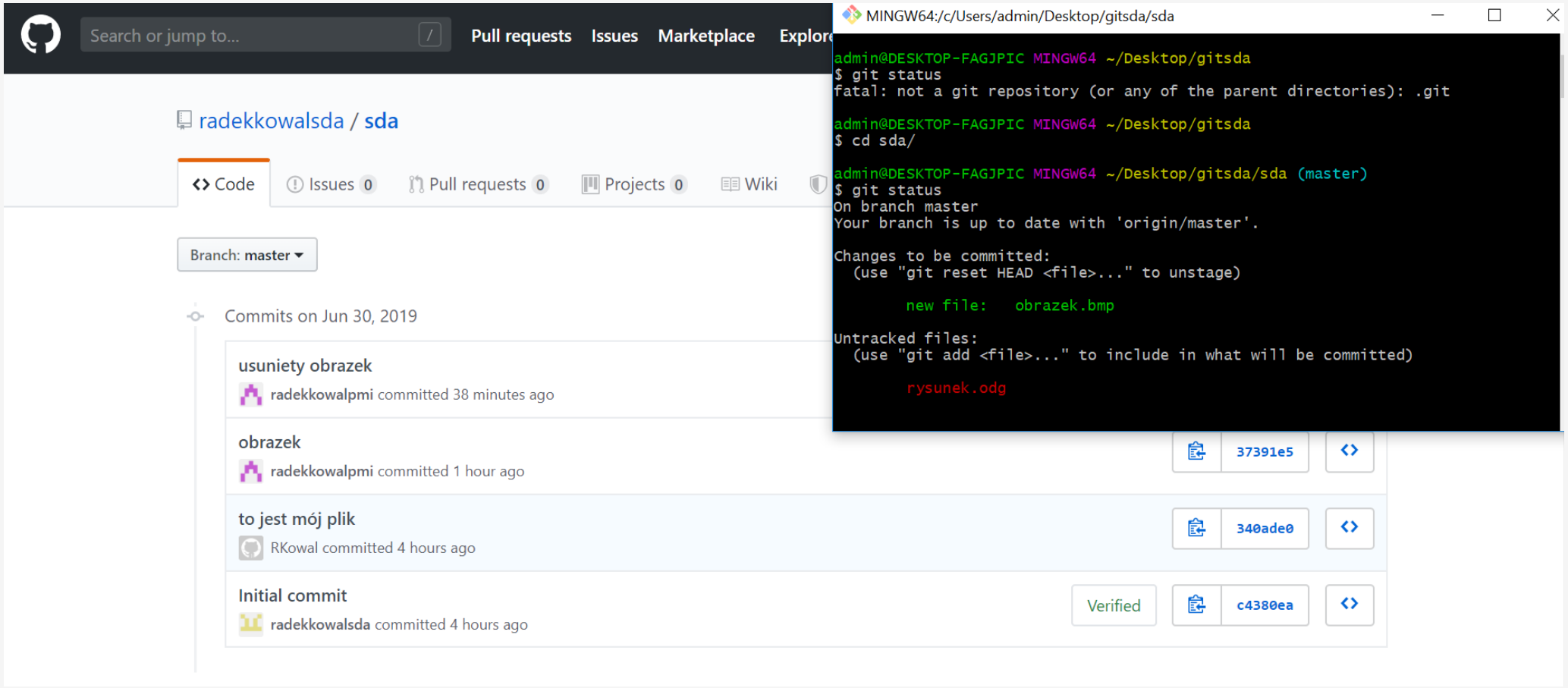
Systemy kontroli wersji

Jest to program zapisujący zmiany zachodzące w plikach (wersje), dzięki czemu możemy przejrzeć ich historię i w razie potrzeby – przywrócić. Wszystkie te informacje są zapisywane w tzw. repozytorium projektu.

Systemy kontroli wersji umożliwiają:
Przegląd historii zmian wraz z informacją kto i kiedy je wprowadził
Przywrócenie Dowolnej wersji pliku lub nawet całego projektu
Pracę zespołową, poprzez wykorzystywanie zdalnych repozytoriów (w serwisach takich jak GitHub, BitBucket lub GitLab)

Git

Najpopularniejszy system kontroli wersji



Git – przydatne komendy

git clone *link_do_repozytorium* – pobiera repozytorium ze zdalnego serwera (wykonujemy tylko na samym początku)
git pull – pobiera zmiany (aktualizuje repozytorium na naszym dysku)
git commit -m 'nazwa wprowadzonych zmian' - zatwierdza dokonane przez nas zmiany
git push – aktualizuje wszystkie nasze zmiany (commit'y) na zdalnym repozytorium
git checkout *nazwa_brancha* – zmiana gałęzi repozytorium
git config --global user.name "imię" – ustawia nazwę użytkownika
git config --global user.email "email" – ustawia email

