Utilitat d'Integració i SmokeTesting per l'equip de 3D FusionJet d'HP

Iván López

Resum– El projecte *Utilitat d'integració i SmokeTesting per l'equip de 3D FusionJet d'HP* presenta el desenvolupament d'una aplicació web per ajudar a millorar tots els procediments de les proves de tipus *Smoke Test* dels departaments de testing de l'empresa Hewlett Packard. Funciona com a planificador, eina de reporting, generador d'informes i de visualització de resultats. Aquest projecte ha estat pensat amb caràcter específic per a l'empresa cobrint necessitats relacionades amb les metodologies de desenvolupament que es fan servir. Pretén substituir les eines antigues que endarrereixen els processos de testing, millorant la productivitat del personal relacionat amb el cicle de vida dels tests, facilitant la planificació, millorant la qualitat dels tests executats i oferint informació detallada de gran interès per a l'empresa.

Paraules clau– Aplicació Web, API REST, Javascript, Planificador, Productivitat, Qualitat del Software, Smoke Test.

Abstract– The project *Utilitat d'integració i SmokeTesting per l'equip de 3D FusionJet from HP* illustrates the development of a web application which focuses on enhancing the procedures of the *Smoke Tests* and Quality Assurance tests from the test departments of the company Hewlett Packard. It works as a test planner, a reporting tool with the ability to generate reports and as a reviewer of the results. This project has been developed specifically for the company, covering the necessities related to the specific methodologies followed by their development guidelines. This application aims to replace the legacy tools used that delay the test processes, increasing the productivity of the personnel related to the test lifecycle, with a better scheduler, improving the test quality and offering detailed information of great interest for the company.

Keywords– Web Application, API REST, Javascript, Scheduler, Productivity, Software Quality, Smoke Test.

1 Introducció

Gairebé totes les empreses fan servir eines antigues per tasques simples que compliquen la vida als treballadors. Aquestes, però, tenen un caràcter burocràtic i no solen estar a la cua de prioritats de les empreses per modificar-les.

Aquest projecte neix d'una necessitat real al departament de Testing i Qualitat del Software de Hewlett Packard per als projectes d'R+D d'impressió 3D. Actualment no es fan servir les eines adequades per planificar, reportar i extreure dades dels diferents tests que es fan diàriament. Entre els tests realitzats, destaquen els *smoke tests*. Tests que s'executen per cada versió de codi generada i proven les parts més crítiques de cada nova versió del codi del producte.

El projecte *Utilitat d'Integració i SmokeTesting* és una aplicació web que segueix l'arquitectura *Single Page Application* [1] (SPA) composta per un client web i diferents API's. Aquesta aplicació té com a objectiu millorar la planificació, seguiment, reporting i l'anàlisi dels resultats de les proves de tipus *smoke test* [2] i de qualitat del software. Aquesta utilitat pretén millorar la productivitat dels testers amb eines àgils i especialitzades. Inclou noves funcionalitats a més de millorar les actuals. L'eina és interessant per tot el conjunt de persones involucrades en el cicle de test del producte, com els desenvolupadors que treballin en una àrea o subsistema concret, oferint els resultats amb més rapidesa i amb una millor presentació d'aquests. També per mànagers i integradors, oferint informació relativa a l'evolució del projecte, com per exemple, quins tests fallen més, el perquè i a on invertir més en desenvolupament, fent vi-

• E-mail de contacte: ivan.lopezm@e-campus.uab.cat
• Mención realitzada: Enginyeria del Software
• Treball tutoritzat per: Xavier Otazu Porter (Departament de Ciències de la Computació)
• Curs 2019/2020

sibles les vulnerabilitats i errades de codi entenent millor l'origen i poder planificar la correcció d'errors.

2 ESTAT DE L'ART

Actualment, HP ofereix una eina de reporting per *smoke testing* obsoleta i poc flexible. Es va substituir per una metodologia de reporting via Outlook amb taules de dades. Aquesta solució s'ha tornat lenta a mesura que el volum de dades ha anat creixent, ha sigut difícil fer-la servir per reportar i veure resultats. Ja que les dades són a Outlook no existeix una base de dades amb l'històric dels resultats i no es pot fer seguiment amb estadístiques, motiu pel qual es perd molta informació valuosa entre milers de correus.

Aquestes eines faciliten errors humans i no tenen cap integració entre elles. Això provoca que realitzar les tasques diàries de testing tingui un cost temporal molt elevat. Fent un càlcul s'inverteixen entre 2 i 2 h 45 min cada dia a l'hora de planificar i reportar els tests. A més d'ineficients, no existeix una integració entre la planificació inicial i el report final, fent poc intuïtiu el seguiment del test, provant la necessitat de buscar o desenvolupar una nova eina especialitzada per als projectes, revisió d'errors, etc.

No hi ha gaires eines dedicades exclusivament a tests de tipus *smoke test*, ja que és una metodologia que ha d'adaptar l'empresa a les seves necessitats. Algunes eines que es podrien fer servir són Trello, Jira, representats a la figura 1, i aplicacions semblants amb gestors de taulells, on sigui possible crear i assignar tasques. Aquestes eines no s'adapten de manera natural a les metodologies de treball d'HP. Cap d'elles permet reportar de manera eficient les tasques i subtasques de manera detallada. Sumat a què als projectes R+D la privacitat és crítica i necessita molta personalització, he decidit resoldre aquest problema amb una nova aplicació.

Fig. 1: Possibles alternatives a l'eina

En ser una utilitat completament especialitzada per als bancs de proves i metodologies d'HP, es pot obtenir un alt grau d'automatització en tasques repetitives, reduint el temps de recopilació d'informació i generació d'informes, oferint més temps per a les tasques realment importants, en aquest cas, realitzar tests de qualitat.

3 OBJECTIUS

Els objectius del projecte es divideixen en objectius prioritaris per satisfer als stakeholders i en objectius secundaris, funcionalitats que són d'utilitat però que en cas de faltar, no afecten el grau de satisfacció.

3.1 Objectius principals

Alguns dels objectius més importants per satisfer les necessitats del projecte són:

- Desenvolupar una solució web que reculli informació i automatitzi processos sobre tests de tipus *smoke test*.
- Desenvolupament d'un *Scheduler* per planificar tests.
- Desenvolupament d'un *Reporter* per documentar els tests executats.
- Desenvolupament d'un *Reviewer* per revisar els resultats del testing.
- Reduir la necessitat de duplicar informació per reportar als diferents sistemes amb eines específiques.
- Reduir com a mínim 4 vegades el temps dedicat a reportar i planificar.
- Eina modular i adaptable a altres equips de l'organització.
- Millorar la comunicació entre enginyers i testers integrant totes les fases del test en una mateixa eina.

3.2 Objectius secundaris

Alguns dels objectius secundaris que podrien aportar al projecte són:

- Consultar resultats amb una App mòbil.
- Sistema d'accés integrat amb el sistema de domini LDAP corporatiu.
- Automatitzar la llista de revisions de *firmware* per planificar nous tests.

4 BENEFICIS

Els avantatges principals que proporcionarà la nova eina en respecte al sistema anterior són els següents:

- Espai dedicat al testing separat d'altres eines.
- Eina fàcil de fer servir per noves incorporacions a l'equip de testing.
- Estalvi de temps a l'hora de planificar i reportar.
- Generació automàtica d'informes dels tests.
- Extracció automàtica de logs sobre la prova realitzada.
- Minimització dels errors humans.
- Fàcil accés als resultats de tots els tests realitzats.
- Visualització dinàmica de dades d'interès per directius i enginyers.

5 MÒDULS DE L'APLICACIÓ

L'aplicació estarà composta per tres mòduls diferenciats per la fase de testing a la qual pertanyen.

5.1 Scheduler

El mòdul *scheduler* permetrà a l'enginyer de test programar una planificació detallada dels tests a executar cada dia. Ofereix una interfície senzilla i àgil. També estarà disponible per als desenvolupadors que necessitin planificar alguna prova ràpida d'alguna versió de codi. Els tests planificats es mostren a la taula de tests disponibles, llestos per ser reportats.

La pantalla inicial, vista a la figura 2, mostra una taula amb tots els tests planificats en ordre de més nou a antic. Un botó d'informació, per cada test de la taula que aporta més detalls de cada test i un altre per eliminar tests. A la dreta de la taula, a la icona d'e-mail, hi ha disponible l'opció d'enviar l'informe per correu electrònic, generant un esborrany en format .eml, format que utilitza Outlook en els seus e-mails, amb tota la llista de tests planificats i els contactes interessats.

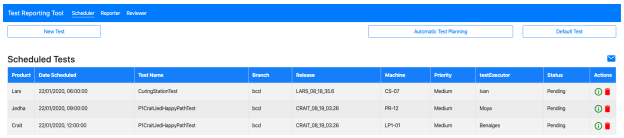

Fig. 2: Pantalla inicial del Scheduler en la que es mostren les dades dels tests i les accions disponibles.

L'opció Default Test mostra un formulari per crear una plantilla de test amb fluxos de treball específics. Permet estalviar feina definint els bancs de proves habituals a l'hora de programar tests.

El botó New Test mostra un formulari per la planificació d'un nou test. Les plantilles definides a Default Test estan disponibles i es poden modificar amb un sistema de drag-and-drop.

Finalment, l'opció Automatic Planning mostra tots els tests diferents per cada producte i permet fer una planificació immediata d'un test. Aquesta opció és útil per tests que es fan cada dia. Amb un clic es pot planificar per al següent dia un test amb tots els fluxos de treball que va tenir aquell test, els detalls es poden editar a la taula de tests disponibles fent doble clic sobre l'element a editar.

5.2 Reporter

El mòdul *reporter* està pensat per als testers. Dóna accés als tests planificats i poden reportar els resultats en una interfície àgil. També permet generar informes dels tests que seleccionin mitjançant checkboxes, estalviant redactar de forma manual e-mails.

A la pantalla inicial es poden veure tots els tests planificats en ordre temporal, representant tota la feina disponible a fer. A sota hi ha un botó, *show completed tests* que desplega els primers 50 tests realitzats de manera *paginada* [3] per evitar oferir grans volums de dades d'un cop i col·lapsar el sistema. A mesura que es navega per la llista de tests apareixen més tests, sempre amb una paginació de 50 blocs.

