10 ZASAD EFEKTYWNEGO CODE REVIEW

PRZYGOTOWANE PRZEZ

MATTCODES

Maciej Mortek

www.mattcodes.pl
hi@mattmortek.co

1. USTALCIE WSPÓLNE WYTYCZNE

Zauważmy, że chcąc dołączyć do pracy nad jakimkolwiek projektem open source, musimy zapoznać się z plikiem, który opisuje zasady współpracy, standardy kodowania i tym podobne sprawy.

Gwarantuje to, że jakikolwiek własny dodany kod będzie spójny z zasadami wyznaczonymi przez autorów tego projektu.

Taki dokument, opisujący standardy kodowania, formatowania kodu czy architektury systemu, powinien istnieć i być dostępny w każdym projekcie, nad którym pracuje nasz zespół . Dzięki temu już na początku eliminujemy wszelkie niepotrzebne dyskusje podczas code review, jak chociażby dotyczące liczby spacji we wcięciach nowych linii.

2. KORZYSTAJ Z ODPOWIEDNICH NARZĘDZI

Kiedy już wspólnie z zespołem macie opracowany jeden standard kodowania do wybranego projektu, nie bójcie się korzystać z narzędzi, które pomogą Wam się tych wytycznych trzymać.

Lintery, bo o nich tu mowa, zarówno te wbudowane w IDE, dostępne jako zewnętrzne narzędzia czy też te uruchamiane podczas „pushowania" kodu na serwer, zadbają o to, aby kod był odpowiednio sformatowany, a proces code review nie będzie wymagał toczenia sporów o takie szczegóły jak właśnie wspomniana wcześniej wielkość wcięcia nowych linii.

3. PRACUJ Z CHECKLISTAMI

Kolejnym ułatwieniem, zarówno dla recenzenta, jak i recenzowanego, mogą być checklisty. Możemy posiadać wiele checklist jednocześnie, zarówno ogólnych, dotyczących standardów w całym projekcie, jak i osobistych, z listą najczęściej popełnianych przez nas błędów lub z rzeczami, o których nigdy nie pamiętamy.

Sprawdzenie checklisty zajmie nam nie więcej niż kilka minut, a jako recenzenci będziemy mieć pewność, że o niczym nie zapomnieliśmy. Jednocześnie tych kilka minut w przypadku developera wysyłającego swoje zmiany na serwer, może oznaczać zaoszczędzonych kilkanaście minut na późniejszych poprawkach kodu i ponownym procesie przesyłania zmian do repozytorium.

4. BUDUJ SWOJE PR-Y Z GŁOWĄ

PR = pull request – proces porównania naszych zmian przed wysłaniem ich na główną gałąź projektu

Informacje o tym, co powinien zawierać nasz pull request, a także jak powinien wyglądać, zazwyczaj znajdują się w dokumencie opisującym zasady współpracy w projekcie. Jednak jeśli takiego dokumentu nie mamy albo dopiero chcemy go stworzyć, to polecam zapoznanie się z kilkoma radami na temat tego, jak powinien wyglądać łatwy do sprawdzenia pull request.

1. Zamieść w opisie pull requestu link do taska z Twojego systemu zarządzania taskami. Dzięki temu recenzent nie będzie musiał tracić czasu na samodzielne poszukiwania tego taska w programie.

2. Staraj się nie wysyłać więcej niż 400 linii kodu ze zmianami w jednym pull requeście. Badania dowodzą, że sprawdzenie około 200 do 400 linii kodu przebiega najłatwiej i jest najdokładniejsze. Powyżej tej liczby tracimy koncentrację i istnieje większe ryzyko, że coś zostanie pominięte.

3. Jeśli podczas prac nad naszym taskiem musimy sformatować jakąś większą część kodu, tak aby odpowiadała naszym wytycznym w projekcie, to starajmy się to robić jako osobne PR-y lub przynajmniej osobne commity.

4. Chociaż nasz kod na ogół nie powinien wymagać tłumaczenia, czasem zdarza się, że coś musi zostać zrobione w taki, a nie inny sposób. Nie bójmy się w takim przypadku zostawić dodatkowego komentarza dla recenzenta, aby ułatwić mu zrozumienie naszego kodu.

5. ZAPLANUJ SWÓJ CZAS

Jeden programista nie powinien poświęcać na code review więcej niż 60 minut dziennie. Im dłużej będziemy pracować w ciągu dnia wyłącznie nad code review, tym bardziej będziemy tracić koncentrację i tym samym zwiększy się niebezpieczeństwo, że pominiemy istotne szczegóły.

