TREBALL FINAL DE CARRERA

AUTOR : DIEGO COMAS TORRES
CONSULTOR: ORIOL MARTÍ GIRONA

DESENVOLUPAMENT D'APLICACIÓ WEB AMB INTEGRACIÓ CONTÍNUA.

DESCRIPCIÓ DEL PROJECTE

El projecte combina dos elements a analitzar mitjançant la creació d'una aplicació web :

1) Crear una aplicació d'enquestes on els usuaris podran publicar preguntes, les respostes i observar els vots que obtenen aquestes en temps real.

2) Desenvolupar aquesta aplicació utilitzant mètodes d'integració contínua en comptes de mètodes clàssics de desenvolupament de programari, al finalitzar considerar els beneficis i inconvenients.

OBJECTIUS

Al finalitzar el projecte s'espera obtenir:

1) Millor coneixement de la metodologia àgil aplicada.
2) L'aplicació web que satisfaci els requisits dels usuaris.
3) Un procés definit i automatitzat per analitzar si funciona la integració contínua en projectes web.
4) Accés a l'aplicació web a l'adreça comas.me
5) Accés als informes d'integració a jenkins.comas.me

METODOLOGIES ÀGILS

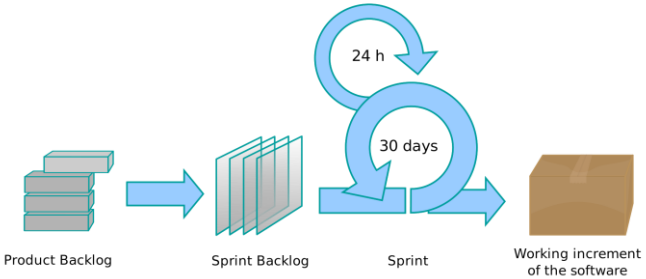
Les metodologies àgils són una sèrie de tècniques per a la gestió de projectes que han sorgit en contrapartida als mètodes clàssic.

Els principals valors són:

- Individus i interaccions per sobre de processos i eines.
- Programari que funcioni per sobre de documentació exhaustiva.
- Col·laboració amb el client per sobre negociació de contractes.
- Respondre als canvis per sobre de seguir un pla estricte.

Els principals mètodes són:

- Scrum
- Extreme Programming (XP)
- Kanban



Esquema de mètode SCRUM

INTEGRACIÓ CONTÍNUA

La integració contínua no és una eina sinó una pràctica que sorgeix de l'Extreme Programming (XP).

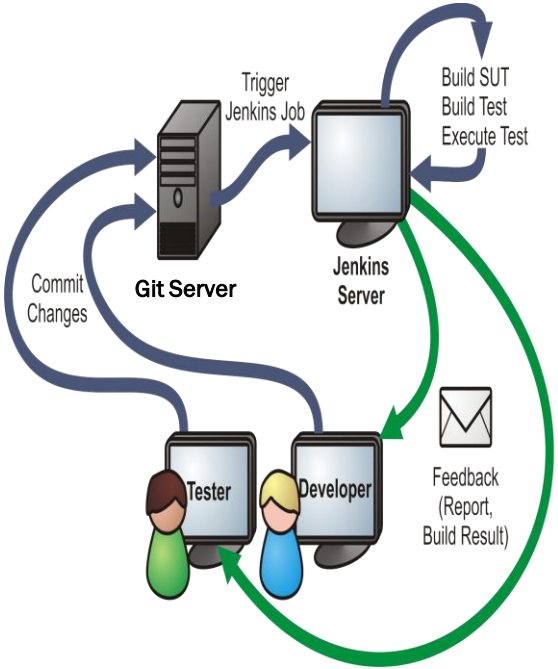
Segons Martin Fowler la integració contínua és una pràctica de desenvolupament de programari on els membres de l'equip integren el seu treball freqüentment, al menys un cop al dia. Cada integració es verifica amb un build automàtic (incloent l'execució de proves) per detectar errors tan ràpid com sigui possible.

Aplicar un procés d'integració contínua aporta sistematització a tot el cicle de vida del producte, evitant sorpreses d'última hora al mantenir sota control tot el referent a les proves, verificació i construcció del mateix.

INTEGRACIÓ CONTÍNUA PROCÉS

Aquestes són les etapes comuns de la integració contínua:

1) Un membre de l'equip crea nou codi o noves funcionalitats a l'aplicació i s'envien els canvis al servidor de versions de codi.
2) El servidor de versions notifica al servidor d'integració (en aquest cas Jenkins) i envia els canvis del codi de l'aplicació.
3) El servidor Jenkins compila o crea l'aplicació i executa totes les proves automàticament.
4) Si el nou codi supera les proves, el tester rep una notificació per si vol fer comprovacions addicionals de forma manual o afegir noves proves. Si no es superen les proves el desenvolupador rep una notificació amb les errades detectades i es torna a iniciar el cicle.



ENTORN DE TREBALL

Aquest és el programari utilitzat per desenvolupar l'aplicació:

Ubuntu 14.04 : El sistema operatiu del servidor de l'aplicació i el servidor Jenkins.

Sublime Text : Editor de codi font per crear tots els arxius HTML, CSS i JavaScript.

MongoDB : Sistema de base de dades NoSQL on es guardaran les dades de l'aplicació.

Node JS : Entorn de programació dissenyat per escriure aplicacions d'Internet escalables.

Express JS : Framework que simplifica les crides que es fan a Node JS.

Angular JS : Framework per desenvolupar aplicacions web d'una sola pàgina.

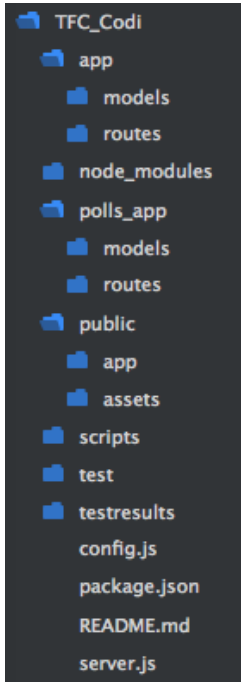
Jenkins-CI : Servidor d'integració contínua pel desenvolupament de programari.

Protractor test : Framework per fer proves "end to end" en aplicacions Angular JS.

Selenium webdriver : Eina per automatitzar les proves en navegadors web.

Google Chrome : Navegador web on es realitzen les proves amb Selenium.

ESTRUCTURA DE L'APLICACIÓ WEB



App: Aquí hi ha els models i les rutes, els models especifiquen com és l'esquema de l'objecte Usuari a la base de dades. Les rutes especifiquen com es gestionen les crides http GET i POST.

Node_modules: Aquí es guarden totes les dependències Node JS de l'aplicació.

Polls_app: Els models especifiquen l'esquema de l'objecte Enquesta i les rutes les crides http.

Public: Aquí es troben tots els arxius estàtics(assests) com CSS i els de l'aplicació Angular JS.

Script: Petits programes per automatitzar processos com reiniciar el servidor.

Test: Aquí es troben tots els arxius per fer les proves, que executarà Protractor Test.

Testresults: Aquí es guarda el informe després de cada prova en format .xml

Config.js: Aquest arxiu especifica detalls com el port per publicar la web o el port d'accés a la base de dades.

Package.json: Aquest arxiu inclou el llistat de dependències que s'han d'instal·lar.

Server.js: Aquest arxiu és el que utilitzarà Node JS per iniciar l'aplicació, és l'arxiu principal, s'indiquen coses imprescindibles com especificar els paquets necessaris per executar l'aplicació.

PRODUCT BACKLOG

Aquestes són les fites que volem assolir per satisfer els requeriments dels usuaris respecte a l'aplicació web.

Per cada fita com a mínim hi haurà que integrar un cop al dia el codi de l'aplicació per detectar errors el abans possible.

Fita	Tasca	Estat	Estimació hores	Dia 1	Dia 2	Dia 3	Dia 4	Dia 5	Revisió
Com a / visitant de la web vull veure una pàgina de inici per saber on estic accedint.	1	Obert	3	2	0	0	0	0	1
Com a / visitant de la web vull tenir la possibilitat d'enregistrar-me.	2	Pendent	13	6	6	0	0	0	1
Com a / usuari enregistrat vull poder crear una enquesta	3	Pendent	9	0	0	7	1	0	1
Com a / usuari de la web vull poder votar una enquesta.	4	Pendent	3.5	0	0	0	2	1	0.5
Com a / visitant de la web vull poder veure els resultats d'una enquesta.	5	Pendent	2.5	0	0	0	0	2	0.5

PROCÉS CREANT L'APLICACIÓ WEB

1) Per cada iteració farem les proves primer (Test-Driven Development).
2) Un cop hem comprovat que les proves fallen es comença a crear nou codi.
3) Un cop finalitzat el codi s'executen les proves en local.
4) Si es superen les proves en local es fa un "commit" per guardar l'arxiu creat al gestor de versió de codi I començar el cicle de la integració contínua.