Si es fa clic sobre l'acció reportar, s'entrarà a la finestra dedicada per reportar el test seleccionat. Aquesta mostrarà la informació rellevant del test i tots els fluxos de treball a executar amb les seves instruccions, tal com es pot veure a la figura 3. Es pot definir l'estat de cada flux de treball i afegir explicacions en text enriquit HTML. Fent clic sobre el botó Show Printer Info es recopilarà la informació del flux executat a la màquina i oferirà els resultats obtinguts de manera automàtica.

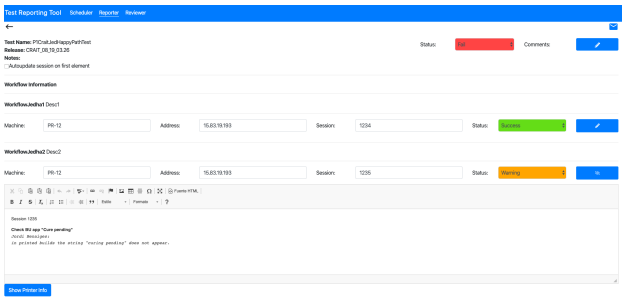

Fig. 3: Vista de report d'un test, en aquesta es veuen els diferents fluxos de treball i els camps a omplir.

5.3 Reviewer

El mòdul *reviewer* permet als desenvolupadors visualitzar els resultats dels tests executats amb explicacions sobre els errors i logs adjunts. Permet filtrar els tests per arribar a la informació desitjada. Mostra mètriques útils per a mànagers i integradors proveint una millor visió global de l'evolució del producte. La informació de cada test apareix en desplegables tal com es veu a la figura 4, en trobar el test a consultar es clica i apareixen els detalls de l'execució. També conté un cercador avançat, on es poden cercar els tests pels criteris més adients per al tipus de projecte.

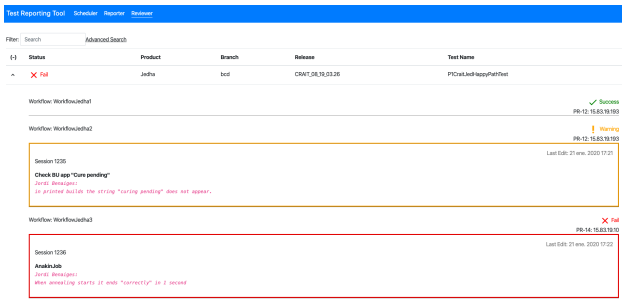

Fig. 4: Vista del reviewer i desplegable d'informació sobre els fluxos de treball executats.

6 PLANIFICACIÓ

La planificació del projecte ha estat en desenvolupament continu durant tota la vida del projecte. Donada la incertesa que poden produir tant els factors interns com externs, s'ha hagut d'anar replanificant a mesura que avançava el projecte, canviaven les necessitats dels stakeholders i es trobaven nous reptes i amenaces. Aquests canvis es van definir al principi del projecte amb una avaluació de riscos i un pla de contingència.

Tot el projecte s'ha organitzat en *milestones*, punts fixats en la planificació per entregar producte. En cada *milestone* s'ha fixat un *producte mínim viable* [4] (MVP) per satisfer un mínim de necessitats a cada entrega.

Per realitzar una correcta planificació, s'ha fet una descomposició jeràrquica de les diferents fases del projecte amb l'eina *Diagrama de Planificació del Treball*, que ha

ajudat a donar una visió global del projecte en un sol full, com es veu a la figura 5, organitzar totes les fases i especificar el treball a executar per complir els objectius i organitzar les entregues necessàries.

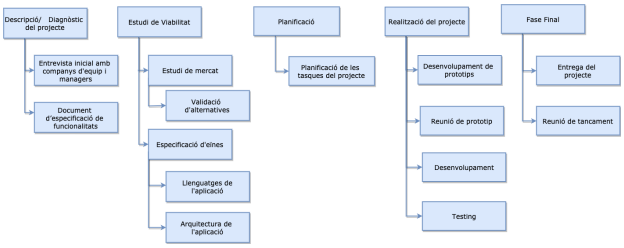

Fig. 5: Diagrama de Planificació del Treball del projecte

El projecte es divideix en tres *milestones* amb les tasques necessàries per arribar a l'objectiu. Les revisions han sigut continuades, facilitant a l'empresa versions del software de manera contínua. A la figura 17 de l'apèndix es pot veure el quadre de planificació global que s'ha consultat durant tot el projecte i el *Diagrama de Gantt* desenvolupat durant les primeres fases i actualitzat amb els canvis de planificació a la figura 19.

6.1 Q1: Entrega fase Alpha del Scheduler

En aquesta etapa es van establir les bases del projecte. Especificacions, eines a utilitzar, disseny i funcionalitats. Es va realitzar una fase d'elicitació per obtenir requisits funcionals, no funcionals i de sistema. Per arribar a aquest objectiu, s'han acordat diferents reunions amb els stakeholders per estudiar les eines actuals, s'ha parlat de les necessitats i s'ha realitzat una navegació sobre l'eina mitjançant un mapa de pantalles, aportant una visió real del producte.

El MVP de la Q1 va ser un planificador de tests, només amb l'opció de planificar nous tests, que es va entregar el més ràpid possible. A mesura que es van desenvolupar noves funcionalitats en cada iteració, com les plantilles de test o exportar a Outlook, es van entregar en forma d'actualitzacions. Durant la primera fase es va assolir:

- Posada a punt de l'entorn de desenvolupament, testing i logging.
- Investigació sobre l'entorn de test i les eines disponibles a Node.js.
- Mapa de pantalles i disseny del *scheduler*.
- Implementació del *scheduler* tant a API com al client.
- Test de l'API, apartat endpoints del *scheduler*.
- Entrega al client un *scheduler* funcional per poder crear tests.

