Praca inżynierska

Mikołaj Kuszowski

18 sierpnia 2024

Streszczenie

Next sections: 1. TDD - Test-driven development 2. CQRS

1 Spotkanie z biznesem

Pierwszym etapem stworzenia aplikacji jest konfrontacja IT z Biznesem. W naszym wypadku Biznesem mogą być klienci czyli zewnętrzne firmy lub inne działy wewnątrz firmy. Biznes przychodzi z problemem, przedstawia swoje cele i wymagania. Dział IT musi więc dobrze rozumieć cele i potrzeby biznesowe, natomiast biznes musi wiedzieć, czy organizacja dysponuje zasobami niezbędnymi do ich realizacji.[https://cyfrowa.rp.pl/technologie/art40278241-konieczna-wspolpraca-miedzy-it-i-biznesem] Łącznikiem między Binzesem, a IT jest analityk biznesowy, który ma wiedzę techniczną oraz jest w stanie w przejrzysty sposób zrozumieć wymagania biznesu.

Proces wytwarzania oprogramowania możemy rozpocząć od warsztatu o fachowej nazwie "Event Storming". Jest to proces na styku biznesu i produkcji oprogramowania. Takie spotkanie ma na celu przedstawienie przez bisnes jak działa albo jak ma działać domena. Do domeny nie zalicza się tylko aplikacja, ale też ludzie wykonujący jakiś proces czy inne zewnętrzne aplikacje. Dzięki poznaniu domemy można poznać wszystkie procesy występujące docelowo wokół naszej aplikacji co ułatwia zrozumienie całego procesu. Finalnie tylko fragment z przedstawionego podczas event stormingu procesu będzie należał do wytwarzanej aplikacji. Zaletą Event Stormingu jest również lepsze zrozumienie domeny przez biznes, który wcześniej mógł nie być świadomy, że oczywiste według nich procesy mogą być bardziej skomplikowane co mogłoby utrudnić pracę programistyczne. W event stormingu najważniejsze jest poznanie całości domeny, dlatego ważne jest, aby osoby biznesowe przedstawiły wszystkie swoje zadania. Zasadność tych zadań będzie dyskutowana w dalszej części. Podczas takiego spotkania uczestniczą:

- Eksperci domenowi - od strony biznesu
- Eksperci techniczni - np. analitycy biznesowi

Sama forma warsztatu wydaje się dość prosta – do dyspozycji mamy dużą tablicę czy ścianę oraz mnóstwo różnokolorowych karteczek. Każdy z uczestników identyfikuje zdarzenia („Domain Events"), które występują w trakcie działania programu – np. „Utworzono konto użytkownika", „Zamówiono towar", czy „Wygenerowano fakturę". Zdarzenia te zapisuje się na karteczkach i umieszcza na tablicy. [https://bulldogjob.pl/readme/event-storming-pierwszy-krok-do-ddd]. Celem takiego spotkania jest klarowne poznanie domeny (biznesu) i zrozumienie potrzeb oraz wymagań.

Wytwarzanie aplikacji zaczniemy od procesu Event Storimngu. Aby ją lepiej zrozumień zdefiniujmy co oznaczają poszczególne kolory karteczek.

- Żółty: Aktor (ang. Actor) - Admin, Klient, Sprzedawca
- Niebieski: Polecenie (ang. command) - planujemy coś zmienić
- Zielony: Model odczytu (ang. read model) - dane, które wykonują decyzję (np. dane z formularza)
- Fioletowy:
- Złoty: Zdarzenia domenowe (ang. domain event) - polecenie zakończone z sukcesem

