

PATRONAT- INTIVE

Etap 1 QA

1. Czym się różnią testy funkcjonalne od нефункциональных?

Testy funkcjonalne	Testy нефункциональные
<ul style="list-style-type: none">- to inaczej testy wymagań funkcji np. systemu, podsystemu, modułu udokumentowanych lub znanych testerom- zawierają opis zachowania systemu „Co system robi?”- wykonuje je się na wszystkich poziomach testów- są mierzalne jeśli chodzi o pokrycie (wszystkie funkcje zostały przetestowane)- są podatne na automatyzację- tester ma dostęp do kodu testowanej aplikacji- testy są wykonywane przez osobę, która nie tworzyła aplikacji w oparciu o dokumentację oraz założenia funkcjonalne- pozwalają na wykrycie błędów związanych z np. brakiem implementacji funkcjonalności opisanych przez wymagania ale nie pozwalają precyzyjnie określić miejsca gdzie błąd występuje- przypadki testowe tworzone są pod kątem sprawdzenia czy aplikacja zachowuje się w oczekiwany sposób- testy funkcjonalne pokrywają funkcje dostępne dla użytkownika jak np. interfejs jak i operacje typu back-end (bezpieczeństwo czy, wpływ aktualizacji na działanie systemu)- często wykonywane pod koniec pod koniec cyklu wytwarzania oprogramowania ale mogą i powinny być uruchamiane znacznie wcześniej <p>Testowanie funkcjonalne obejmuje : testy jednostkowe, dymne, integracyjne, testy interfejsu i użyteczności, systemowe, regresji, alfa i beta, testy akceptacyjne użytkownika, testowanie metodami białej i czarnej skrzynki.</p>	<ul style="list-style-type: none">- może być wykonywane na wszystkich poziomach testów- w większości przypadków wykorzystują techniki czarnoskrzynkowe- właściwości нефункциональные opisują następujące cechy testowanego produktu: Wydajność (testy wydajności, obciążeniowe i przeciążające), użyteczności, ergonomii, współpracy (międzyoperacyjności) niezawodności, efektywności, przeznaczalności, możliwości procedury instalacyjnej- testy нефункциональные odpowiadają na pytania „Jak działa system?”

2. Co to są 'smoke testy' i 'testy regresji'? Kiedy je stosujemy?

Smoke testy

Termin "smoke test" funkcjonuje obecnie głównie w kontekście procesów wytwarzania oprogramowania, a zwłaszcza kontroli jakości. W kontekście testowania oprogramowania i złożonych systemów informatycznych należy rozumieć smoke test (przekładany czasem na "test dymny") jako:

- **test pobieżny** (będzie to często test przeszukujący "wszerz", a nie "w głąb"; często będzie to test najbardziej typowej ścieżki czynności, którą może przebyć potencjalny użytkownik),
- **test zajmujący niewiele czasu** (ma szybko udzielić informacji koniecznych do podjęcia decyzji, co zrobimy dalej w ramach testowania),
- **test poszukujący bardzo wyraźnych problemów** (test zdany pomyślnie powinien wykluczyć zachodzenie ewidentnej awarii na swojej ścieżce),
- **test dopuszczający do kolejnego etapu prac** (zwłaszcza w kontekście zaangażowania w tym kolejnym etapie znaczących zasobów).

Mogą być zaprojektowane zarówno we wczesnych etapach wytwarzania oprogramowania, kiedy ustalane są najistotniejsze ścieżki przechodzenia przez aplikację - możemy na przykład wybrać najbardziej priorytetowe albo ryzykowne przypadki użycia (use case'y) i zaprojektować smoke test, który przejdzie przez te z najistotniejszych dla klienta, które pokrywają najwięcej obszarów aplikacji za jednym zamachem.

Nie wyklucza to możliwości tworzenia dodatkowych smoke testów później w procesie wytwarzania, gdy na przykład wydarzenia podczas projektu wskazują na prawdopodobieństwo zachodzenia "krytycznej" awarii. Bardzo typowe będzie dla zespołu testowego wykonywanie smoke testu, który konkretnie ma upewnić zespół, że, przykładowo, po ostatnim wdrożeniu najnowszej wersji systemu, ta wyjątkowo groźna awaria w znanym nam miejscu nie występuje.