1

```
// spec.js
describe('Comprovar titol Enquestes UOC', function() {
    var message = element(by.css('.form-group'));

    beforeEach(function() {
        browser.get('http://localhost:5000');
    });

      it('should have a title', function() {
        expect(browser.getTitle()).toEqual('Hola UOC!');
      });

      it('should show a welcome message', function(){
        expect(message.getText()).toEqual('Benvinguts a la pagina enquestes UOC');
      });
});
```

2

```
Finished in 4.395 seconds
2 tests, 2 assertions, 2 failures

[launcher] 0 instance(s) of WebDriver still running
[launcher] chrome #1 failed 2 test(s)
```

3

```
<!DOCTYPE html>
<html ng-app>
<head>
    <meta charset="UTF-8">
    <title>Hola UOC!</title>
```

4

```
5 specs, 0 failures
Finished in 7.242 seconds
```

SERVIDOR D'INTEGRACIÓ CONTÍNUA

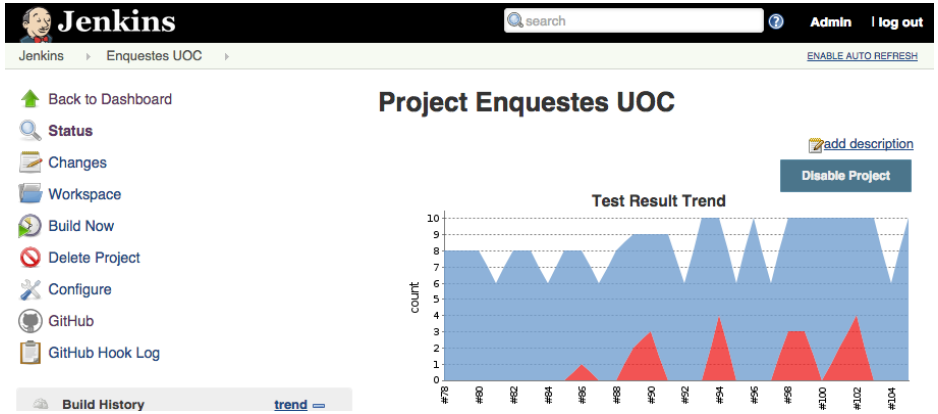
Per fer el procés automatitzat es configura un servidor Jenkins.

1) Aquest rebrà els canvis efectuats al repositori de codi remot mitjançant un "webhook".

2) Executarà les accions segons les especificacions a la configuració.

3) Si tot és correcte, enviarà els canvis al servidor de producció.

4) Crearà un informe de tot el procés.




```
[32minfo[39m:    Forever processing file: [90mserver.js[39m
+ protractor test/conf.js
Using the selenium server at http://localhost:4444/wd/hub
[launcher] Running 1 instances of WebDriver
Started
[32m.[0m[32m.[0m[32m.[0m[32m.[0m[32m.[0m[32m.[0m[32m.[0m[32m.[0m[32m.[0m[32m.[0m


10 specs, 0 failures
Finished in 53.254 seconds
[launcher] 0 instance(s) of WebDriver still running
[launcher] chrome #1 passed
+ ./scripts/deployLiveWeb
Deploying app to live server
```

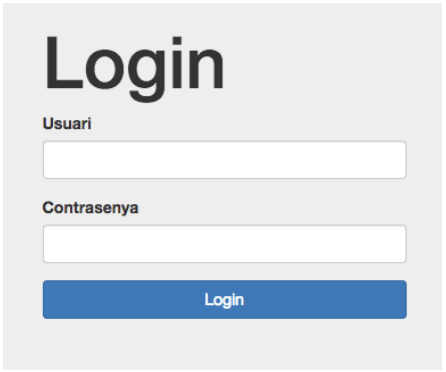
PRODUCTE OBTINGUT

Un cop fetes totes les iteracions hem creat l'aplicació satisfent els requeriments del backlog.

URL : comas.me:8080

Fita 1 – Pàgina benvinguda

Fita 2 – Enregistrar-se



Fita 3 – Crear enquestes

Fita 4 – Votar enquestes

Fita 5– Veure vots enquestes

CONCLUSIONS

Al finalitzar del projecte podem concloure el següent:

1) Amb la metodologia àgil que hem practicat, el disseny s'enfoca únicament a les necessitats de l'usuari, això és beneficiós perquè evita la pèrdua de temps en altres extensions innecessàries per l'usuari.

2) El codi creat és de més qualitat gràcies a la simplicitat en el disseny de l'aplicació.

3) Hi han menys errors a resoldre en etapes finals del projecte gràcies a que el servidor d'integració ha executat totes les proves de les fases prèvies.

4) Com a inconvenient podem dir que a vegades la creació de les proves abans del codi de l'aplicació pot ser difícil, en conseqüència es destina molt temps a fer suposicions de com interactuarà l'usuari amb l'aplicació web.

5) També com a inconvenient podem dir que la configuració de l'entorn ha de ser molt robusta ja que petits errors entre els servidors, les dependències per fer proves remotes etc. han fet fallar moltes vegades el procés automatitzat.