6.2 Q2: Entrega Scheduler i Reporter

En aquesta etapa ha tingut més pes el desenvolupament de funcionalitats, testing i correcció de requeriments de la fase anterior.

El MVP de la Q2 ha sigut entregar l'eina funcional pels enginyers de test i testers sense l'opció textitreviewer. A mesura que s'han desenvolupat noves funcionalitats i s'han solucionat bugs en cada iteració, s'han entregat en forma d'actualitzacions. Durant aquesta fase es va assolir:

- Refactorització de codi de l'API i client web.
- Investigació sobre l'entorn de test i les eines disponibles a Node.js.
- Implementació del *reporter* i començament del *reviewer*.
- Redefinició de requisits segons el feedback dels stakeholders.
- Entrega al client de l'aplicació funcional en mínims.

Durant les proves es va rebre feedback per poder exportar els tests reportats en format de correu Outlook, limitar la inserció de dades en formats erronis, un nou estat *'Running'* per als tests que s'estan executant actualment, permetre la inserció d'imatges per explicar resultats de manera gràfica i optimitzar i automatitzar l'ompliment de camps de formulari repetitius.

Aquesta entrega va servir per trobar errors tant en l'àmbit de codi com d'experiència d'usuari. Un d'aquests errors va ser la inserció de text amb espais al principi o final, provocant errors de consistència i la pèrdua de totes les dades a la base de dades. Trobar errors d'aquest tipus després d'haverhi realitzat *Unit Test* va demostrar la poca importància que s'havia dedicat al test i va provocar un canvi en la planificació, que es pot veure en la figura 18 de l'apèndix, accentuant el temps dedicat a *Unit Test i Code Coverage* de tots els mòduls de l'aplicació.

6.3 Q3: Tancament del Projecte

En aquesta etapa de major duració, s'ha continuat el desenvolupament el mòdul *reviewer*, a la vegada s'han realitzat tests d'integració entre els diferents mòduls i test d'unitat. També s'han preparat els informes finals, documentació i presentació. Aquesta fase ha donat problemes per l'entrega anterior al client. Es va entregar una aplicació amb uns tests insuficients durant la Q2 i va causar un endarreriment en les següents fases, i es va prioritzar fer un banc de tests més rigorosos per a tota l'aplicació. Finalment s'ha fet el tancament del projecte amb l'entrega del codi, setup i documentació.

6.4 Avaluació de Riscos

És molt important per qualsevol projecte poder identificar els riscos potencials i detallar un pla de contingència per cada tasca.

- **No captar requeriments clau:** Fer diferents entrevistes amb el client, pluja d'idees, realitzar enquestes, etc.
- **Seguretat de les dades i privacitat d'HP denegant l'entrega de funcionalitats:** Separar les funcionalitats privades en una API interna que es quedarà als servidors d'HP de les funcionalitats genèriques pel funcionament de l'aplicació.
- **Conflicte de comunicacions HTTP entre servidors a la xarxa R+D d'HP:** Contactar amb el departament de ciberseguretat per intentar gestionar els problemes.

7 METODOLOGIA

En aquesta secció s'explicarà la metodologia àgil escollida per al desenvolupament del projecte, tant com l'estratègia d'elicitació, la gestió de la configuració, el testing i la integració i desplegament continu.

7.1 Metodologia de desenvolupament

Ja que el projecte ha sigut portat a terme per una sola persona, s'ha escollit una metodologia *Kanban* [6], més aplicable per a projectes petits o unipersonals. Permet simplificar la planificació en comparació a altres metodologies col·laboratives.

De manera iterativa s'ha modificat el contingut del product backlog en funció dels nous requisits i canvis durant el desenvolupament.

L'eina per seguir la metodologia i tenir una planificació del projecte serà Trello. Permet tenir una visió global del projecte, un seguiment de les tasques a fer, i l'obligació de tancar les tasques de manera seqüencial, amb un nombre de tasques a la columna *To-do* limitades, evitant així treballar paral·lelament en diferents funcionalitats i focalitzar un únic problema. El tutor per part d'HP del projecte ha sigut col·laborador del taulell, afegint tasques a realitzar per correccions, millores o possibles noves funcionalitats a mesura que es desplegaven noves versions de la utilitat.

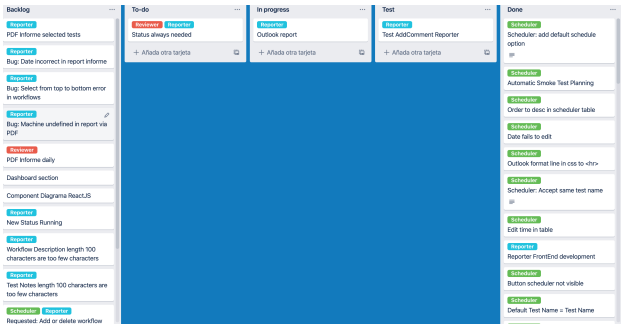

Fig. 6: Estat del taulell Trello després d'entregar una nova versió als usuaris durant la Q3 i rebre feedback sobre errors i millores

7.2 Elicitació de requisits

Per elicitar i analitzar requeriments, s'han seguit diferents passos:

1. Identificació de potencials stakeholders. Entre els stakeholders, trobem mànagers, testers, enginyers de software i integradors.

