Krzysztof Jadczyk

PASJA TESTOWANIA

Wydanie II rozszerzone

Helion

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Helion SA dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Helion SA nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Redaktor prowadzący: Grzegorz Krzystek
Projekt okładki: Studio Gravite

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/paste2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-7639-7

Copyright © Krzysztof Jadczyk 2021

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

SPIS TREŚCI

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Czego jeszcze się nauczyć?

Struktura zespołu — role

Agile

Praca testera w agile

SQL

API testing

Angielski

Kup książkę Poleć książkę

W tym rozdziale znajdziesz informacje na temat kolejnych zagadnień, które moim zdaniem warto zgłębić przed wybraniem się na pierwszą rozmowę rekrutacyjną. Ta wiedza może być ważna z co najmniej kilku powodów. Z jednej strony pozwoli Ci lepiej zrozumieć to, o czym będą opowiadać rekruterzy. Z drugiej zaś wiedza ta może być konieczna, by odpowiedzieć na część pytań, które otrzymasz.

Rozmowy rekrutacyjne wyglądają różnie w zależności od firmy. Zresztą tak samo jest z całym procesem rekrutacyjnym. Część elementów zwykle się powtarza.

Rozmowa przeważnie zaczyna się od tego, że rekruterzy opowiadają trochę na temat firmy, jej historii, struktury, projektów, tym, co jest u nich fajne, itp. Następnie kandydat ma opowiedzieć coś o sobie, swoich dokonaniach, dotychczasowym doświadczeniu. Zdecydowanie jest to część, którą warto przemyśleć i przygotować sobie przed spotkaniem. Później zazwyczaj jest część związana z różnymi zadaniami praktycznymi albo testami — ta część bardzo często zależy od firmy, do której aplikujemy.

Rozmowa kończy się zwykle standardowymi pytaniami na temat oczekiwań finansowych oraz dostępności kandydata. To jednak nie jest regułą, bo mogą one pojawić się dużo wcześniej, np. podczas krótkiej rozmowy telefonicznej, która pomaga przyśpieszyć cały proces rekrutacyjny.

Jak wspomniałem powyżej, warto przygotować się do opowiadania o sobie. Kolejnym elementem, o który należy zadbać przed spotkaniem, jest lista pytań, które możesz zadać w trakcie spotkania. Pamiętaj, że rozmowa rekrutacyjna to rodzaj dialogu. Wykorzystaj tę okazję, by wypytać o interesujące Cię tematy i dowiedzieć się jak najwięcej, by później podjąć właściwą decyzję.

STRUKTURA ZESPOŁU — ROLE

Dla osób, które zaczynają swoją przygodę nie tylko z testowaniem, ale i z samym IT, pewną trudnością może być mnogość ról, stanowisk osób, z którymi przyjdzie im pracować. Poniżej opiszę w kilku zdaniach najważniejsze z nich.

Kup książkę Poleć książkę

To może pomóc zrozumieć, które osoby za co są odpowiedzialne w projekcie. Możesz też tę wiedzę wykorzystać podczas rozmowy, by dowiedzieć się, jakie role występują w danej firmie, do której aplikujesz. Oczywiście każda firma ma swoje podejście do realizacji projektów, więc struktura zespołów może wyglądać różnie. Tym bardziej warto dowiedzieć się, jak to wygląda, ponieważ brak którejś z ról może oznaczać, że zadania, które powinny być wykonywane przez innych członków zespołu, mogą stać się Twoim udziałem. Poniżej wymieniam najczęściej występujące role, z którymi ja się spotykałem.

Role w zespole

- **Tester/quality engineer** — odpowiedzialny za przeprowadzenie testów w projekcie. Zwykle pracuje zgodnie ze zdefiniowanym procesem lub podejściem, które opisuje sposób testowania. To podejście może się różnić w zależności od projektu. Dobrze, gdy blisko współpracuje z programistami oraz analitykami.
- **Test manager** — odpowiedzialny za zdefiniowanie podejścia do testów w danym projekcie. Zarządza zespołem testerów, monitoruje postęp prac, kontaktuje się z klientem, przekazując informację na temat statusu testów oraz bieżących ryzyk. Zwykle rola ta pojawia się w większych projektach lub zespołach.
- **Programista/deweloper** — osoba odpowiedzialna za implementację danego rozwiązania, które później będzie testowane. Programiści piszą także testy automatyczne (np. *unit test*), a kod przechodzi proces przeglądu (*code review*) przez innego programistę. Dobry deweloper sprawdza także kilka podstawowych ścieżek, zanim przekaże swoje zmiany do testów. Robi to, by upewnić się, czy to, co zakodował, rzeczywiście działa.
- **Analityk biznesowy/business analyst** — zbiera wymagania od klienta i pomaga zespołowi zrozumieć cel biznesowy projektu oraz poszczególne wymagania, których realizacja ma pomóc w osiągnięciu tego celu.

Kup książkę Poleć książkę

- **Kierownik projektu/project manager** — czuwa nad całym projektem, terminami, wspiera zespół, pomagając usuwać ewentualne przeszkody, które pojawiają się w trakcie trwania projektu. Pilnuje także budżetu, co dla zespołu zwykle oznaczać może regularne rozliczanie czasu pracy, jaki poświęca na wykonanie poszczególnych zadań. Pomaga to w śledzeniu zużycia budżetu oraz postępu prac.
- **Architekt** — odpowiedzialny za zdefiniowanie architektury całego rozwiązania. Zwykle blisko współpracuje z zespołem deweloperskim i wspiera programistów w celu uzyskania finalnego rozwiązania.
- **UX** — na bazie informacji pozyskanych od użytkowników projektuje makiety aplikacji tak, by były jak najbardziej użyteczne. Aplikacje webowe mogą wymagać różnych makiet na desktopy oraz urządzenia mobilne. UX przeprowadza testy z użytkownikami końcowymi, by upewnić się, że rozwiązanie jest dobrze zaprojektowane od strony interfejsu użytkownika i jest użyteczne.
- **UI deweloper** — osoba odpowiedzialna za odpowiednie ostylowanie (kolory, wielkości czcionek itp.) aplikacji, by było ono zgodne z makietami zaprojektowanymi przez UX.

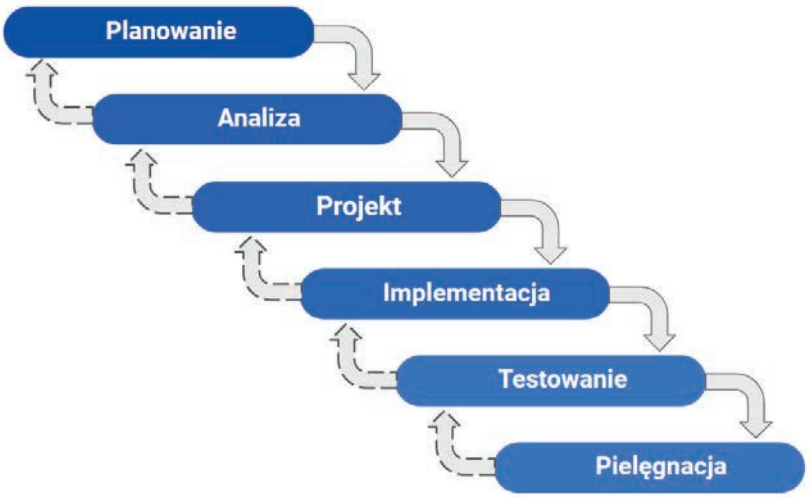
AGILE

Tego punktu nie mogło zabraknąć. Jest to teraz gorący temat. Każda firma aktualnie deklaruje, że pracuje agilowo. Dlatego konieczne jest zapoznanie się chociaż z podstawowymi pojęciami omawianego terminu.

Agile jest odejściem od bardzo popularnego wcześniej kaskadowego modelu (*waterfall*) wytwarzania oprogramowania. W podejściu tym aplikacja wytwarzana jest w fazach, które następują po sobie. Kolejna faza może się rozpocząć dopiero po zakończeniu poprzedniej. Warto mieć świadomość, że niektóre z faz mogą trwać tygodniami, a nawet miesiącami, a cały projekt

Kup książkę Poleć książkę

ponad rok. To powoduje, że zebranie informacji na temat wytworzonego oprogramowania następuje bardzo późno w procesie. W takim wypadku wprowadzenie zmian jest też niezwykle kosztowne. Zmiany wymagań lub ich doprecyzowanie, nawet w trakcie trwania projektu, zdarzają się często. Nie inaczej było w czasach, gdy *waterfall* przeżywał lata świetności. Wtedy jednak reakcja na takie zmiany nie była możliwa w krótkim czasie, zwłaszcza w późniejszych fazach projektu. Efektem tego był duży odsetek projektów, które kończyły się niepowodzeniem.



I tu wkracza Agile, cały ubrany na biało, ze swoim *Agile manifesto* (diagram modelu kaskadowego oraz *Agile manifesto* zaczerpnięte z Wikipedii):

- **ludzie i interakcje** ponad procesy i narzędzia,
- **działające oprogramowanie** ponad obszerną dokumentację,
- **współpraca z klientem** ponad formalne ustalenia,
- **reagowanie na zmiany** ponad podążaniem za planem.

Według Manifestu agile docenia się to, co wymieniono po prawej stronie, jednak bardziej ceni się to, co po stronie lewej. Zgodnie z tym podejściem ludzie pracują ze sobą bardzo blisko. W idealnym przypadku cały zespół (programiści, testerzy, analityk) pracuje w jednym

Kup książkę Poleć książkę

pokoju, co pozwala na częstą interakcję. Bliska współpraca przyśpiesza wyjaśnianie wątpliwości, umożliwia częste zderzenie różnych punktów widzenia, co pomaga w wykrywaniu i rozwiązywaniu problemów na wczesnym etapie.

Zespół skupia się na częstym dostarczaniu działającego rozwiązania w sposób przyrostowy. Każdy przyrost powinien wiązać się z wartością biznesową. Przygotowywanie dokumentacji jest ograniczone. Stawia się też duży nacisk na bliską współpracę z klientem i częste demonstrowanie efektów pracy (spotkania demo). To z kolei pozwala na skrócenie pętli informacji zwrotnej i szybkie wprowadzanie zmian w wytwarzanej aplikacji. Dokonywanie zmian w tym podejściu jest też dużo tańsze. Dzięki temu, że klient znacznie częściej widzi postępy prac i może je korygować, projekty w tym podejściu mają dużo większe szanse powodzenia (potocznie mówiąc: częściej są dowożone).

W agile pojawiają się nowe role, które zostały opisane poniżej.

- **Product Owner** — często rolę tę pełni przedstawiciel klienta. Nadaje on kierunek pracy zespołu deweloperskiego poprzez określanie priorytetów zadań, którymi zespół powinien się zająć. To on przedstawia wizję dla rozwoju produktu. Zbiera też informacje od przedstawicieli biznesu, ekspertów dziedzinowych w celu rozwoju aplikacji. Przekazuje także feedback na temat aplikacji do zespołu deweloperskiego. Product Owner zarządza **backlogiem** zadań, czyli rejestrem wszystkich wymagań do realizacji. Jest też osobą podejmującą decyzje.
- **Scrum Master** — osoba dbająca o to, żeby członkowie zespołu deweloperskiego rozumieli, czym jest scrum, a także, żeby scrum był stosowany. W praktyce wspiera zespół, by ceremonie scrumowe odbywały się zgodnie ze swoim przeznaczeniem. Wspiera także osobę w roli Product Owner poprzez pomoc w zarządzaniu backlogiem oraz w przygotowaniu planu rozwoju aplikacji. Usuwa także przeszkody, które pojawiają się w pracy zespołu deweloperskiego. Może moderować spotkania zespołu, jeśli istnieje taka potrzeba.

Kup książkę Poleć książkę

Pełni też ważną funkcję na poziomie całej organizacji. W ramach tego obszaru przede wszystkim pomaga pracownikom zrozumieć scruma. Jest również zaangażowany w proces wdrażania scruma w organizacji.

W agile praca prowadzona jest zazwyczaj w sprintach (1 – 4 tygodnie), a aplikacja rozwijana jest w sposób iteracyjny. Według założeń każdy sprint powinien kończyć się dostarczeniem kolejnej wersji aplikacji, która zawiera wartość dodaną z biznesowego punktu widzenia. W praktyce różnie to jednak wygląda. Często zależy to od ustaleń z klientem. W przypadku krótkich projektów, które trwają kilka, kilkanaście tygodni, sprinty mogą kończyć się tylko prezentacją wykonanej pracy podczas sesji *Demo* z klientem. W takim przypadku przekazanie działającej wersji może odbyć się dopiero po zakończeniu całości prac.

Każdy sprint zaczyna się od sesji **Sprint Planning**, podczas której zespół planuje pracę na najbliższy sprint. Dobiera odpowiednią liczbę historyjek użytkownika (*user story*), estymuje je oraz zobowiązuje się (commitment zespołu) do realizacji przed końcem sprintu.

Sesja planowania poprzedzona jest zwykle spotkaniem (**Refinement Meeting**), podczas którego zespół omawia poszczególne historyjki użytkownika w celu zrozumienia wymagań. Sprawdza też jakość ich przygotowania do realizacji (spełnienie *Definition of Readiness* — DoR) oraz przygotowuje wstępną estymację w story pointach. W trakcie tego spotkania omawia się historyjki użytkownika będące kandydatami do realizacji w kolejnej iteracji. Wybierane są one z backlogu.