Testy regresji

upewnienie się, że aplikacja działa po dokonaniu w niej modyfikacji, poprawieniu błędów lub po dodaniu nowej funkcjonalności.

Cecha: powtarzalność

Co dają testy regresywne:

- Wyszukanie błędów powstałych w wyniku zmian kodu/środowiska.
- Ujawnienie wcześniej nie odkrytych błędów.
- Jest to dobry kandydat do automatyzacji ze względu na swoją powtarzalność.

Iteracyjne metodologie oraz krótkie cykle w których dostarczane są kolejne funkcjonalności powodują, że testy regresywne pozwalają upewnić się czy nowe funkcjonalności nie wpłynęły negatywnie na istniejące już i działające części systemu.

Testy regresywne mogą być przeprowadzane dla kompletnego produktu lub jego części. Jeżeli pomiędzy cyklami testów nie można przeprowadzić pełnych testów regresywnych aplikacji (koszty, czas), przypadki testowe, które włączamy do testu, powinny być dobierane na podstawie:

- Jakie błędy zostały poprawione, jakie rozszerzenia czy zmiany zostały wprowadzone
- Których obszarów aplikacji zmiany te dotyczą najbardziej
- Jaki jest wpływ wprowadzonych zmian na inne części systemu

Testy regresywne powinny być przeprowadzane po smoke testach lub testach typu sanity. Ten typ testów pozwala upewnić się czy otrzymana nowa wersja aplikacji jest “testowalna” i czy warto zaczynać z nią pracę.

3. Co jest celem testowania

Wykrywanie podstawowych błędów, by je usunąć,

-Sprawdzanie zgodności z innymi aplikacjami,

-Ułatwianie podjęcia decyzji interesariuszom o wypuszczaniu lub niewypuszczaniu produktu,

-Minimalizacja kosztów pomocy technicznej,

-Kontrola zgodności z wymaganiami i z prawnymi uregulowaniami.

- W testach akceptacyjnych głównym celem może być potwierdzenie, że system działa tak jak powinien oraz nabranie pewności, że spełnia wymagania.
- Testowanie pielęgnacyjne często zawiera testy sprawdzające, czy nie wprowadzono nowych usterek podczas wykonywania zmian. Głównym celem testowania produkcyjnego może być ocena atrybutów systemu takich jak niezawodność lub dostępność

4. Jak tester może się upewnić, że błąd został naprawiony?

Poprzez wykonanie Re-testu (powtórzenie testu, test potwierdzający) – powtórne wykonanie przypadku testowego, którego pierwotne wykonanie wykazało awarię (nieprawidłowe działanie) testowanego produktu. Celem re-testu jest sprawdzenie, czy defekt będący przyczyną awarii został usunięty.

5. Testujesz aplikację termometr która wykonuje pomiar temperatury. Co byś zrobił aby przetestować zachowanie aplikacji przy skrajnych wartościach -50C i 200C ? Wykonałabym test dla wartości -50C oraz 200C.

Przetestowałabym dla następujących wartości: -50 °C, 200 °C ewentualnie dla wartości brzegowych. Można wykorzystać symulator czyli urządzenie lub program albo system używany podczas testowania, który przy zdanym zbiorze wejść zachowuje lub działa tak jak dany system lub emulator (urządzenie, program komputerowy lub system, który przyjmuje takie same wejścia i generuje takie same wyjścia jak dany system. Są to wirtualne środowiska nie tylko rozszerzyć zakres badań na większej liczbie urządzeń, ale również to szybki i łatwy sposób, aby przetestować niewielkie zmiany. Dostępne są np. Apple iOS Symulator, Android Emulator, BlackBerrySymulatory itd. Można też wykorzystać w przypadku niektórych urządzeń „zdalne laboratorium” np. dedykowane dla firmy Samsung.