2. Analitzar els stakeholders per situar-los a una matriu de poder/interès per definir reunions de requisits i enquestes per als més rellevants.

3. Extreure i refinar els requisits obtinguts durant totes les iteracions, mitjançant reunions, validacions de models, prototips, etc.

7.3 Gestió de la Configuració

La gestió de la configuració s'ha controlat des de Github. S'ha escollit aquesta plataforma, ja que és la més coneguda i ofereix accés a repositoris privats per estudiants. S'han configurat tres repositoris, un per al frontend i dos de backend. Es treballa sobre branques per desenvolupar funcionalitats específiques que fan merge a master una vegada funcionen correctament.

7.4 Metodologia de Codificació

S'ha seguit una metodologia de codi autodocumentat, molt freqüent en *APIs* desenvolupades en Node.js. Això no vol dir no documentar el codi, sinó reduir el nombre de comentaris innecessaris. Intenta aconseguir que el codi sigui entenedor per al lector. S'ha tingut especial cura en la bona interpretació del codi, seguint guies i patrons de programació trobats al llibre *Clean Code* [8] amb noms ben definits, funcions concretes i tècniques de refactorització.

7.5 Test d'Unitat, d'Integració i Coverage

En tot projecte de software una de les tasques més importants per assegurar la qualitat és el testing. En aquesta secció es comenten algunes de les tècniques de test aplicades i els errors trobats.

7.5.1 Eines utilitzades

Per desenvolupar el codi de proves de les API's s'ha fet servir *Mocha i Chai* [7], aquests *frameworks* permeten desenvolupar proves d'unitat molt completes per al testing de codi Javascript. Proveeixen mètodes específics per fer proves sobre API's amb una sintaxi fàcil i un sistema *d'asserts* molt complet.

7.5.2 Test d'Unitat i d'Integració

Sobre cada *endpoint* de cada mòdul de l'API (*scheduler/reporter/reviewer)* desenvolupat s'ha realitzat test unitari per verificar que el codi satisfaci els requisits, sigui robust i els resultats sempre siguin els esperats. Els tests unitaris s'han agrupat per mòdul de l'aplicació, per tipus de petició HTTP, alguns d'aquests casos de prova són a la figura 7.

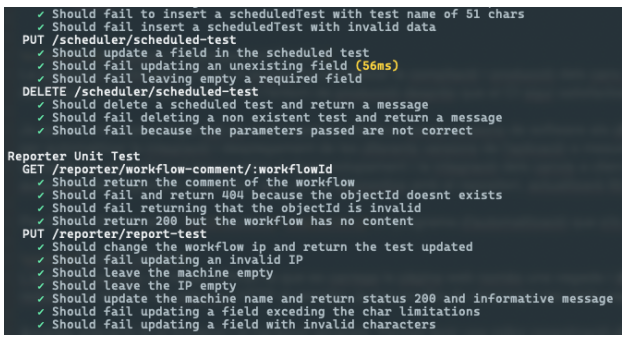

Fig. 7: Fragment d'una part de l'informe dels testos de caixa negra, organitzats per mòduls i per tipus de petició HTTP, cada test mostra la seva funció, estat i temps d'execució

S'ha executat test d'integració interactuant amb diferents endpoints per simular accions complexes i tests E2E, com

per exemple el flux corresponent a crear un producte, afegir dues tasques i planificar un test. Un dels casos de prova es pot veure a la figura 8.

```
it('Should return an array of workflows', async function() {
  let workflowList = [workflow1, workflow2, workflow3];
  let defaultTest = new ScheduledTest.DefaultTest({
    product: 'productName',
    testName: 'defaultTest',
    workflows: workflowList
  });
  await defaultTest.save();
  await chai
    .request(server)
    .get('/api/scheduler/workflow/productName/defaultTest')
    .then(function(res) {
      assert.strictEqual(res.body.length, 1);
      assert.isArray(res.body);
      assert.strictEqual(res.statusCode, 200);
      assert.strictEqual(res.body[0].workflows.length, 3);
    });
});
```

Fig. 8: Cas de test desenvolupat pel mòdul de planificació. Guarda un test a una base de dades mock, crida al endpoint i avalua els resultats obtinguts a la petició de resposta.