Każdego dnia zespół spotyka się (**Daily Meeting**), by omówić postęp prac i ewentualne problemy.

Sprint kończy się przeglądem zrealizowanych zadań (**Sprint Review**) oraz omówieniem tego, jak wyglądała praca w ostatniej iteracji (**Retrospective Meeting**). Ostatnie spotkanie pomaga zidentyfikować pola do dalszych usprawnień oraz zdefiniować działania mające na celu ich wdrożenie. Efekt pracy zespołu jest także prezentowany klientowi podczas **Demo Session**

Kup książkę Poleć książkę

w celu zebrania opinii od klienta, które mogą wpłynąć na dalszą realizację projektu. Spotkanie to ma też na celu otrzymanie od klienta potwierdzenia odbioru wykonanych zadań.

PRACA TESTERA W AGILE

Tester w projekcie prowadzonym w metodyce agile powinien być zaangażowany od samego początku trwania projektu. Uczestniczy on we wszystkich ceremoniach (spotkaniach) scrumowych opisanych powyżej.

Jak wygląda praca testera w agilowych projektach? Przecież na początku sprintu nie ma nic do testów. Czy to znaczy, że tester się nudzi?
Nie do końca tak jest. Przez większość czasu są zadania, które można realizować. Zgadza się, że na początku wiele rzeczy do testów może nie być dostępnych. Rzeczywistość jednak pokazuje, że nie zawsze sprinty kończą się sukcesem. Czasami część historyjek przechodzi na kolejną iterację i można je testować. Jeśli jednak będziesz miał przyjemność pracować w zespole, który „dowozi" wszystko na koniec sprintu, to jednym z pierwszych zadań może być pisanie przypadków testowych.

Tutaj mała dygresja. Jest jeszcze jedno, często stosowane spotkanie. Nazywane jest różnie, albo **Triangle meeting**, albo **Tri Amigos**. W tym spotkaniu uczestniczą trzy osoby, co może zresztą sugerować jego nazwa. Spotyka się programista, tester oraz analityk, by przedyskutować dane *user story*, nad którym zaczyna się praca. Ważnym aspektem jest przedyskutowanie wszelkich wątpliwości i ich rozwianie. Dodatkowo można przedyskutować przypadki testowe, które będziemy chcieli wykonać w czasie testów. To już może naprowadzić programistów na miejsca o szczególnym ryzyku, które warto pokryć testami automatycznymi niskiego poziomu, czyli przez unit testy.

Kup książkę Poleć książkę

Po tym spotkaniu można się zabrać za pisanie przypadków. Inną aktywnością, która często pojawia się na początku iteracji, jest pisanie lub aktualizowanie testów automatycznych, które wykorzystują warstwę graficzną (UI — *user interface*). Dobrą praktyką jest też stosowanie tak zwanych **Triangle meeting zamykających**. To spotkanie odbywa się na koniec developmentu danej *user story*, ale zanim kod zostanie przekazany do testowania. Skład jest taki sam jak w przypadku pierwszego spotkania. Tym razem jednak pokazywany jest rezultat pracy programistów. Celem jest upewnienie się, że wszystko zostało zaimplementowane, i zweryfikowanie, czy z grubsza wygląda to poprawnie. Czasami może się okazać, że wprowadzone zostały dodatkowe zmiany, które nie były wcześniej zakładane. Trzeba będzie je zatem przetestować, by upewnić się, że nie mają one negatywnego wpływu na działanie aplikacji. Ten typ spotkania pozwala także wykryć pewne błędy, zanim kod zostanie formalnie przekazany i wdrożony w środowisku testowym.

Po pewnym czasie powinny pojawiać się funkcjonalności do testów. Wtedy zwykle jest sporo pracy. Poza wykonywaniem test case'ów oraz oznaczaniem ich statusów w narzędziu do zarządzania testami dochodzi kolejna aktywność, czyli zgłaszanie defektów. To uruchamia koło ich naprawiania (*bugfixing*) oraz weryfikacji (*bug retest*). Zwykle naprawa bugów jest priorytetowym zadaniem, którym deweloperzy powinni się szybko i sprawnie zająć. Ma to przyspieszyć zamykanie kolejnych *user story* zamiast rozgrzebywania wszystkich naraz. Iteracja powinna zakończyć się działającym produktem z dostarczoną wartością biznesową.

Koniec sprintu może być gorącym okresem dla testerów, bo pracy lawinowo przybywa. Coraz więcej funkcjonalności trafia do testów. Sprint może skończyć się także krótką fazą **testów regresyjnych**, by upewnić się, że wcześniejsze funkcjonalności działają bez zarzutu po dodaniu nowych fragmentów kodu.