Rysunek 1: Chaotyczna eksploracja

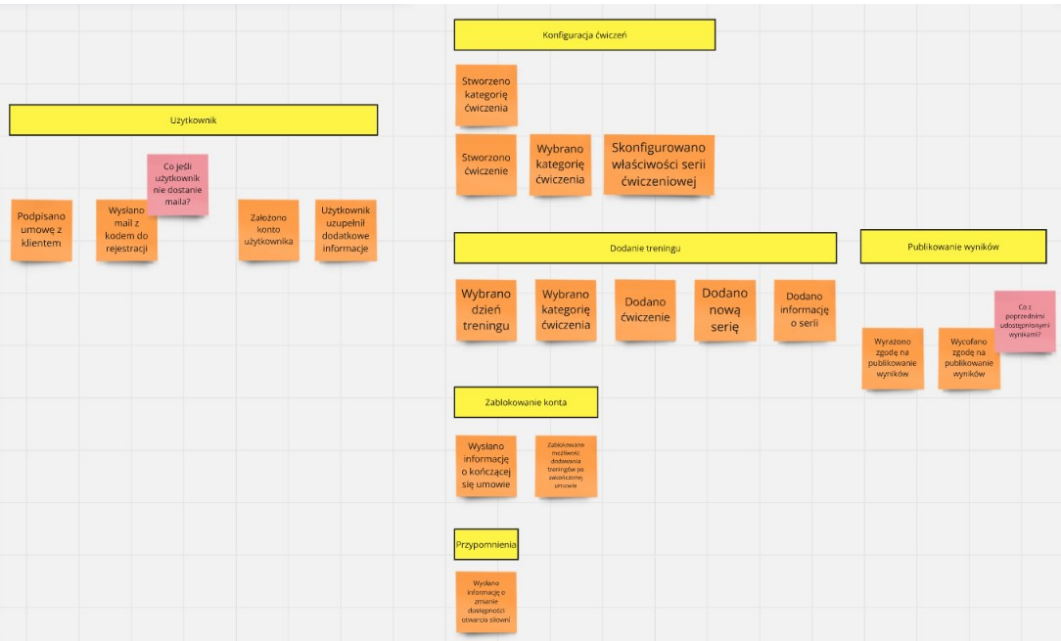
Karteczki opcjonalne

- Różowy: Problemy (ang. Issues)

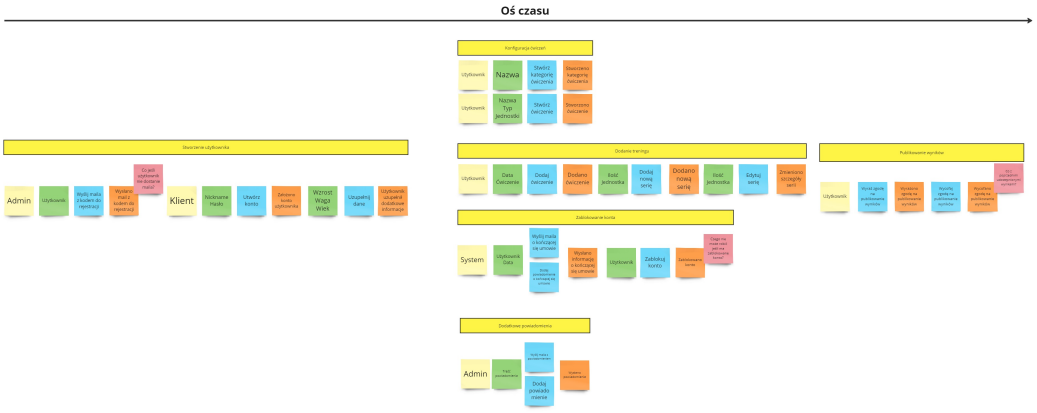
Event Storming możemy zacząć od procesu zwanego "Big Picture". Proces ten rozpoczynamy od zdefiniowania zdarzeń czyli pomarańczowych karteczek. Aby zdarzenie było prawidłowe musi spełniać następujące wymagania:

- musi być opisane w czasie przeszłym
- opisuje zmianę w systemie
- występuje w konkretnym momencie w czasie