7.5.3 Code Coverage

Per oferir una òptima qualitat de codi, també s'ha realitzat test de coverage per assegurar que es passa per totes les línies de codi i localitzar codi redundant o no utilitzat, arribant a un *code coverage* del 97,47% vist a la figura 9. El 3% restant correspon a codi de protecció d'errors de sistema.

```
============================= Coverage summary =============================
Statements   : 95.88% ( 233/243 )
Branches     : 86.11% ( 62/72 )
Functions    : 95.24% ( 40/42 )
Lines        : 96.05% ( 219/228 )
============================================================================
-------------------------------|----------|----------|----------|----------|-------------------
File                           |  % Stmts | % Branch |  % Funcs |  % Lines | Uncovered Line #s
-------------------------------|----------|----------|----------|----------|-------------------
All files                      |    95.88 |    86.11 |    95.24 |    96.05 |
 APITestReportingTool          |    96.88 |     62.5 |      100 |    96.88 |
  index.js                     |    96.88 |     62.5 |      100 |    96.88 |                53
 APITestReportingTool/config   |      100 |      100 |      100 |      100 |
  test.js                      |      100 |      100 |      100 |      100 |
 APITestReportingTool/controllers |       94 |    87.04 |    94.44 |    94.24 |
  reporter.js                  |      100 |      100 |      100 |      100 |
  reviewer.js                  |      100 |      100 |      100 |      100 |
  scheduler.js                 |    89.89 |       75 |       92 |    90.59 | ... 83,213,214,217
 APITestReportingTool/models   |      100 |      100 |      100 |      100 |
  scheduledTest.js             |      100 |      100 |      100 |      100 |
 APITestReportingTool/routes   |      100 |      100 |      100 |      100 |
  reporter.js                  |      100 |      100 |      100 |      100 |
  reviewer.js                  |      100 |      100 |      100 |      100 |
  scheduler.js                 |      100 |      100 |      100 |      100 |
-------------------------------|----------|----------|----------|----------|-------------------
```

Fig. 9: Informe en format de taula dels tests de caixa blanca executats sobre el codi de l'API principal.

7.5.4 Caiguda del sistema i pèrdua de dades

Al final de la fase Q2 es va realitzar l'entrega de la segona release amb grans canvis en l'arquitectura i especificacions, oferint una eina a primera vista funcional tant per planificar com per reportar tests. Es va començar a fer servir per planificar, guardar totes les suites de test disponibles i reportar. Poc després es van perdre totes les dades dels bancs de proves per un error en el sistema. La causa de l'error va ser la inserció de caràcters ascii no permesos, generats en copiar i pegar tests del banc de proves de l'aplicació antiga, provocant que MongoDB no reconegui els tests associats al nom del banc de proves, fent les dades no accessibles.

Aquest i altres problemes es van detectar durant l'entrega es podrien haver evitat realitzant *Exploratory Testing* i *Unit Testing* de manera més rigorosa. Van provocar un endarreriment en la planificació del projecte. Durant la Q3 es va dedicar la primera part a redissenyar el sistema de testing. A la figura 10 es pot veure l'augment de temps dedicat al testing després de trobar un error crític.

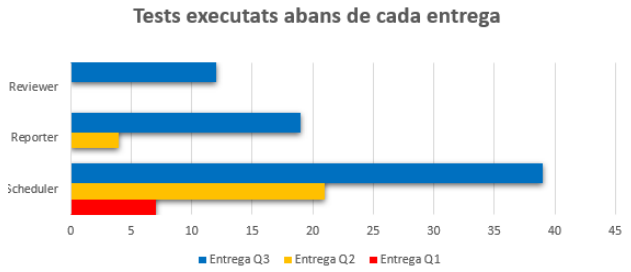

Fig. 10: Anàlisi del nombre de casos de prova de cada mòdul abans de cada entrega de codi al client al final de cada milestone.

7.6 Integració i Desplegament Continu

La integració continuada (CI) és el procés d'automatitzar la compilació i producció dels canvis, tant de codi com de test cada cop que un desenvolupador fa canvis al control de versions. El desplegament continu (CD), extensió del CI, es basa en el procés del desplegament i posada en marxa de l'entorn de producció després que el CI sigui satisfactori.

Ja que en una metodologia àgil és molt important distribuir noves versions de software als clients tan aviat com estigui disponible, s'han fet servir les pràctiques combinades *d'integració continuada i desplegament continu* [9] (CI/CD) per automatitzar la integració i desplegament de les diferents versions de l'aplicació. L'objectiu és minimitzar el temps entre el desenvolupament i la integració dels canvis a client. Cada vegada que es realitza un commit a la branca master, si passa tots els tests d'integració, unitat i E2E, s'executa un desplegament automàtic al servidor, és a dir, se sincronitza amb el repositori, actualitzarà llibreries, comprovarà les dependències i recàrrega el servidor.

Per CI/CD s'ha fet servir *Jenkins* [10] un programa d'automatització que s'instal·larà sobre el servidor on es fa el desplegament de l'aplicació.

7.7 Esquema MongoDB

MongoDB evita el tradicional esquema de taules relacionals en favor de documents JSON amb esquemes dinàmics, fent la integració de les dades en l'aplicació més fàcil i ràpid. Ha permès desenvolupar el sistema de base de dades en poc temps i l'estructura de documents dinàmics ha permès tenir objectes a la mateixa col·lecció amb camps diferents, aportant flexibilitat i reduint el volum de dades a tractar. Ofereix una fàcil adaptació amb NodeJS fent servir plugins com *Mongoose* [11]. També bon rendiment amb la correcta indexació per tractar amb grans volums de dades, i és que la clau del problema està en el fet que les insercions de tests sempre es fan d'una en una, però la informació es visualitza de manera massiva. A la figura 11 es pot veure l'esquema de dades empleat a l'aplicació.

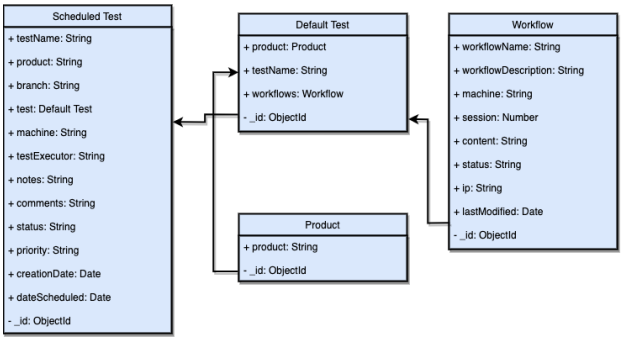

Fig. 11: Esquema de dades de l'aplicació amb totes les possibles dades dinàmiques i les seves relacions.

8 TECNOLOGIES

L'aplicació segueix un model SPA , la pàgina web només es carrega una vegada i després es van actualitzant els elements necessaris al Document Object Manager (DOM). Aquesta arquitectura aconsegueix que l'aplicació sigui més interactiva i es redueixin el nombre de peticions HTTP al servidor, ja que no s'ha de recarregar per cada vegada que es modifiqui el contingut de la web.

Aquesta aplicació té el frontend i el backend separats. Permet tenir una millor organització del codi, en la qual el backend té la lògica i executa les peticions a la base de dades i el frontend s'encarrega de mostrar els elements i fer peticions de recursos a l'API. L'arquitectura de l'aplicació es pot veure a la figura 12. S'ha desenvolupat una API privada dins dels servidors d'HP que no s'ha entregat al projecte, això es deu al fet que d'aquesta API s'agafa informació sensible de les impressores i dels servidors d'HP. Aquesta API també desenvolupada en Node.js, executa a les màquines d'on es vol extreure la informació uns scripts escrits en Bash, que recullen dades mitjançant un túnel SSH entre l'API i la màquina, oferint una millor interacció amb l'eina, automatitzant i oferint informació extra.

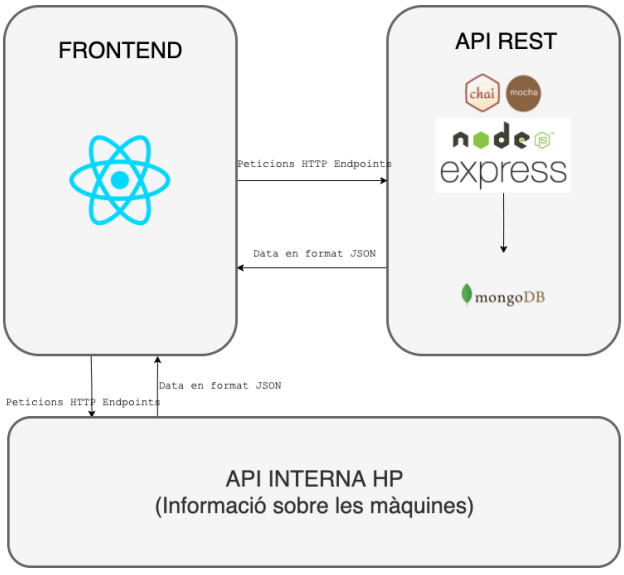

Fig. 12: Arquitectura de l'aplicació amb un client web desenvolupat i el flux de peticions HTTP entre servidors.

8.1 MERN Stack

Aquesta arquitectura permet desplegar diferents clients per diferents departaments interessats en l'eina utilitzant un mateix servidor per l'API. Tota la tecnologia que es fa servir és Javascript, utilitzant un únic llenguatge per Servidor/Client/BD. Les dades viatgen en JSON (Javascript Object Notation). L'Stack tecnològic s'anomena *MERN* [12] (MongoDB, Express, ReactJS i NodeJS).

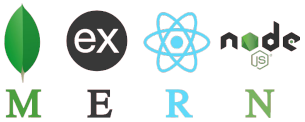

Fig. 13: Il·lustració de l'stack MERN, aplicat pel desenvolupament del projecte.

- *MongoDB* [13] és un sistema gestor de bases de dades NoSQL que guarda la informació en format de documents. És on s'emmagatzemen les dades dels tests realitzats. És un gestor molt eficient quant a lectura de dades, requisit important per al nostre sistema.
- *Express* [14] és una infraestructura d'aplicacions web Node.js, proporciona l'encaminador i els punts d'accés a l'API. Permet amb poques línies de codi generar els punts d'entrada, mètodes HTTP i *middlewares* per crear les API's d'una manera senzilla i amb un rendiment excel·lent.
- *ReactJS* [15] és la tecnologia en la qual es desenvolupa el frontend. És una llibreria de Javascript desenvolupada per Facebook molt popular en l'entorn laboral. El seu punt més fort és la creació de webs SPA, interfícies d'usuari i la reutilització de codi, element bàsic per crear una aplicació ràpida, flexible i reutilitzable.
- *Node.js* [16] és una llibreria de Javascript per servidor, pensada per crear aplicacions en xarxa escalables i optimitzades. Serà el llenguatge utilitzat per la implementació de l'API amb l'ús d'Express tant com per gestionar les dades i les connexions amb la base de dades MongoDB.

9 RESULTATS

En aquest apartat es tractaran els objectius assolits.

S'han pogut dissenyar els mòduls per planificar, reportar i revisar els tests. La informació ja no ha de ser duplicada en diferents sistemes com Outlook i Excel, ja que l'eina permet una exportació dels resultats des de la mateixa al format adient per redactar e-mails de planificació i de reporting requerits.

S'ha treballat conjuntament amb els usuaris de cada part de l'eina per arribar a desenvolupar una interfície i funcionalitats específiques, adaptades a les necessitats del projecte, obtenint un alt grau d'automatització en tasques rudimentàries i reduint la taxa d'errors humans.