Jeśli na przykład jesteśmy team leaderem i odpowiadamy za planowanie code review w naszym zespole, nie ustalajmy reguł, że jeden programista wykonuje code review przez cały tydzień, a potem zastępuje go kolejny. Zamiast tego dynamicznie, w zależności od liczby pull requestów do sprawdzenia, przydzielajmy do nich konkretne (różne) osoby z projektu.

6. MĄDRZE DOBIERAJ RECENZENTÓW

Ta zasada odnosi się w głównej mierze do osoby odpowiedzialnej za przydzielanie osób do code review.

Jeśli dany pull request dotyczy konkretnego fragmentu aplikacji, nad którym pracował już inny programista, to zapewne on będzie najlepszą osobą do sprawdzenia nowych funkcjonalności w tej części. Jednakże nie ograniczajmy się tylko do jednego recenzenta. Przypiszmy do tego pull requestu również drugą osobę, najlepiej taką, która jest albo nowa w projekcie, albo nie miała jeszcze do czynienia z tym fragmentem aplikacji. W ten sposób poszerzamy wiedzę o projekcie w całym zespole i w przyszłości łatwiej będzie nam wymieniać developerów pomiędzy taskami. Dodatkowo świeże spojrzenie na dany fragment aplikacji również może okazać się bezcenne.

7. BUDUJ, ZAMIAST NISZCZYĆ

Pamiętajmy, że code review pomaga w osiągnięciu wspólnego celu, jakim jest zbudowanie aplikacji łatwej do rozwoju i pozbawionej błędów. To nie jest miejsce na kłótnie, niepotrzebne spory i dyskusje. Zamiast tego bądźmy otwarci na nowe idee, a komentarze wykorzystujmy do burzy mózgów, a nie do kłótni.

Proces code review powinien scalać zespół, rozwijać młodszych stażem programistów, wprowadzać w projekt nowych programistów, dołączających do naszego zespołu.

8. NAUCZ SIĘ UDZIELAĆ FEEDBACKU

Najważniejszą osobą w procesie code review jest recenzent. Od tego, jaki będzie dawał feedback, zależą nastroje w zespole i to, czy code review będzie nam przynosiło więcej korzyści niż strat. Z powyższych względów warto, aby każdy recenzent stosował się do poniższych reguł:

1. Unikaj zgadywanek – wskazuj dokładnie błędy i możliwości ich rozwiązania.
2. Nie uogólniaj – każdy komentarz powinien odnosić się do konkretnego fragmentu kodu, a nie do programisty, jego sposobu programowania czy przyjętego sposobu rozwiązywania zadania.
3. Nie szufladkuj developerów – to może być najgorsze, co możesz zrobić. Nie odrzucajmy czyichś zmian tylko dlatego, że jest to osoba młodsza stażem lub jest programistą z zewnątrz (np. z zewnętrznej firmy).
4. Zaplanuj spokojny czas na CR – jeśli jesteś po ciężkim dniu, po trudnym releasie na produkcji albo męczącej dyskusji z klientem – proszę, nie siadaj do code review.

9. BĄDŹ OTWARTY NA FEEDBACK

Jednakże każdy kij ma dwa końce. Tym samym jako osoba recenzowana nie powinniśmy być zamknięci na feedback, który dostajemy.

Warto być otwartym na sugestie i propozycje, ponieważ każdy z nas ciągle się uczy i nawet jeśli jesteśmy senior developerami, nie oznacza to, że nie popełniamy błędów lub że coś nie może zostać zrobione lepiej.

A w kwestiach spornych, zamiast tracić czas na zbędne dyskusje w komentarzach, zawsze możemy się zwrócić do osób przełożonych w projekcie, aby pomogły nam rozwiązać dany problem raz a skutecznie.

10. PAMIĘTAJ, ŻE MACIE WSPÓLNY CEL

Na koniec rada dla wszystkich programistów biorących udział w procesie code review. Pamiętajmy, że każda osoba w projekcie dąży do tego samego celu, więc jeśli ktoś ma inne zdanie od nas, to nie wynika to z niechęci do nas, a po prostu z troski o projekt i jego dalszy rozwój.

Każdy z nas chce jak najlepiej dla projektu i chce, aby spełniał on wszystkie postawione przed nim zadania. Dlatego nie obrażajmy się, bądźmy otwarci na zdanie innych i pamiętajmy, że code review to nie tylko recenzja kodu. Jest to także nauka programowania, poznawanie innych punktów widzenia, proces wdrażania się do projektu i wiele więcej.