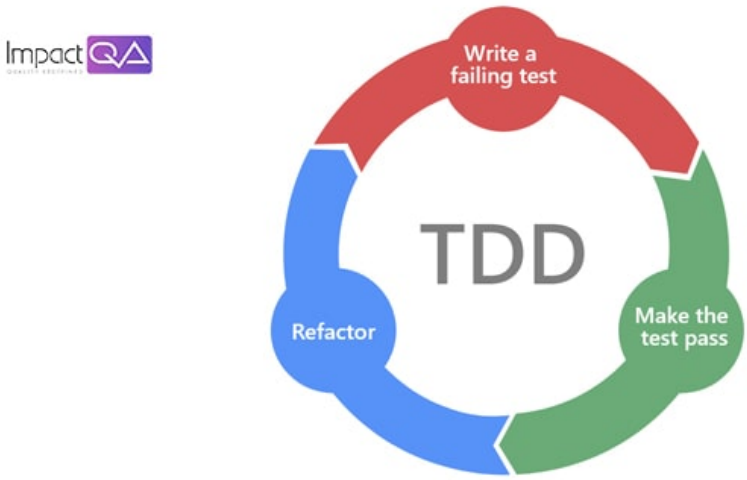
W początkowym etapie definiowania zdarzeń nie jest dla nas istotne miejsce ich występowania na osi czasu. Faza definiowania zdarzeń bez ich umieszczania na osi czasu nazywa się chaotyczną eksploracją. Oto początkowo zdefioniowane zdarzenia: Po zdefiniowaniu zdarzeń można poukładać je na osi czasu. Zdarzenia mogące występować w tym samym czasie znajdują się na jednym poziomie (w tej samej kolumnie). Podczas porządkowania zdarzeń na osi czasu mogą pojawić się nowe zdarzenia. Możemy dodawać tutaj również nasze problemy/pytania związane z danym zdarzeniem. Możemy dodać złotą taśmę z nazwą procesu dotyczącą kilku zdarzeń. Jeżeli zdarzenie jest zbyt ogólne powinniśmy je rozbić na kilka mniejszych zdarzeń. Po etapie big picture możemy zacząć fazę zwaną "Process level".



Rysunek 2: Porządkowanie linii czasu



Rysunek 3: Process level



Rysunek 4: Cykl TDD

2 TDD - Test-driven development

Klasyczne tworzenie oprogramowania rozpoczynamy od pisania dowolnej funkcjonalności. W naszej aplikacji może to być funkcjonalność stworzenia nowego treningu. Po napisaniu takiej funkcjonalności można ją przetestować ręcznie (manualne testowanie oprogramowania przy użyciu scenariuszy testowych) lub przy użyciu skryptów (testy automatyczne np. testy jednostkowe). Jednak w TDD testy jednostkowe piszemy przed napisaniem funkcjonalności. Takie podejście ma wiele zalet:

- Lepsze zrozumienie jak ma działać dana funkcjonalność: wcześniejsze napisanie testów wymusza przeanalizowanie przyszłej funkcjonalności
- Większa dbałość o architekturę modularność i czytelność kodu: duże metody bardzo trudno się testuje dlatego testy wymuszają pisanie bardziej modularnego i bardziej czytelnego kodu
- Mogą stanowić dokumentację: dobrze nazwane testy określają za co odpowiedzialna jest dana funkcjonalność
- Łatwiejsze utrzymanie i rozwijanie aplikacji: po dodaniu nowej funkcjonalności możemy użyć wcześniejszych testów do sprawdzenia czy nowe fragmenty kodu nie zepsuły działania wszystkich pozostałych funkcjonalności
- Pewność zrealizowania wymagań: dzięki dobrze napisanym testom możemy być pewni, że nasza aplkacja realizuje wszystkie założenia bez wykraczania poza wymagania

Jeżeli znamy zalety tego rozwiązania przejdźmy do zasad:

- przed napisaniem kodu produkcyjnego musisz napisać nieprzechodzący test
- nie można napisać większego testu niż jest to potrzebne tylko po to aby test nie przeszedł
- nie można napisać więcej kodu produkcynego niż jest to potrzebne tylko po to aby test przeszedł

TDD to cykliczny proces, który wygląda następująco: