Kvalitetssäkring

Skapad av: Mikael Johansson

Version: 1.0

Senast ändrad: 2022-05-02

Index

Index **2**

Bakgrund **3**
- Det här dokumentet 3
- Kvalitet 3
- Sträva efter hög kvalitet 3

Definition of Done **4**
- Introduktion 4
- Före pull request 4
  - Kodgranskning 4
  - Enhetstester 4
  - Integrationstester 4
  - Manuell testning 4
  - Sidoeffekter 4
  - Regressionstester 4
- Efter pull request 5
  - Kodgranskning – av kollega 5
  - Testning av ansvarig kvalitetssäkrare 5
  - Godkänd av PO 5

Kvalitetssäkring **6**
- Introduktion 6
- Säkra själv 6
  - Fungerar allt felfritt? 6
  - Är beteendet logiskt? 6
  - Fungerar lösningen som den ska? Uppfyller vi kundens krav på funktionen? 6
  - Fungerar allt gammalt fortfarande? 6

Bakgrund

Det här dokumentet

Det här är ett dokument om hur vi på devies kan jobba för att säkra kvaliteten i mjukvaran, produkterna och tjänsterna som vi levererar till kund. Det ska också kunna ge riktlinjer för hur vi jobbar internt, både för produktutveckling och för individuell utveckling.

Kvalitet

Kvalitet är något som påverkar det upplevda värdet. När det gäller mjukvara så handlar det primärt om pålitlighet, prestanda, säkerhet, upplevelse och kanske framför allt om syftet med systemet uppfylls.

Sträva efter hög kvalitet

Anledningarna till att sträva efter hög kvalitet är många. Hög kvalitet ger nöjda kunder vilket i sin tur ger återkommande kunder. Nöjda kunder pratar också gott om företaget och sprider ordet, vilket kan leda till nya kunder.

Hög kvalitet ger förhoppningsvis också nöjda utvecklare som trivs på jobbet och känner stolthet över värdet de levererar. Det ger också utvecklare som pratar gott om sina uppdrag och sin arbetsplats.

En hög kvalitet på leveransen ger också nöjda projektledare och produktägare som oftast har ett slutansvar mot intressenter hos kund. Det är aldrig roligt att behöva stå till svars för system som inte fungerar som tänkte eller går sönder.

Definition of Done

Introduktion

Tanken med en ”definition of done” är att, på en organisatorisk nivå, ha en sorts lägstanivå för vad som kan anses vara färdigt. Genom att höja vår lägstanivå på koden vi levererar kommer vi att förbättra kvaliteten på slutprodukten.

Denna ”definition of done” utgår från två tillstånd; före en ”pull request” och efter en ”pull request”. Före man blandar in en kollega ska man ha betat av en checklista med aktiviteter. Därefter ska en kollega beta av en annan checklista innan något kan anses vara klart. En pull request i det här kontextet behöver inte nödvändigtvis vara en faktiskt pull request, som man har i git, utan det är när man påstår att en utvecklingsaktivitet är klar.

Före pull request

Kodgranskning

Man ska alltid kodgranska sig själv innan man skickar in sin kod och man bör göra det på samma sätt som om man skulle kodgranskat en kollega. Genom att dubbelkolla sitt eget arbete kan man förhindra att det slinker igenom saker av lägre kvalitet, och ju tidigare man upptäcker dessa, desto bättre.

Enhetstester

Funktionaliteten och logiken som vi utvecklar ska ha automatiska enhetstester. Koden som vi skriver ska brytas ner i små funktioner och testas där det är relevant. Det kommer inte att finnas några krav på 100% testtäckning, men det ska strävas efter att all logik testas. Finns det inga automatiska tester kan det inte anses vara klart.

Integrationstester

Likt enhetstester så ska det finnas integrationstester där det är relevant.

Manuell testning

Innan ens arbete kan anses vara färdigt så ska man ha testat resultatet i relevanta webbläsare på relevanta enheter. Om en majoritet av systemets användare använder Internet Explorer 11 så går det inte att anse att man är klar innan man har testat i just denna miljö. Detsamma gäller olika typer av enheter.

Sidoeffekter

När man gör en manuell testning är det bra att säkerställa, i så stor utsträckning som möjligt, att det man gjort inte har fått några sidoeffekter. Är man osäker så flaggar man för detta när man säger att man är klar.

Regressionstester

En viktig del i att undvika sidoeffekter är att man kör regressionstester. Det är inte konstigare än att man ser till att köra alla tester och ser till att förändringarna som gjorts inte ändrat några gröna lampor

till röda.