Kup książkę Poleć książkę

Zwykle na koniec sprintu pojawia się też potrzeba przygotowania raportu z testów. W tej aktywności mogą okazać się pomocne metryki opisane w innym rozdziale. Dodatkowo warto dodać informacje o ryzykach i liście nierozwiązanych błędów (**Known issue**). Raport powinien zawierać wszystkie potrzebne informacje, które będą pomocne w podjęciu decyzji o wdrożeniu aplikacji w środowisku produkcyjnym lub w rozpoczęciu kolejnej fazy testów.

Co poza sprintami?
Taka praca zorganizowana w sprintach zwykle nazywana jest **Development Phase**. W niektórych projektach mogą pojawić się kolejne fazy testów, które nie muszą być podzielone na sprinty. Mowa tu o takich testach jak testy integracyjne systemów (SIT — *System Integration Tests*), testy akceptacyjne (UAT — *User Acceptance Tests*), a także faza testów niefunkcjonalnych (*performance*, *security*).

SQL

SQL to temat, którego nie pomijałbym w trakcie przygotowywania się do zawodu testera. Wynika to z pewnej prostej przyczyny. W trakcie swojej dotychczasowej kariery przeszedłem przez wiele różnych projektów w naprawdę różnych technologiach. Prawie w każdym z nich była jakaś baza danych. Do pewnego momentu można testować bez tej wiedzy, ale raczej nie trwa to zbyt długo. Jest to zresztą temat, który często pojawia się podczas rozmów rekrutacyjnych, więc warto się do tego przygotować. Nie mówię tutaj o bardzo zaawansowanej wiedzy, ale umiejętności takie jak sprawdzanie zawartości bazy danych, modyfikacja rekordów, ich dodawanie lub usuwanie są absolutnie niezbędne.

Kup książkę Poleć książkę

Nie będę się tutaj rozpisywał o szczegółach. Ten rozdział jest zajawką do dalszego, samodzielnego zgłębiania wiedzy. Należy poświęcić na to co najmniej kilka dni. Możesz zajrzeć na stronę *https://www.w3schools.com/sql/*, na której znajdziesz mnóstwo przykładów, a także zadania do wykonania. Oczywiście nie musisz opanować całej zawartości tego kursu. Warto jednak opanować:

- select (top, distinct);
- update;
- delete;
- insert;
- where;
- order by;
- and, or, not;
- like;
- join (inner join, left join, right join, full join);
- group by;
- having;
- create, drop.

API TESTING

API testing to temat, który w ostatnim czasie staje się coraz bardziej istotny do opanowania przez testerów. Wynika to z rosnącej popularności systemów opartych na mikroserwisach. Coraz więcej firm decyduje się na zmianę dotychczasowej architektury monolitycznej w swoich systemach na rzecz mikroserwisów. Ich zastosowanie pozwala między innymi na przyspieszenie procesu wdrażania (*release*) zmiany. W takim podejściu *release* może dotyczyć tylko danego komponentu (serwisu), a nie całego systemu. Nie ma

Kup książkę Poleć książkę

więc potrzeby przeprowadzania kosztownych testów regresji całego dużego systemu. Samo wdrożenie również jest prostsze, ponieważ dostarczany może być jeden z wielu komponentów, a nie całe rozwiązanie.

Nie sądzę, żeby od osób startujących na pozycję juniora oczekiwana była bardzo zaawansowana wiedza w tym zakresie. Ze względu na to, że jest to gorący temat, warto jednak znać kilka podstawowych pojęć, takich jak metody oraz kody odpowiedzi HTTP. Dodatkowo z pomocą po raz kolejny przychodzi Mr Buggy. Tym razem koniecznie spędź chwilę z wersją 7, w której organizatorzy zawodów postawili przed uczestnikami zadanie związane z testami API. Ale o tym nieco później.

Zacznijmy od tego, co oznacza API. Jest to skrótowiec od *Application Programming Interface*. API to zestaw reguł, które definiują komunikację pomiędzy programami komputerowymi. Kolejnym ważnym skrótowcem, który się pojawił, jest HTTP — *Hypertext Transfer Protocol*. HTTP jest wykorzystywany przy tworzeniu stron internetowych. Istnieje dziewięć metod HTTP, ale w pierwszej kolejności skup się na czterech z nich.

- GET — metoda wykorzystywana do pobierania danych.
- POST — metoda wykorzystywana do tworzenia nowych danych.
- PUT — metoda wykorzystywana do aktualizacji istniejących danych.
- DELETE — metoda wykorzystywana do usuwania danych.