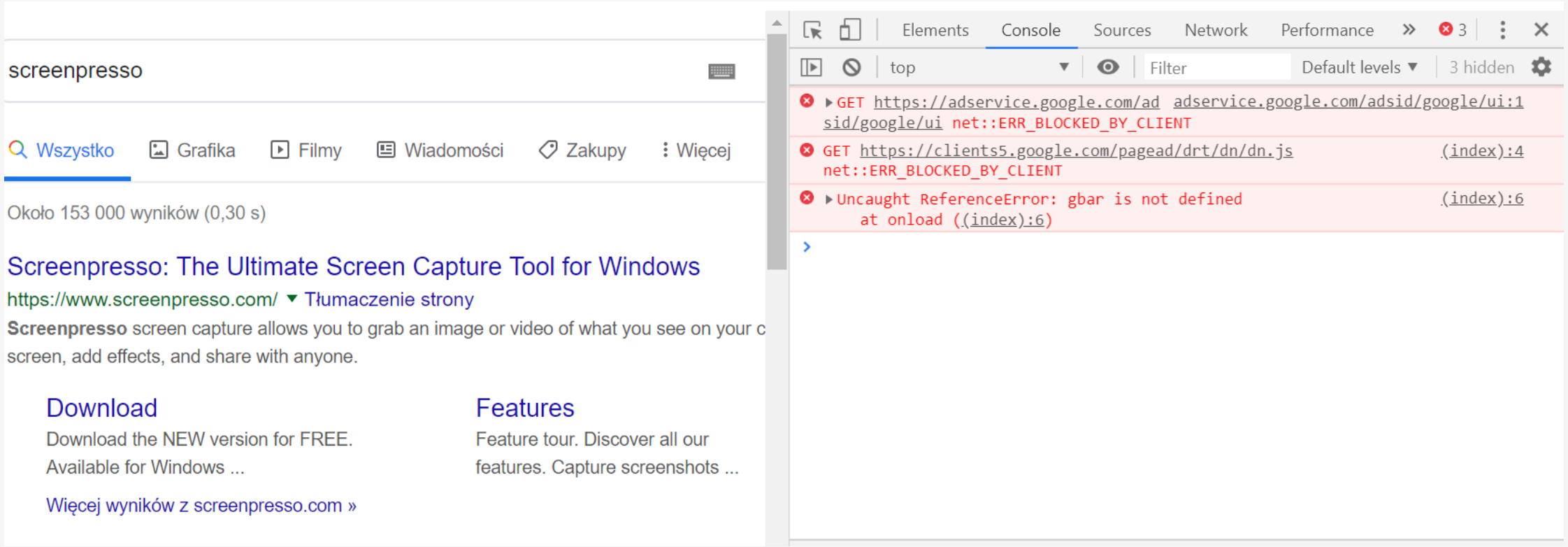
SVN

Repozytorium SVN służy do kontroli wersji plików niebinarnych (czyli np. pliki tekstowe, html, php, bash). Użytkownicy przechowują w nim różne wersje plików, np. skryptów PHP. Możliwe jest również wysłanie innych plików do repozytorium. Należy jednak pamiętać, że SVN służy do kontroli wersji głównie plików tekstowych i wysyłanie innych plików mija się z celem.

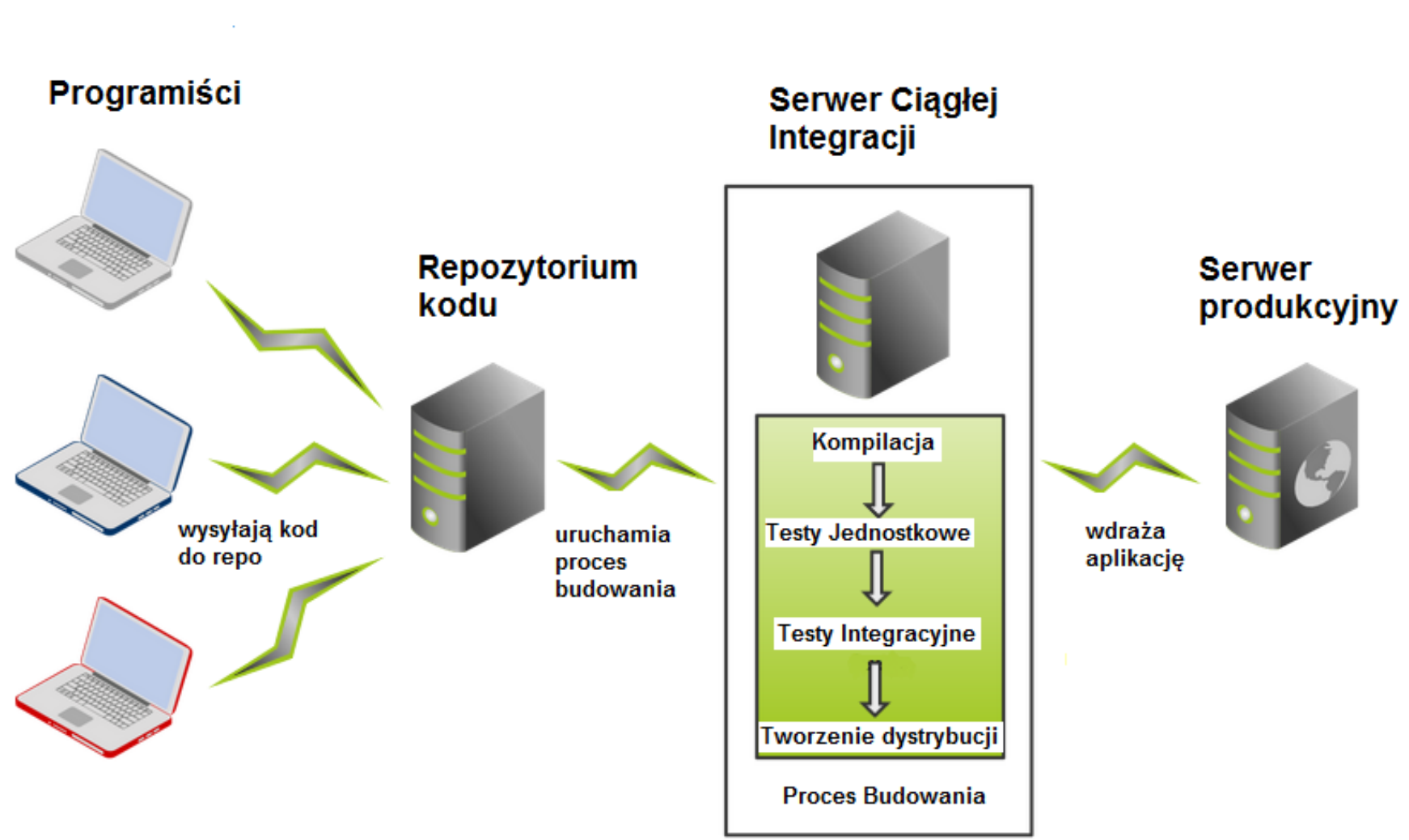
Git vs. SVN

- w SVN – jeden etap przenoszenia zmian na serwer, w Git są to dwa etapy – zapis do lokalnego repozytorium, a potem na serwer;
- pozwala to na pracę offline, a ponadto możliwe jest wysłanie nie wszystkich zmian, które dokonaliśmy;
- Git jest dużo szybszy :)