Następnie można przesłać dane stosując mockowanie, symuluje to wartości jakie przekazuje urządzenie, mierzące temperaturę. Mockowanie to tzw. „udawanie” – dzięki temu możemy przetestować wybrany komponent, dostarczając mu „udawane” zależności, przez co nie musimy np. tworzyć testowej bazy danych czy obawiać się, że jakieś dane zostaną zmienione podczas testów. Daje też wiele dodatkowych możliwości, które możemy wykorzystać w naszych testach mockowanie daje nam przewidywalny wynik działania konkretnego serwisu, niezależny od np. połączenia z innymi systemami, danych w bazie danych czy innych

okoliczności. Dzięki temu unikamy sytuacji w której testy dają różne wyniki w zależności od tego, kiedy, jak i na jakim urządzeniu zostaną uruchomione.

6. Ile przypadków testowych potrzeba, aby pokryć wszystkie możliwości?

Aby pokryć wszystkie możliwości potrzebne są 4 przypadki testowe

7. Dany jest input „wiek”, który przyjmuje wartości od 18 do 60. Twoim zadaniem jest przetestować go za pomocą techniki wartości brzegowych. Jakie wartości wpisujesz do inputu, podaj wszystkie liczby, które wpisujesz.

17, 18, 19, 59, 60, 61 zgodnie z teorią wartości brzegowych.

8. Dołączasz do projektu w trakcie developmentu aplikacji, do której nie ma dokumentacji. Jakie pytania zadasz analitykowi, zanim przystąpisz do testów logowania?

- Czy przeprowadzono walidację logowania? Czy jest określona ilość znaków minimalna i maksymalna. Czy nie przekracza np. maksymalnej ilości znaków
- Czy zostało sprawdzone użycie znaków takich jak apostrof, cudzysłów, średniki, ukośniki
- Czy sprawdzono logowanie pod kontem użycia białych znaków?
- Czy został stworzony przypadek testowy dla funkcji logowania np.

Kroki do wykonania:

podaj login i hasło [zgodnie ze zdefiniowaną walidacją]

Oczekiwany rezultat:

Poprawny login / hasło - zalogowany

Niepoprawny login / hasło - niezalogowany

9. Czym się różni metoda GET od POST?

Metoda GET- Żądanie zasobu od serwera w formie nagłówka i treści.

Używamy do pobierania zasobu lub jego wyświetlenia. Np. wyświetlanie formularza lub strony. Parametry można przekazywać jedynie przez adres (np. ?nazwa=wartosc&nazwa2=wartosc2). Parametry przekazywane metodą GET umieszczane są w adresie strony po znaku ? i rozdzielone znakiem &. Parametry są zwykle przekazywane w postaci par *nazwa=wartość*. Ponieważ w nazwie lub wartości parametru mogą pojawić się znaki, które nie są dozwolone w adresie lub mają w tym kontekście specjalne znaczenie, stosowany jest tak zwany *url encoding*. Dane są odkodowywane i prezentowane w łatwej do przeglądania i edycji postaci. W adresie strony do której nastąpiło odwołanie widoczny jest cały URL razem z parametrami przekazanymi w GET. Właśnie z powodu możliwości ujawnienia danych, nie tylko w nagłówku ale choćby w logach serwera WWW czy serwera proxy, nie należy stosować przekazywania parametrów metodą GET, a przynajmniej nie należy w ten sposób przekazywać istotnych danych jak na przykład hasła czy identyfikatory sesji.

Metody POST. przesłania danych zapisanych jako pary klucz-wartość do serwera (np. wysłanie formularza, gdzie kluczem jest nazwa danego pola a wartością wpisana przez nas wartość) – obie rzeczy możemy realizować pod tym samym adresem URL. W przypadku przekazywania parametrów metodą POST, nazwy i wartości parametrów nie są widoczne w adresie URL.