Użycie tych metod skutkuje uzyskaniem jednego z wielu kodów odpowiedzi HTTP. Poniżej znajdziesz te najbardziej popularne[20].

Kod	Opis
200	OK
201	Created
400	Bad Request
401	Unauthorized
404	Not Found
500	Internal Server Error

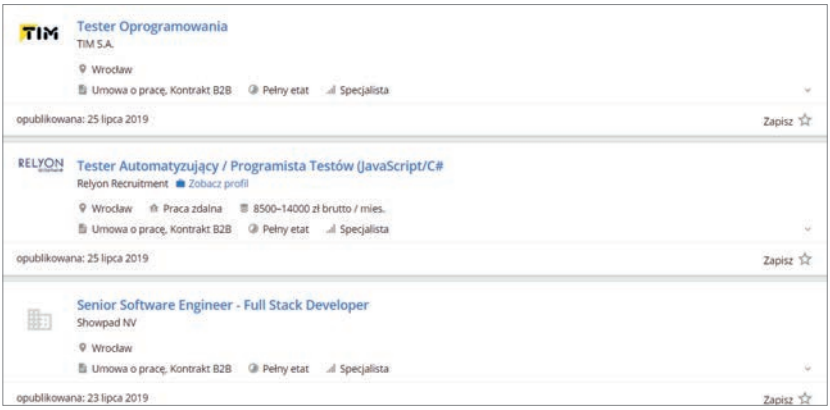
20 Więcej możesz poczytać na stronie: *https://pl.wikipedia.org/wiki/Kod_odpowiedzi_HTTP*.

Kup książkę Poleć książkę

Teraz wypada zadać sobie pytanie: Gdzie to wszystko można zobaczyć? Z pomocą przychodzą narzędzia deweloperskie dostępne w przeglądarce. W kolejnym rozdziale opisuję w kilku zdaniach, co możesz dzięki nim podejrzeć.

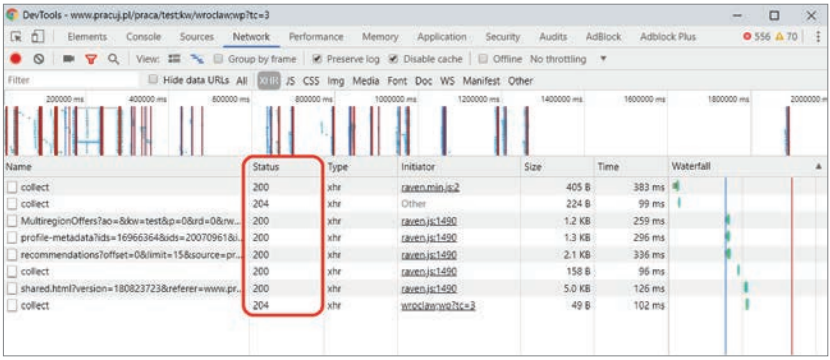
Tutaj chciałem skupić się jedynie na metodach i kodach odpowiedzi i pokazać Ci, jak do nich dotrzeć. Nie jest to specjalnie skomplikowane. Wystarczy w przeglądarce wcisnąć *F12* i przejść do zakładki *Network*. Upewnij się, że masz włączoną opcję zbierania logów w czasie poruszania się po stronie — pierwsza z ikon powinna być czerwona (zaznaczona na poniższym rysunku). Wtedy w zakładce *Network* zbierane będą requesty. Już z tego poziomu możesz zobaczyć, jaki jest kod odpowiedzi.

Sprawdźmy, jak to działa, na przykładowej stronie. Na potrzebę tego ćwiczenia wybrałem pracuj.pl. Po ustawieniu kilku filtrów i wyszukaniu dostałem listę ofert. Poniżej widać tylko część wyników z okna przeglądarki.