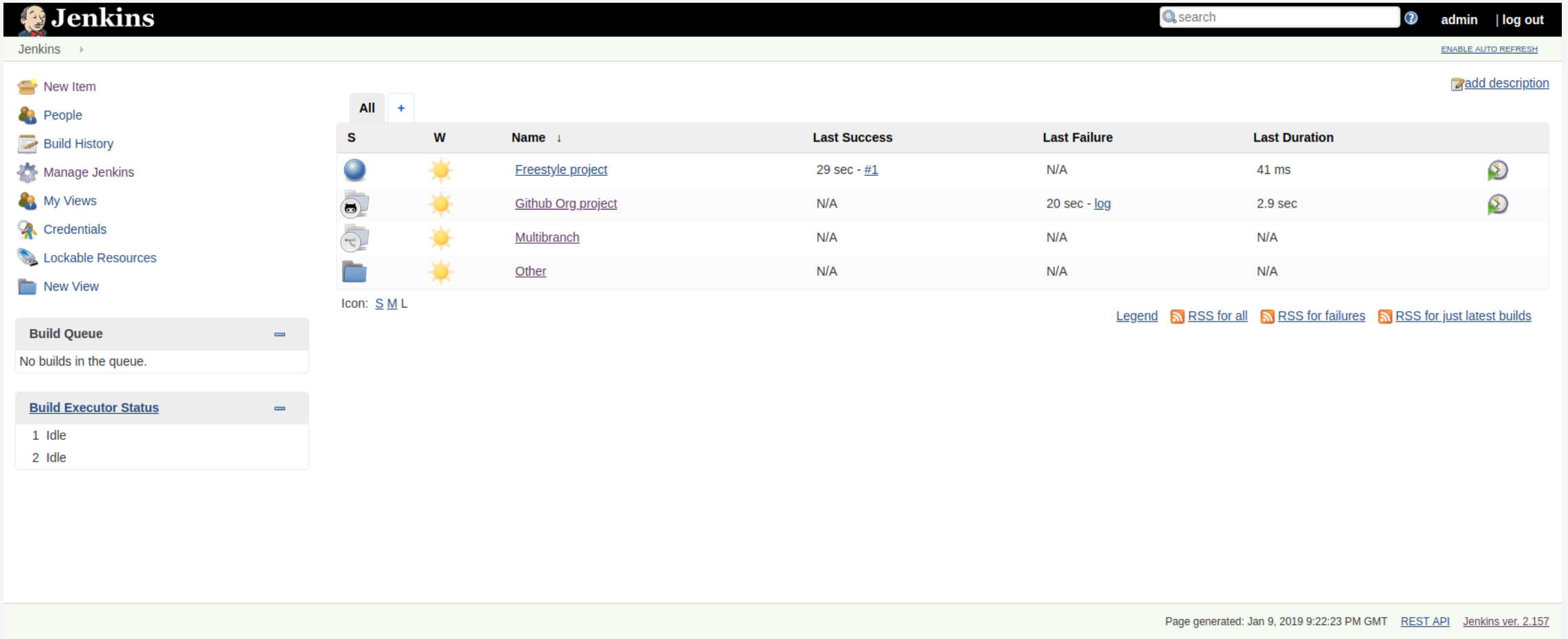
Narzędzia developerskie przeglądarki



CI w procesie tworzenia oprogramowania



Jenkins



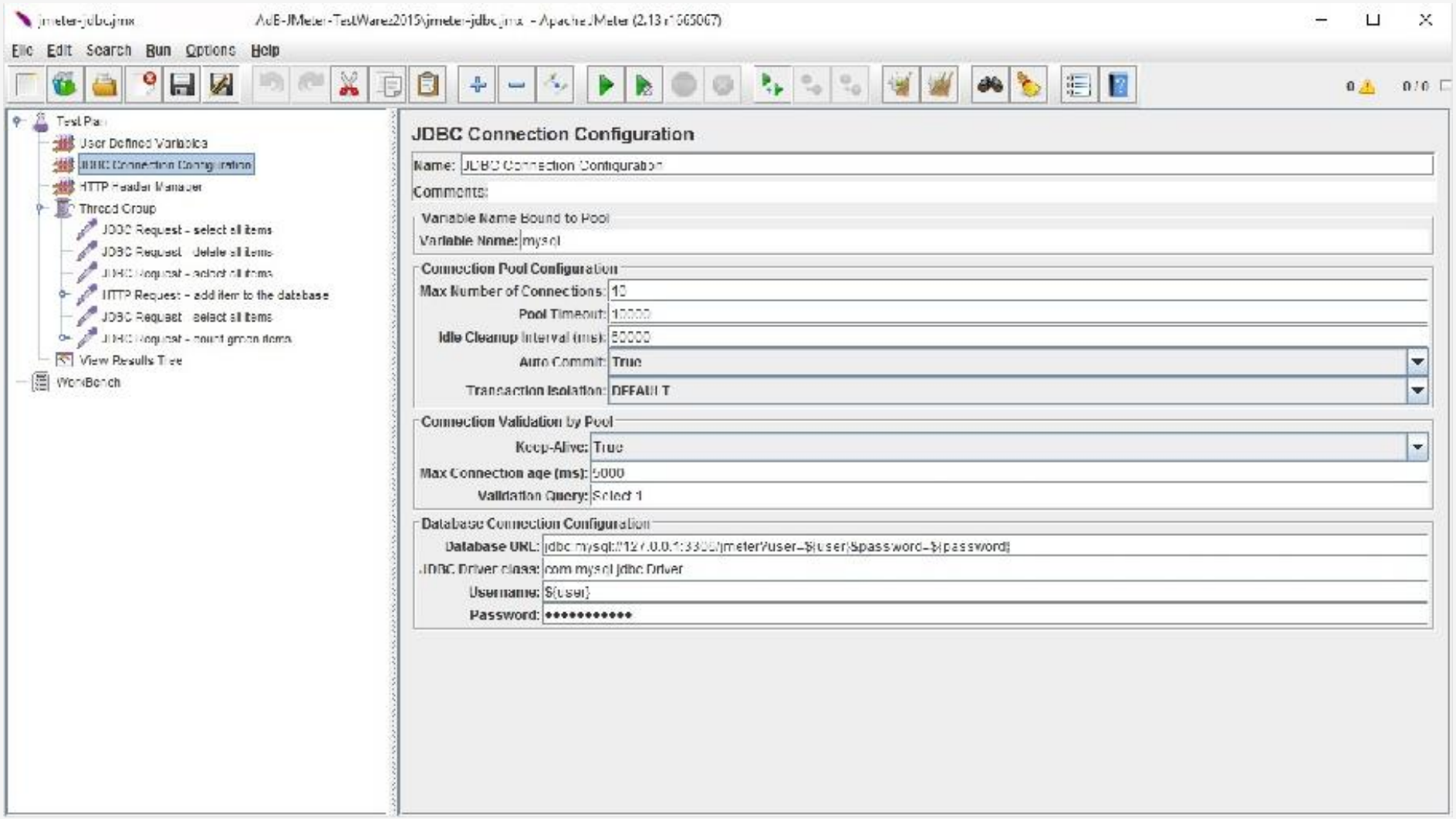
Po co CI to?

