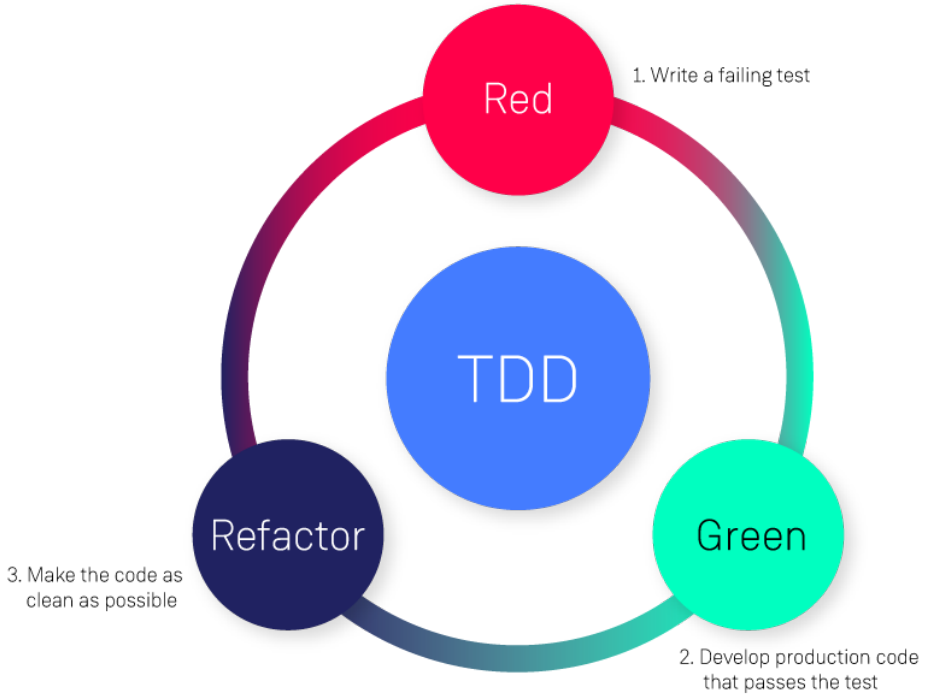

VALIDACIÓ DE PROGRAMES

CFGS Desenvolupament d'aplicacions web
M5:UF3

Neus Bravo Arias

ÍNDEX

ÍNDEX.....2

INTRODUCCIÓ..... 3

PREGUNTA 1: Els meus errors..... 4

Pregunta 2: El cicle de TDD.....5

Pregunta 3: Qui fa què..... 6

Conclusions.....7

INTRODUCCIÓ

El següent document tractarà la metodologia TDD (Test Driven Development), fent una petita reflexió sobre els errors personals a l'hora de programar i després fent énfasis en el cicle d'aquest nou concepte que estem aprenent.
El TDD es conforma per un cicle de cinc passes basat en que les proves que testejaran el codi han de ser la base sobre la que construïm el mateix codi. Això vol dir que abans de començar a desenvolupar les funcionalitats, hem de parar-nos a pensar en com farà el nostre codi per passar les proves que prèviament haurem dissenyat.
En les preguntes 2 i 3 parlarem una mica sobre aquest cicle i profunditzarem en cada fase i també veurem quines persones són les encarregades de dissenyar i elaborar aquestes proves.
Finalment trobem les conclusions finals sobre el TDD.

PREGUNTA 1: Els meus errors

Pensa un error de programació concret que t'hagi passat. Intenta trobar un que et costés prou de trobar i/o de corregir. Descriu en detall en què consistia l'error, com te'n vas adonar, com vas realitzar les proves, en que va consistir la correcció i intenta explicar com et va fer sentir cadascuna de les fases.

L'error que més recordo va ser fa poc en un exercici de vins. El principal problema era que havia de fer una nova implementació de tantes a l'exercici i havia d'incloure molts mètodes nous que fessin funcionar aquesta nova funcionalitat.

El primer error que vaig tindre va ser fer-ho absolutament tot de cop sense molestar-me en testejar ni un print. Això va fer que em trobés amb un tros de codi gegant el qual tenia errors per totes part i no sabia jo per on començar a arreglar-ho. Mentre ho feia em sentia bastant segura pensant que tot funcionaria com ho ideava al meu cap, a la que em va petar la compilació la primera vegada vaig ser conscient de que havia d'haver anat pas per pas.

Va ser una mica complicar de trovar l'error, ja que jo pensava que era un problema que venia des del codi de l'exercici anterior, el qual estava reutilizant. Vaig perdre molt de temps rellegint el codi anterior i buscant l'error a base de mirar per sobre fins que vaig començar a fer proves a aquell codi. Després d'un par de dies vaig concloure en que no era problema de l'anterior perquè tot funcionava com havia de funcionar i vaig tornar al codi actual.

El que més em va confondre va ser que l'error m'apareixia a una línia bastant anterior a la que realment estava el problema. I jo crec que això va ser el que em va tindre tant de temps mirant per on no era. Vaig començar a fer proves més a prop d'on s'originava el problema i sentia molta satisfacció cada vegada que trobava un error, perquè em feia sentir que entenia més el funcionament del que estava fent i això donava peu a que, per cada error que solucionaba, se m'ocurrien més errors que podien estar causant el problema.

Finalment i a base de proves, vaig arribar al mètode on es trobava l' error, que va resultar ser una petit condició.

Òbviament després de tants dies buscant, em vaig sentir bastant estúpida com cada vegada que l'error és una tonteria que es pot arreglar en 5 minuts, però sempre passa així que ho considero part del procés ja.

Però que satisfacció quan es passen els exercicis!!!

Pregunta 2: El cicle de TDD

Descriu amb les teves paraules les cinc passes que conformen el cicle de Test Driven Development. Compara aquest cicle amb la teva manera de programar actual i digues quina et sembla meś adequada segons criteris com velocitat de desenvolupament, qualitat de codi generat, mantenibilitat, etc.

El primer pas del Test Driven Development és pensar en el casos d'ús que tindrà l'aplicació i centrar-nos en crear una funció, un mètode o un tros de codi que s'escrigui pensant en com passarà les proves de la prova que tenim prèviament preparada.

A mesura que tenim el codi preparat per passar la prova, l'executem i esperem que falli on estava pensant que fallaria. Si no ho fa, a lo millor el test no està lo suficientment ben preparat.

Altre pas seria tenir en compte que el codi que estem desenvolupant ha de ser el més simple possible i centrar-se només en com passarà les proves. Les funcionalitats extres o més complexes les deixarem per més endavant.

En aquest metodología utilitzem un patró que consisteix en vermell-verd-refactorització. Això vol dir que al principi tots els test haurien de trobar errors, el qual significa que el test ha estat ben desenvolupat i ha complit el seu objectiu. Després d'això, tornem a executar i hauria de passar tot correcte per després, com a últim pas, refactoritzar el codi necessari per tenir una bona mantenibilitat.

Jo crec que el cicle TDD és bastant òptim per començar codis des de cero i per portar un seguiment de que tot el que crees està funcionant adequadament. Sense dubte, a mi em serviria molt aprendre esta metodologia, ja que la meva és totalment la contraria i es basa en fer tot de cop i començar a esbrinar error per error que està ocorrent.

Pregunta 3: Qui fa què

Per cadascun dels nivells de prova, indica quin és el perfil dels testejadors més adequats per realitzar-les. Justifica-ho.
Proposa qui hauria de dissenyar aquestes proves i quan ho hauria de fer(a quina fase del desenvolupament).

Segons l'abast de les proves trobem categories com: proves de sistema, d'integració, funcional o unitàries.

Les proves de sistema són adequades per ser testejades pels testers o inclós pels usuaris, ja que no es tenen tant en compte els detalls de codificació sinó les funcionalitats finals que demanen els client. Aquestes proves haurien de ser dissenyades de la mateixa manera pels testers a la primera fase del TDD, abans d'escriure el codi.

Les proves d'integració posen a prova el funcionament de diverses parts del sistema conjuntament, per tant haurien de testejar-les els testers en la segona fase del desenvolupament, o sigui quan estem provant totes les proves esperant trobar errors en les últimes proves creades. Haurien de ser ells perquè no han sigut els creadors del codi i aniran directament a provar el funcionament esperat de certa part del codi.

Les proves funcionals les haurien de provar els tester o també els desenvolupadors, ja que són proves de una part de codi molt concreta i de la qual s'espera una funcionalitat simple i específica i no hauria cap problema en que fossin els mateixos desenvolupadors els que la provessin.

Per últim, les proves unitàries haurien de testejar-les el mateixos desenvolupadors durant l'última fase del TDD, la fase de refactorització, ja que és una fase que tracta de minimitzar dependències d'una peça de codi amb la resta i de millorar estructures i els developers són els que millor coneixen el codi que s'està tractant.

Conclusions

Sense dubte, la metodología TDD aporta una manera de crear codi molt òptima quan es tracta de començar des de cero, i també quan utilitzem altres biblioteques. El fet de centrar-se en escriure les proves abans que el codi és molt útil per comprovar que està funcionant correctament, per reduir errors i per millorar la qualitat del programa.
Sembla una mica difícil al principi el fet d'haver de plantejar proves per un codi que encara no existeix i que a lo millor no sabem com funcionarà del tot encara ni com es relacionarà amb les altres classes, però, a pesar d'això, crec que val la pena poder portar un control de tot el que estàs fent en tot moment, a més que dona més facilitat per implementar noves funcionalitat o fer canvis en en el codi.