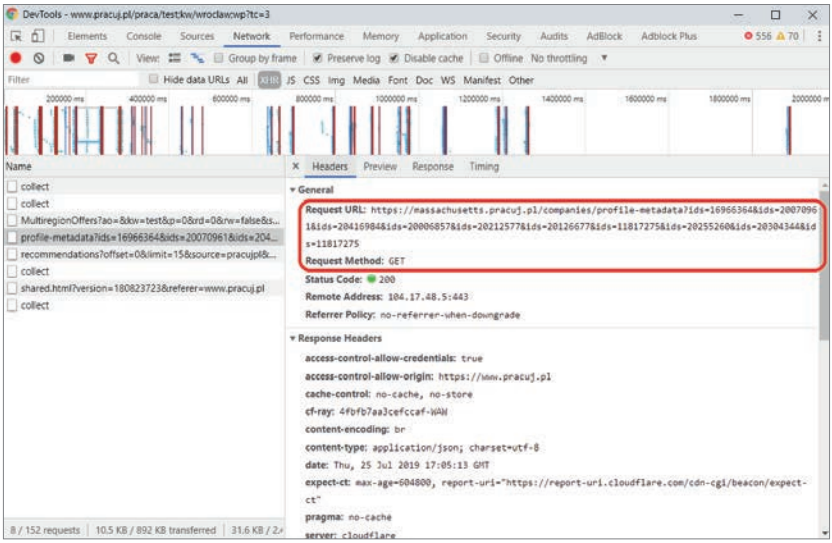


Kup książkę Poleć książkę

W oknie *devtools* widać requesty, które zostały wykonane wraz ze statusami.

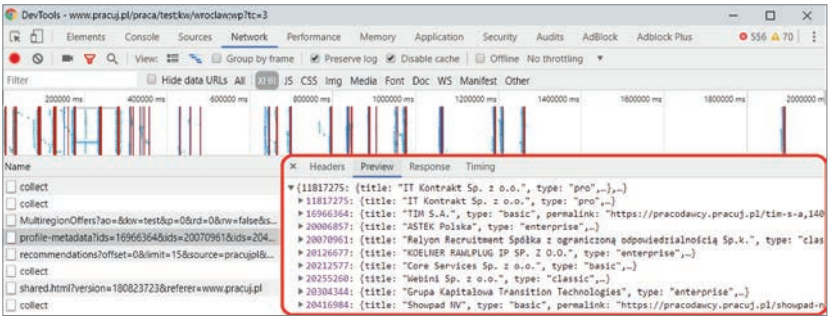


Po zaznaczeniu jednego z wierszy otwiera się okno z detalami. Tutaj znajdziesz też metodę HTTP, która została użyta (w tym przypadku jest to GET), oraz URL. Te dwie rzeczy przydadzą się w kolejnym kroku.

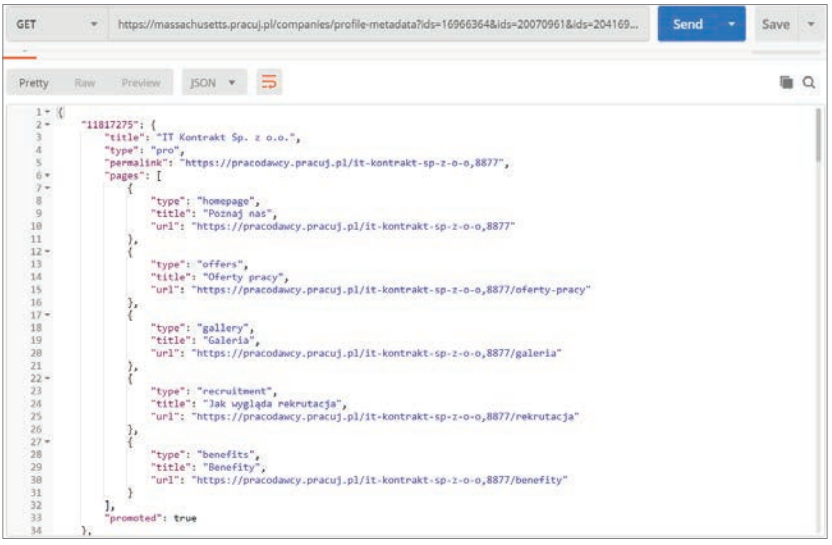


Kup książkę Poleć książkę

Po przejściu na zakładkę *Preview* możesz podejrzeć także odpowiedź (*response*) — obrazek poniżej. Prawda, że podobne do listy na pierwszym obrazku? Porównując powyższy i poniższy obrazek, widzimy, że *Id* poszczególnych ofert trafia jako parametr lub lista parametrów do URL.

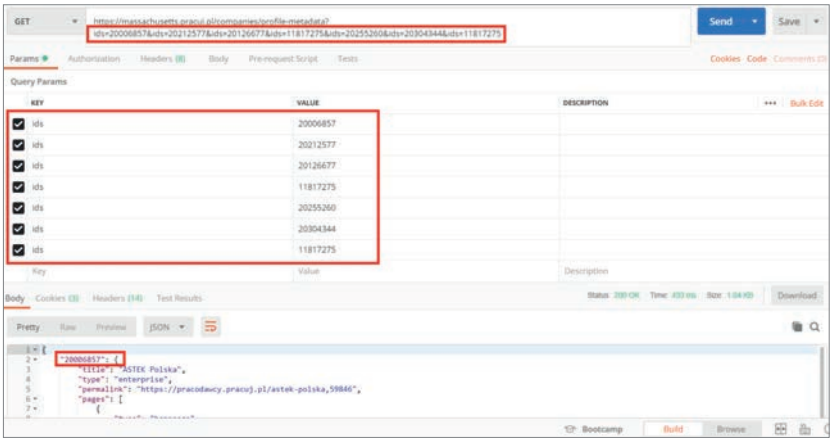


Teraz uruchamiam Postmana i dodaję nowy request. Wybieram metodę GET i kopiuję URL, który wcześniej zaznaczyłem na obrazku powyżej. Otrzymuję tę samą listę.