•Ciągła Integracja zmniejsza ryzyko związane z integracją na samym końcu projektu – błędy, niekompatybilność interfejsów, trudny do oszacowania czas na poskładanie całości.
•CI ułatwia naprawę błędów: ich szybkie wykrywanie sprawia, że łatwiej zlokalizować przyczynę – wiadomo, co było ostatnio modyfikowane i jaka wersja działała poprawnie.
•CI chroni przed niespodziankami wynikającymi z różnic pomiędzy środowiskiem deweloperskim a produkcyjnym (np. inne środowisko uruchomieniowe danego języka, niestandardowe biblioteki).
•CI umożliwia demonstrowanie aplikacji i konsultację z klientem w dowolnym momencie dzięki stałej dostępności ostatniej działającej wersji.
•CI ułatwia refaktoryzację (po każdej „kosmetycznej” zmianie możemy szybko sprawdzić, czy wszystko gra).
•Ciągła integracja zdejmuje z programistów obowiązek wykonywania wielu powtarzalnych, nierozwijających (a jednak trudnych!) czynności.

Testy wydajnościowe

•Testy wydajnościowe są przeprowadzane w celu oceny stopnia spełnienia wymagań wydajnościowych przez system lub moduł.
•Istnieje kilka rodzajów wymagań wydajnościowych:
•wymagania na szybkość przetwarzania,
•wymagania na równoległość przetwarzania,
•wymagania na wielkość obsługiwanych danych.
•Testy wydajnościowe przeprowadza się zwykle w dwóch sytuacjach: na granicy wymagania wydajnościowego oraz powyżej wymagania wydajnościowego. W tym drugim przypadku testy są nazywane przeciążeniowymi.

JMeter



Gatling

- jest darmowym narzędziem do wykonywania testów wydajnościowych,
- działa na systemach Windows, Linux oraz MacOS,
- został napisany głównie w języku Scala i jest oparty o AKKA i NETTY,
- dzięki zastosowanym rozwiązaniom posiada asynchroniczną architekturę, wprowadza model aktora, który jest zorientowany na wysyłanie wiadomości zamiast tworzenia dedykowanych wątków, pozwalając na generowanie większych obciążeń,
- skrypty testowe są pisane w Scali, przy czym wystarczy podstawowa znajomość tego języka, gdyż skrypty są tworzone z wykorzystaniem łatwego w użyciu DSL (Domain Specific Language), przez co tworzenie i późniejsze zrozumienie skryptów jest proste,

Testy webservice

Usługi sieciowe (ang. webservice) to mechanizmy, które pozwalają na komunikację klienta z serwerem. Klient, czyli użytkownik, chciałby wyświetlić swoje ulubione restauracje na mapie - aby to zrobić, musi wysłać do serwera odpowiednie żądanie (ang. request). Jeśli żądanie jest prawidłowe, użytkownik otrzyma odpowiedź (ang. response) w odpowiednim formacie. Błędnie sformułowane zapytanie lub brak zasobu na serwerze spowoduje odpowiednie akcje informujące o niepowodzeniu.

REST (ang. Representational State Transfer) opiera się na konkretnych adresach URL, które poniekąd działają jak identyfikatory. Na dany adres URL zostaje wysłane zapytanie na serwer, następnie serwer przetwarza nasze zapytanie i dostajemy odpowiedź. Wyróżnia się 7 podstawowych metod do komunikacji.