Efter pull request

Kodgranskning – av kollega

För att främja den individuella utvecklingen, hos både granskare och den granskade, och för att försäkra oss om att vi håller en hög kvalitet i allt vi gör, så ska minst en kollega granska det man har gjort. Vid en kodgranskning ska man bocka av och svara på ett gäng saker;

- Självförklarande kod
  - Är koden självförklarande? Finns det dokumentation där det behövs?
- Förbättringspotential
  - Går det att lösa på ett bättre sätt? Genom att vi utmanar varandra så utvecklar vi varandra. Man bör inte fråga sig om det funkar utan om det går att göra bättre.
- Buggar
  - Kolla av logiken och försök stämma av så att det fungerar som det är tänkt. När man utvecklar får man lätt ett tunnelseende.
- Automatiska tester
  - Finns det automatiska tester? Finns det otestad logik så skicka tillbaka requesten direkt.
  - Om det finns tester, så granska dessa. Tester som kontrollerar att 1 = 1 ger oss inte så mycket, utan kontrollera så att de faktiskt testar det som ska testas.
- SOLID
  - Följer requesten SOLID-principerna? Om inte, förklara vad som kan förbättras och varför.

Tänk på att när man godkänner en pull request, så ska man se det som att man överför en del av ansvaret för koden. All kod som kommer in i ”master” har man ett gemensamt ansvar över och man kan inte komma och *gnälla* på något som man tidigare har godkänt.

Tänk också på att lägga fram eventuell feedback eller invändningar på ett bra sätt. En kodgranskning måste ske med vetskapen om att det inte är någon prestige inblandad, men man bör välja sina ord med omsorg.

Testning av ansvarig kvalitetssäkrare

Vid behov har ett projekt en ansvarig kvalitetssäkrare. Denne ska ha testat slutresultatet före något kan anses vara klart. Vid testning så stäms acceptanskraven av och det utvärderas om dessa uppfylls. Här är det viktigt att inte krångla till det. Finns det ingen möjlighet till testserver och allt vad det innebär, så sätt upp mjukvaran och bjud över personen i några minuter. Finns det ingen ansvarig kvalitetssäkrare för projektet, så ta ansvar och säkra kvaliteten i utvecklingsteamet.

Godkänd av PO

Om projektet har en dedikerad produktägare så är det denna som har fullständig beslutsrätt om vad som kan anses vara klart, då det också är denne som har ansvaret gentemot intressenter. Kommunikation och korta feedbackloopar är nyckeln till framgång.

Kvalitetssäkring

Introduktion

Kvalitetssäkrare och produktägare kommer att gå igenom kundens användningsscenarier och klicka sig igenom alla möjliga sätt att interagera med mjukvaran. Allt som går att göra finns det någon användare som förr eller senare kommer att göra, så det behöver hanteras i mjukvaran.

Alla fel eller konstigheter som du själv upptäckt före kvalitetssäkraren och produktägaren är saker som ni slipper bolla fram och tillbaka mellan er sen.

Säkra själv

Börja med att utföra uppgiften som kunden kommer att utföra. Kan man göra på olika sätt? Testa alla möjliga kombinationer. Detta tar inte många minuter att gå igenom.

Kombinera även ny funktionalitet med gammal där det är möjligt (t.ex. om du har implementerat en redigeringsfunktion, testa då även att det går att redigera något som du precis har skapat upp, något som legat sedan tidigare, något som du redan redigerat en gång, osv.).

Du kan ställa dig följande frågor medan du testar av lösningen:

Fungerar allt felfritt?

Inga krascher eller errors/varningar i konsolen.

Är beteendet logiskt?

Beter allt sig som man kan förvänta sig? Får användaren feedback på vad som händer?

Fungerar lösningen som den ska? Uppfyller vi kundens krav på funktionen?

Uppnår användaren sitt mål? Uppfyller vi eventuella acceptanskrav?

Fungerar allt gammalt fortfarande?

Fungerar det nya ihop med det som redan fanns på plats?

Ju konstigare grejer du testar desto större chans att hitta dolda fel innan någon annan gör det. Då har du som utvecklare möjlighet att rätta till dem innan någon annan vet om att felen någonsin har funnits. I längden leder det till att du blir känd bland kollegorna som en som sällan levererar några fel, någon man kan lita på. Samtidigt har du sparat in på dina kollegors värdefulla tid.