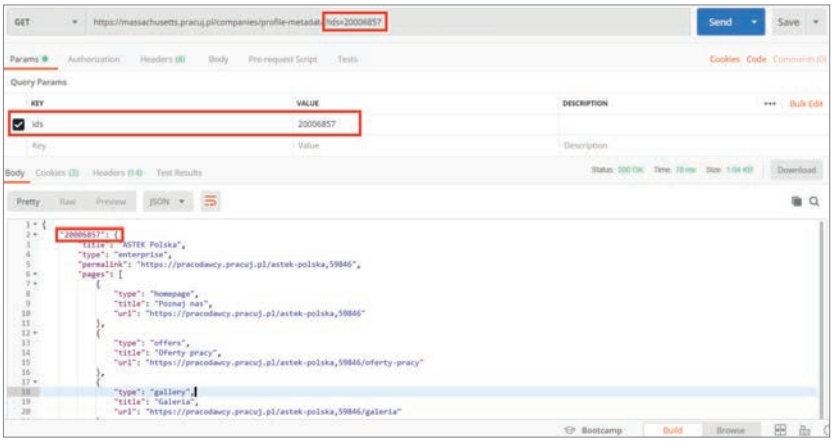


Kup książkę Poleć książkę

Tutaj dobrze widać, że *Id* oferty traktowane są jako parametry.



Teraz mogę ograniczyć listę *Id*, by w wyniku otrzymać dane dla pojedynczej oferty.



W celu dalszej nauki testowania API proponuję przejść do ćwiczenia z Mr Buggy 7, dostępnego w rozdziale „Zostań mistrzem testowania".

Kup książkę Poleć książkę


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion

PRZETESTUJ SIĘ W ROLI TESTERA

Pasja testowania to podręcznik przeznaczony dla osób, które stawiają pierwsze kroki w świecie IT i rozważają, czy praca testera, ważna i nieźle płatna, jest dla nich. Tester oprogramowania — to brzmi dobrze! Tylko na czym polegają jego zadania? Jakich umiejętności się od niego wymaga? Znajomość których narzędzi będzie oczekiwana? Skąd czerpać o nich wiedzę? I przede wszystkim: jak się przekonać, czy będzie się dobrym testerem?

Na te i inne pytanie odpowiada *Pasja testowania*. Dzięki tej książce poznasz teoretyczne podstawy pracy testera, zrozumiesz, na czym polega jego warsztat i z jakich elementów się składa, dowiesz się też, jak zabrać się do pracy testerskiej. Autor, praktyk z kilkunastoletnim stażem, wprowadzi Cię w te zagadnienia krok po kroku. Co więcej, obali najpowszechniejsze mity krążące w światku testerów i zwróci uwagę na klienta, czyli prawdziwego odbiorcę efektów Twojej pracy. Na deser zaś pozwoli Ci samemu się przetestować — proponuje bowiem szereg ćwiczeń, dzięki którym zostaniesz prawdziwym mistrzem testowania. Czego zresztą szczerze Ci życzy!

- **Opanuj podstawy**
- **Poznaj narzędzia**
- **Znajdź pracę w zawodzie**

Krzysztof Jadczyk — ekspert w dziedzinie testowania oprogramowania z kilkunastoletnim stażem. Brał udział w wielu projektach realizowanych w różnych technologiach, a jako lider budował zespoły testowe oraz rekrutował testerów. Jest twórcą *Bugfree blog* (*https://bugfreeblog.com/*) oraz aktywnym użytkownikiem serwisu LinkedIn.

Helion

helion.pl

HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

Sprawdź nasze szkolenia!

SZKOLENIA

AKADEMIA IT & BUSINESS

HELIONSZKOLENIA.PL

KOD KORZYŚCI
Sięgnij po więcej! ▶

ISBN 978-83-283-7639-7

9 788328 376397

INFORMATYKA W NAJLEPSZYM WYDANIU

Cena: 49,00 zł