Testy webservice (Postman)

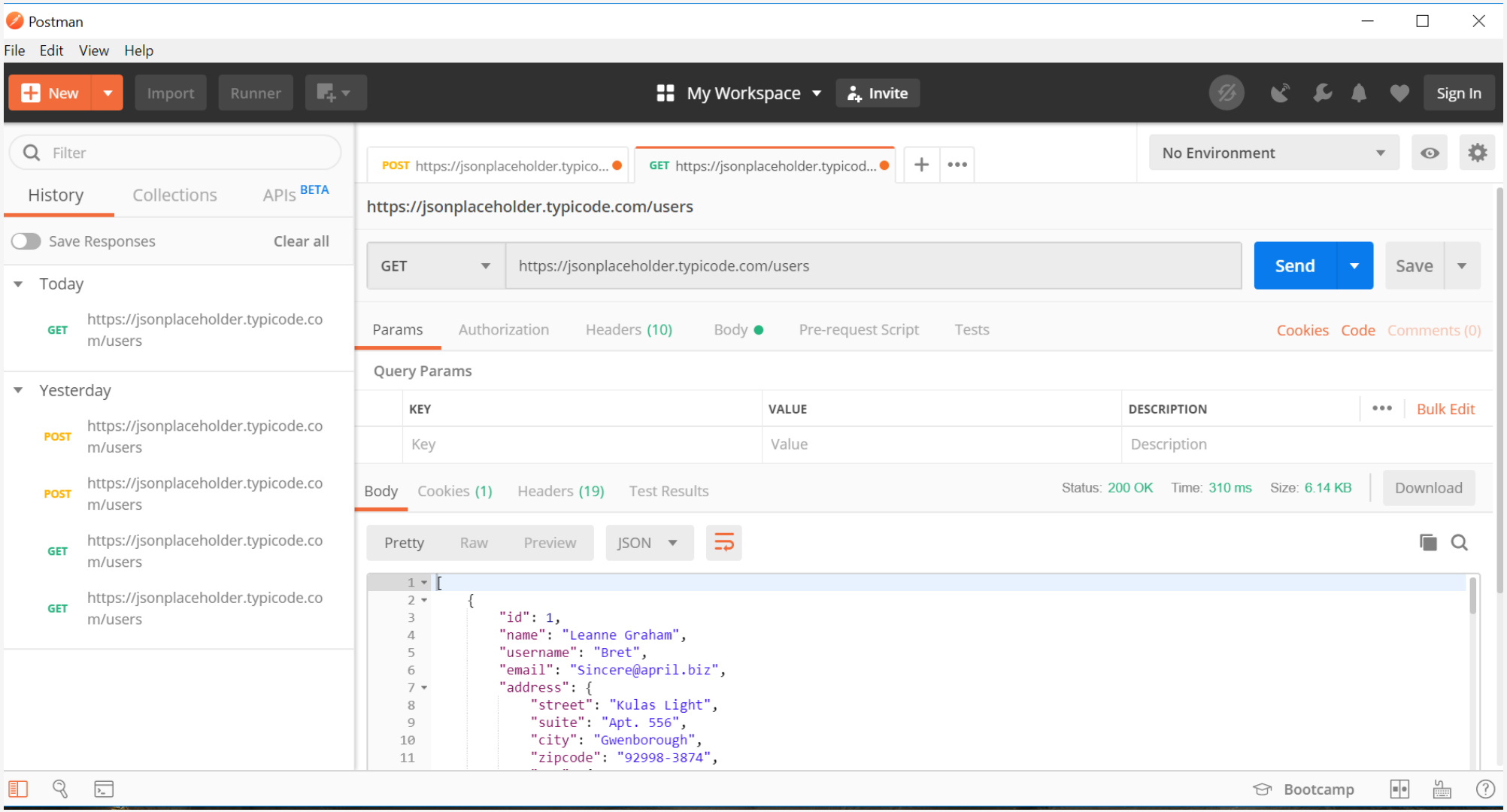
Postman służy głównie do wysyłania żądań różnego typu:

- GET
- POST
- PUT
- DELETE

Są to cztery najbardziej popularne żądania wysyłane przy pomocy REST API.
Po każdym z wysłanych żądań, wysyłany jest kod odpowiedzi w postaci trzycyfrowego numeru. Pierwsza liczba kodu odpowiedzi definiuje rodzaj komunikatu. Są to odpowiednio:
Kody informacyjne - rozpoczynające się od 1, np. 101, 111

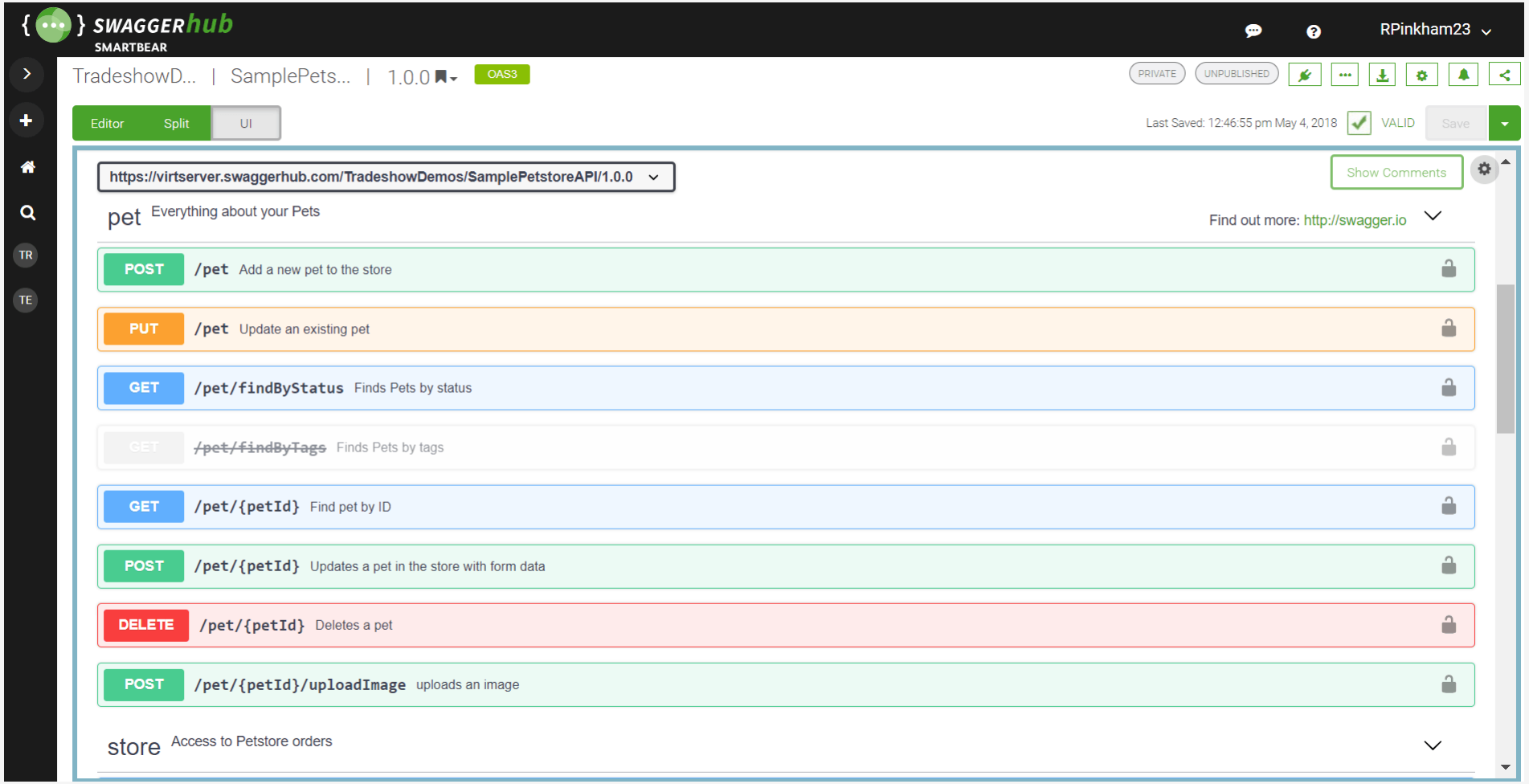
Kody powodzenia - rozpoczynające się od 2, np. 200, 201
Kody przekierowania - rozpoczynające się od 3, np. 301, 306
Kody błędu aplikacji klienta - rozpoczynające się od 4, np. 404
Kody błędu serwera HTTP - rozpoczynające się od 5, np. 500, 501

Testy webservice (Postman)



Testy webservice (Swagger UI)

https://petstore.swagger.io/?_ga=2.166840766.1007213295.1590004620-467914185.1590004620#/



Testy bezpieczeństwa

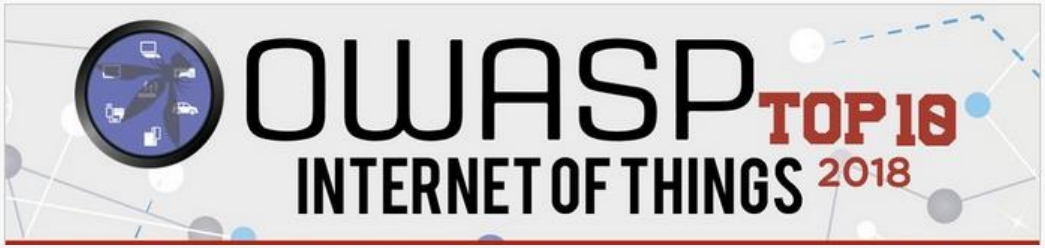
Testy bezpieczeństwa są kolejnym typem testów niefunkcjonalnych. Testy bezpieczeństwa są kluczowe dla niektórych typów aplikacji — szczególnie dla aplikacji przechowujących dane poufne. Testy bezpieczeństwa mogą być wymuszone poprzez umowę czy uwarunkowania prawne. Obecnie widać coraz większy nacisk na testy bezpieczeństwa, osoba zajmująca się testami bezpieczeństwa nazywana jest Pentesterem lub Testerem bezpieczeństwa. Zazwyczaj zajmuje się tylko tym jednym typem testów. Podstawowe testy bezpieczeństwa można wykonać ręcznie, ale bardziej zaawansowane jak np. analiza pakietów sieciowych, jest możliwa tylko i wyłącznie przy pomocy narzędzi.

Testy bezpieczeństwa (OWASP)

OWASP - globalna, profesjonalna fundacja, działająca charytatywnie (non-profit), otwarta dla każdego, kto interesuje się zabezpieczeniami w oprogramowaniu

OWASP Top10

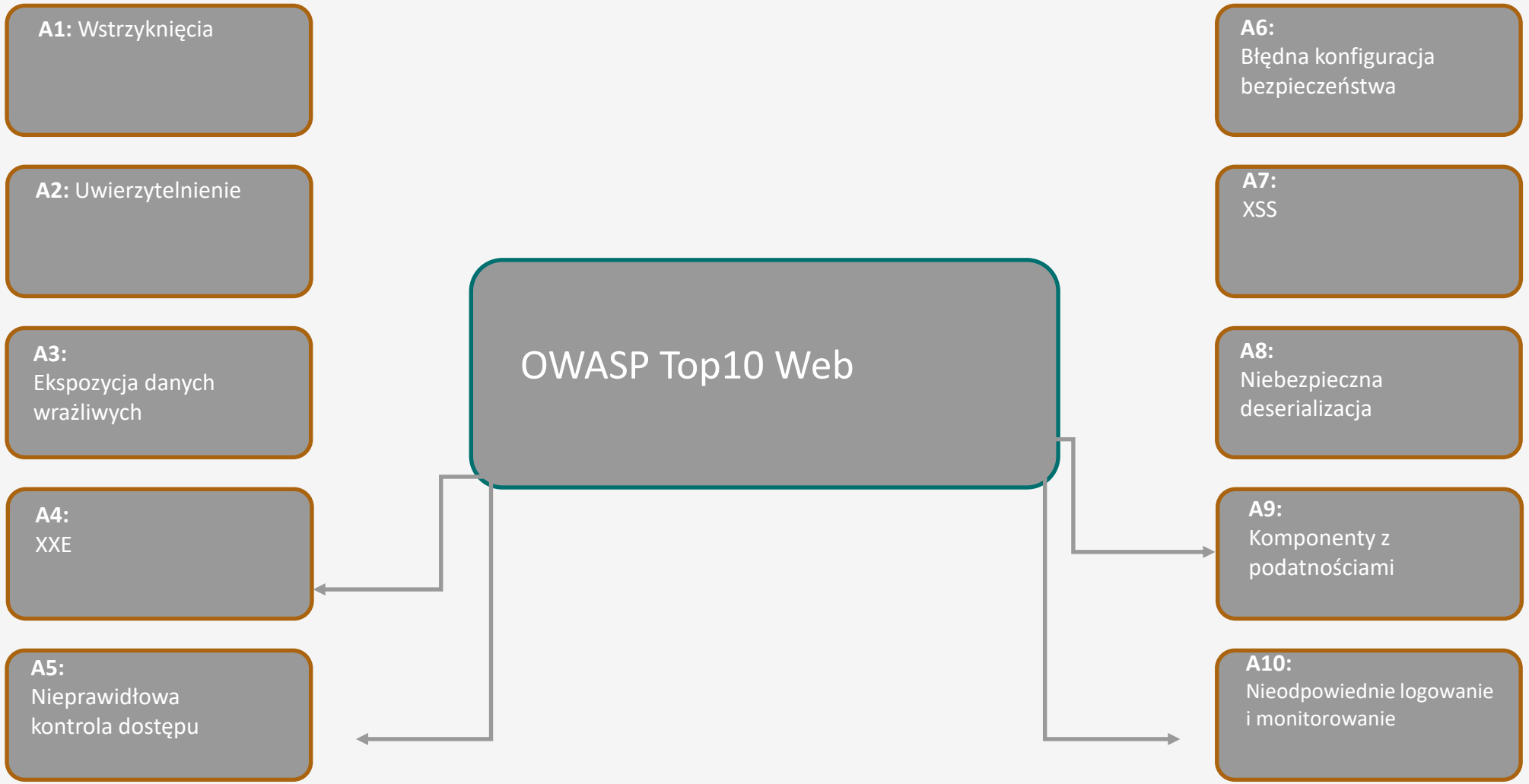
OWASP Top10 - web



OWASP Top10 - IoT

OWASP Top10 - mobile

OWASP Top10 Web



OWASP Top10 Web

Wstrzyknięcia - jeżeli w aplikacji nie zostanie zastosowane odpowiednie filtrowanie danych, to atakujący będzie mógł przesłać exploit w formie tekstowej wykorzystujący odpowiednią składnię docelowego interpretera. Wartości zostaną potraktowane jak polecenia, co może skutkować nieautoryzowanym dostępem do poufnych informacji, a nawet przejęciem kontroli nad atakowanym systemem.
Wadliwa obsługa uwierzytelniania i sesji - poprawne wdrożenie funkcji związanych z uwierzytelnianiem i obsługą sesji w aplikacji nie jest łatwe. Atakujący może wykorzystać nie tylko odkryte przez siebie usterki techniczne w implementacji lub konfiguracji oprogramowania, ale też błędy projektowe i organizacyjne. Konsekwencje ataku bywają zwykle poważne i obejmują uzyskanie nieautoryzowanego dostępu do sesji, przejęcie haseł lub tokenów, wykonanie poleceń na prawach zalogowanego użytkownika itp.

OWASP Top10 Web

Cross-Site Scripting (XSS) – skrypty międzyserwisowe - Luki XSS, w przeciwieństwie do wspomnianych wyżej wstrzyknięć, nie mają wpływu na logikę aplikacji po stronie serwera, pozwalają za to atakującemu na wykonywanie złośliwych skryptów w przeglądarce ofiary. Dzieje się tak, gdy aplikacja pobiera niezaufane dane i wysyła je do przeglądarki bez wcześniejszej walidacji. Skutkiem wykorzystania błędów tego typu może być np. przechwycenie sesji zalogowanego użytkownika, dynamiczna podmiana zawartości strony, jak również hostowanie złośliwego oprogramowania z wykorzystaniem zaatakowanej aplikacji.

Insecure Direct Object References – W aplikacjach, w których występują różne poziomy uprawnień, zdarzają się problemy wynikające z możliwości bezpośredniego dostępu do różnych obiektów w systemie (takich jak pliki, katalogi czy klucze bazy danych). Brak zdefiniowanych reguł dostępności sprawia, że atakujący może odpowiednio manipulować odwołaniami w celu dostania się do poufnych danych. Przykładowo, jeśli aplikacja nie sprawdza uprawnień użytkownika na poziomie funkcji przyjmującej identyfikator obiektu, a te tworzone są w przewidywalny sposób, to znajomość identyfikatora będzie wystarczająca, by móc wykonać takie same operacje na obiekcie jak uprawniony użytkownik.

OWASP Top10 Web

Security Misconfiguration – niepoprawna konfiguracja - błędy konfiguracji zabezpieczeń mogą wystąpić w każdej warstwie aplikacji – nie tylko w jej własnym kodzie, ale też w innych elementach składających się na całość systemu, m.in. w użytych przez programistów bibliotekach i frameworkach, silnikach baz danych, serwerach aplikacyjnych czy urządzeniach sieciowych. Atakujący wykorzystuje zwykle domyślne konta, nieużywane strony, niezałatane podatności lub niezabezpieczone pliki i katalogi, by uzyskać nieautoryzowany dostęp do danych. Może się zdarzyć, że umożliwi mu to całkowite przejęcie kontroli nad zaatakowanym systemem.

Sensitive Data Exposure – nieodpowiednie zabezpieczenie poufnych danych - omawiając to zagrożenie, należy przede wszystkim wspomnieć o niewystarczających zabezpieczeniach kryptograficznych i niewłaściwym zabezpieczeniu wymiany danych. Wciąż wiele aplikacji przechowuje poufne dane (takie jak hasła użytkowników czy numery kart kredytowych), używając błędnie zaimplementowanej enkrypcji lub hashowania bez tzw. salta. W wyniku ataku może dojść do kradzieży takich danych i ich ujawnienia. Równie często aplikacje przesyłają w sieci dane, nie dbając o ich poufność i integralność. Mogą np. stosować wygasłe certyfikaty lub zbyt słabe algorytmy szyfrowania, co stwarza szerokie pole do nadużyć.

OWASP Top10 Web

Missing Function Level Access Control – nieodpowiednia kontrola uprawnień – użytkowników - aplikacje często obsługują zapytania do stron bez odpowiedniej walidacji. Niesprawdzanie, czy dana osoba powinna mieć dostęp do żądanej strony, pozwala atakującemu na wykonywanie akcji bez uwierzytelnienia lub z prawami innego użytkownika. Głównym celem tego typu ataków są oczywiście funkcje administracyjne.
Cross-Site Request Forgery (CSRF) – fałszowanie żądań - podatność ta często bywa mylona z XSS, ponieważ tak jak ona pozwala zaatakować przeglądarkę użytkownika, nie część serwerową aplikacji webowej. W tym przypadku celem atakującego jest wykorzystanie uprawnień ofiary do wykonania interesujących go nieautoryzowanych akcji. Odbywa się to dzięki podmienionym zapytaniom HTTP. Powodzenie ataku zależy od tego, czy atakujący jest w stanie przewidzieć, jak powinno wyglądać żądanie, które zostanie zaakceptowane przez serwer.

OWASP Top10 Web

Using Components with Known Vulnerabilities – używanie komponentów ze znanymi podatnościami - zdecydowana większość powstających obecnie aplikacji bazuje na gotowych już bibliotekach i frameworkach, które – jak każde oprogramowanie – mogą mieć błędy. W teorii można temu zaradzić, instalując udostępniane przez producentów poprawki. Często jednak okazuje się, że zaktualizowane komponenty nie będą współdziałać z tymi, które nie otrzymały łatek. W efekcie aplikacja pozostaje niezałatana, co pozwala na przeprowadzanie mniej lub bardziej wyrafinowanych ataków.
Unvalidated Redirects and Forwards – nieodpowiednia walidacja przekierowań - Ostatnie zagrożenie w zestawieniu OWASP dotyczy sytuacji, w których aplikacje webowe przekierowują użytkownika na inne strony, wykorzystując niezaufane dane. Przy braku odpowiedniej walidacji atakujący może dodać do oryginalnego odnośnika ciąg znaków, który zaprowadzi ofiarę na stronę ze złośliwym oprogramowaniem albo wyłudzającą poufne dane. Powyżej opisane błędy należą do najbardziej krytycznych i najczęściej wykorzystywanych, dlatego warto je mieć na uwadze, tworząc i zabezpieczając własne aplikacje internetowe.

OWASP Testing Guide

Obszerne opisy testowania aplikacji zarówno w ujęciu **black box** jak i **white/grey box**

Dobrze się czyta

Dzieli przeprowadzane testy na 2 fazy: **pasywną i aktywną**

Stanowi kompendium wiedzy o testach bezpieczeństwa i poza metodologią wykonywania testów może być źródłem szerokiej wiedzy z zakresu testów.

https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents

Penetration Testing Execution Standard

Zwięzłe opisy zagrożeń i elementów istotnych podczas testów

- Dzieli testy na 7 etapów:
  - ✔ **Przygotowanie**
  - ✔ **Gromadzenie informacji**
  - ✔ **Modelowanie zagrożeń**
  - ✔ **Analiza podatności**
  - ✔ **Eksploitacja**
  - ✔ **Post-eksploitacja**
  - ✔ **Raportowanie**
- **Niektóre rozdziały nie są ukończone!**

http://www.pentest-standard.org/index.php/Main_Page

Terminologia

ATAK

- *Wektor ataku*: czynnik, który umożliwia przeprowadzenie ataku (jeżeli np. atakujemy **aplikację internetową**, to wektorem jest np. **framework, który wykorzystuje ta aplikacja**)
- *Exploit*: wykorzystanie istniejącej w oprogramowaniu podatności w celu zaburzenia działania aplikacji lub wyrządzenia szkód użytkownikom aplikacji

CEL

- *Powierzchnia ataku*: Opisuje, co potencjalnie jest narażone na atak (jeżeli np. wystawiamy do sieci 10 portów serwera, to powierzchnią ataku jest te 10 portów.)
- *Podatność*: słaby punkt aplikacji, który może zostać wykorzystany w ataku (np. **XSS**, czy **nieaktualny Windows** z luką EthernalBlue)

Terminologia (CIA)

CONFIDENTIALITY

- **poufność**

czy odpowiednie osoby mają dostęp do odpowiednich danych?

INTEGRITY

- **integralność**

czy dane są spójne i godne zaufania?

AVAILABILITY

- **dostępność**

czy aplikacja jest dostępna dla uprawnionych użytkowników (czy nie jest awaryjna)?

Terminologia (AAA)

AUTHENTICATION

- **uwierzytelnienie**

kim jesteś?

AUTHORIZATION

- **autoryzacja**

czy masz prawo do tego działania

ACCOUNTING

- **rozliczanie**

jak wykorzystać te zasoby?

Przygotowanie i przeprowadzenie testów bezpieczeństwa - przykład

1. Przygotowanie środowiska lub ustalenie z administratorem, czy testy mogą być wykonywane na zwykłym środowisku testowym
2. Ustalenie trybu testów
3. Uzyskanie dostępów do kont z odpowiednimi zestawami uprawnień
4. Identyfikacja potencjalnych zagrożeń
5. Weryfikacja i próba wykorzystania podatności do przeprowadzenia ataku
6. Stworzenie raportu z analizą krytyczności zagrożeń i sugerowanymi poprawkami

PYTANIA?

kowal.radek@gmail.com