GET w niektórych przeglądarkach ma ograniczenie długości, oznacza to, że można w ten sposób wysłać tylko małą ilość danych. POST nie ma takich ograniczeń

W praktyce metody pozwalają nam ,rozdzielać' zapytania trafiające pod ten sam adres.

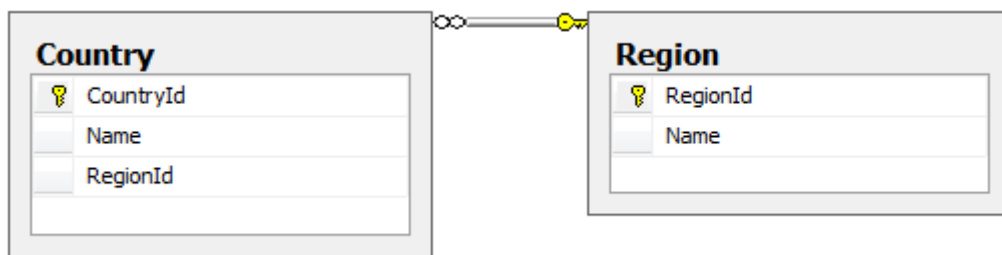
10. Czy HTTP jest protokołem zmiennostanowym?

HTTP zaliczane jest to protokołów bezstanowych- nie zachowuje żadnych informacji o poprzednich transakcjach z klientem (po zakończeniu transakcji wszystko "przepada"). Pozwala to znacznie zmniejszyć obciążenie serwera, jednak jest kłopotliwe w sytuacji, gdy np. trzeba zapamiętać konkretny stan dla użytkownika, który wcześniej łączył się już z serwerem.

11. Czym różni się LEFT JOIN od INNER JOIN?

System między innymi Left Join i Inner Join został stworzony w taki sposób ,aby wyrażać związki pomiędzy rekordami bez dublowania informacji. Na przykładzie SQL

Tabela Country zawiera ID , nazwę kraju oraz id regionu. ID regionu może być puste. Dany kraj może nie posiadać regionu.



SQL

```
INSERT INTO [dbo].[Region]
    ([RegionId]
    , [Name])
VALUES
    (1, 'Europe'), (2, 'Asia'), (3, 'Africa'),
    (4, 'South America'), (5, 'North America'), (6, 'Australia')

INSERT INTO [dbo].[Country]
    ([CountryId]
    , [Name]
    , [RegionId])
VALUES
    (1, 'Poland', 1), (2, 'Japan', 2),
    (3, 'Algeria', 3), (4, 'Argentina', 4),
    (5, 'New Zealand', NULL), (6, 'Federated States of Micronesia', NULL)
```

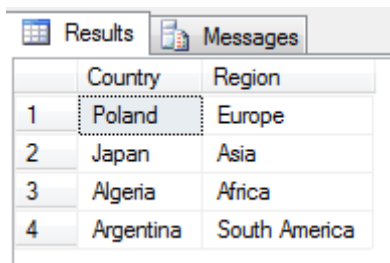
REGIONY : “5,North America” oraz “6,Australia” nie są powiązane z żadnym krajem.

KAJE : “5,'New Zealand'” i “6,'Federated States of Micronesia'” nie mają swojego regionu.

INNER JOIN zwraca wiersze , w których obie wartości w obydwu tabelkach się zgodziły.

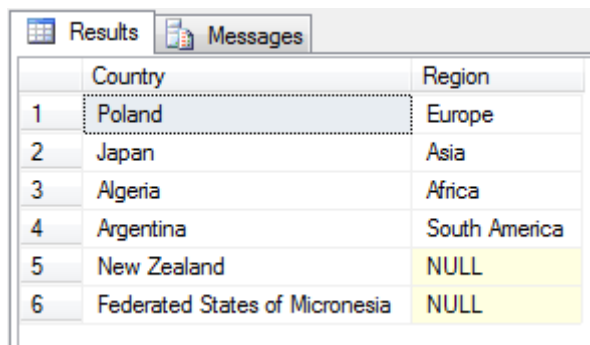
Wyniki tego zapytania powinny zwrócić tylko kraje powiązane z regionami jak i regiony, które mają przypisane kraje. Rekordy zostały powiązane po RegionId.

Kraje nie mające RegionID zostały zignorowane. Podobny los trafił na regiony, które nie zostały przypisane do żadnego kraju. Np. wpisujemy komendę (SELECT * FROM Table 1t INNER JOIN Table t2 ON t1.Colld = t2.Colld)



	Country	Region
1	Poland	Europe
2	Japan	Asia
3	Algeria	Africa
4	Argentina	South America

LEFT JOIN : na podstawie przykładu zwraca wszystkie wiersze tabelki po lewej plus wiersze, które mają uzupełnienie z prawej tabelki. Jeśli nie ma żadnych wartości odpowiadającej prawej tabelce jest zwracana wartość NULL. Np. wpisujemy komendę (SELECT * FROM Table t1 LEFT OUTER JOIN Table t2 ON t1.Colld = t2.Colld)



	Country	Region
1	Poland	Europe
2	Japan	Asia
3	Algeria	Africa
4	Argentina	South America
5	New Zealand	NULL
6	Federated States of Micronesia	NULL

12. W jakim katalogu, standardowo Linux trzyma pliki konfiguracyjne:

- a. /boot
- b. /var
- c. /etc
- d. /cfg

Odp. etc

13. Jak przetestowałbyś bashową komendę cp? (argumenty funkcji można pominąć)

Można przetestować komendę bash cp. Sprawdzając trzy warianty. W najbardziej podstawowym użyciu polega na podaniu dwóch argumentów, z czego pierwszy to plik (lub pliki), który ma zostać skopiowany, a drugi to miejsce do którego ma zostać skopiowany. Jeśli drugim argumentem jest nazwa pliku, to pierwszy plik zostanie skopiowany do drugiego (należy pamiętać, że podczas kopiowania nie wyskakuje żadne pytanie, czy na pewno chcemy plik podmienić/nadpisać, jeśli już o takiej samej nazwie istniał wcześniej).

```
$ cp [opcje] [-T] źródło cel
$ cp [opcje] źródło katalog
$ cp [opcje] -t katalog źródło
```

Sprawdzamy czy działa polecenie kopiowania plików

```
joanna@dd: ~/Desktop/test$ ls
kat1 kat2 kat3 plik1
joanna@dd:~/Desktop/test$ cp plik1 plik2
joanna@dd: ~/Desktop/test$ ls
kat1 kat2 kat3 plik1 plik2
joanna@dd:~/Desktop/test$[]
Czy działa kopiowanie plików do katalogu:
```

```
joanna@dd: ~/Desktop/test$ cp plik1 plik2 kat1/
joanna@dd: ~/Desktop/test$ cp plik* /home/joanna/Desktop/test/kat2
joanna@dd: ~/Desktop/test$ cp plik* ~/Desktop/test/kat3
joanna@dd: ~/Desktop/test$ ls kat*
kat1:
plik1 plik2
kat2:
plik1 plik2
kat3:
plik1 plik2
joanna@dd:~/Desktop/test$[]
```

```
Czy działa katalogu z zawartością: cp -r
joanna@dd: ~/Desktop/test$ cp -r kat1/ kat2/
joanna@dd: ~/Desktop/test$ cp -r plik1/ /home/joanna/Desktop/test/kat3
joanna@dd: ~/Desktop/test$ ls kat2/ kat3/
kat2:
kat1 plik1 plik2
kat3:
kat1 plik1 plik2
joanna@dd:~/Desktop/test$[]
```

Dodatkowo możemy sprawdzić kopiowanie z potwierdzeniem nadpisania jeśli chcemy mieć pewność że podczas kopiowania nie dojdzie do zastąpienia pliku bez powiadomienia. cp -i (interactive). Dostaniemy komunikat cp : overwrite 'plik2'? y

```
joanna@dd: ~/Desktop/test$ cp -i plik1 plik2
cp: overwrite 'plik2'? y
joanna@dd:~/Desktop/test/kat2$[]
```