Per valorar els resultats, també s'ha d'avaluar el grau de satisfacció i l'ús de l'aplicació pels diferents stakeholders. S'han realitzat enquestes de satisfacció a cada milestone per tenir una imatge clara de l'acceptació i el compliment dels

requeriments. Un resum d'aquestes es pot veure a la figura 14. Es pot interpretar el decreixement durant la Q2 pels errors trobats, també el creixement en la satisfacció dels interessats un cop els mòduls que fan servir són a producció.

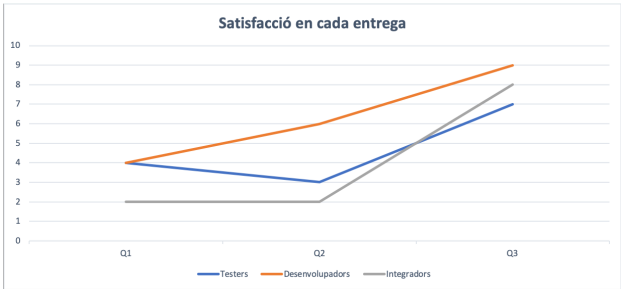

Fig. 14: Gràfica sobre el creixement de la satisfacció dels diferents stakeholders a mesura que ha avançat el projecte.

L'enginyer de test s'ha pogut adaptar a l'eina de manera immediata, realitza les planificacions de manera més precisa, en menys temps i no ha volgut tornar a utilitzar les eines anteriors. La funcionalitat més desitjada que va demanar era poder exportar la planificació diària en format d'e-mail amb un disseny formal i es va implementar durant la Q3, reduint dràsticament el temps requerit per realitzar la planificació i exportar-la a format e-mail. A la figura 15 es pot veure la reducció de temps en la planificació.

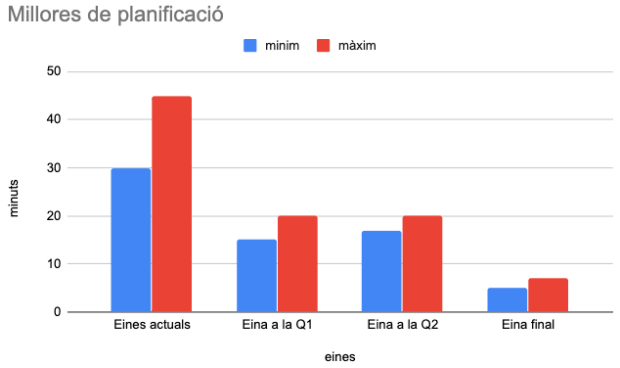

Fig. 15: Gràfica sobre un mostreig durant dues setmanes del temps mitjà invertit a realitzar la planificació dels tests.

Els testers després d'oferir feedback per millorar l'eina l'han continuat fent servir, ja que els evita molts errors i accelera el procés de reportar, informació vista a la figura 16.

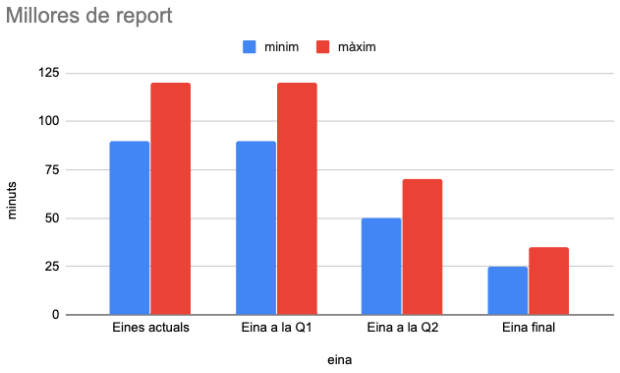

Fig. 16: Gràfica sobre un mostreig durant dues setmanes del temps mitjà invertit a realitzar els reports dels tests diaris.

Per unanimitat es va demanar una automatització en el sistema de reporting, constant de dos processos. El primer un sistema d'obtenció de dades de manera automàtica de les impressores des de l'eina web, un requisit al qual se'l va donar prioritat a la Q3. Per altra banda, una opció per poder exportar a format e-mail tots els tests executats amb les dades que han introduït a l'eina, ja que les dades es volen tenir tant a l'aplicació com via e-mail per poder visualitzar-les des de fora de la xarxa R+D d'HP.

La part de visualització de resultats especificada al reviewer ha rebut una acceptació global i és la més útil per als integradors i mànagers, ja que els ha permès tenir un històric de tots els tests executats amb un cercador potent, filtrant només pels resultats dels productes en els quals treballen. Per altra banda també es va desenvolupar un dashboard amb informació genèrica, mètriques i resultats que permeten veure en una sola pàgina molta informació d'interès per al projecte.

10 LÍNIES FUTURES

En aquest apartat s'explicaran funcionalitats que no s'han pogut portar a terme durant la durada del TFG però que es desenvoluparan en els pròxims mesos.

L'enginyer de test va demanar en la fase final del projecte un sistema drag-and-drop per ordenar els tests a la taula del *scheduler* i l'opció d'afegir nous workflows en el procés de report d'un test. Aquestes dues funcionalitats requereixen redissenyar mòduls i es van deixar com a línies futures per poder entregar les funcionalitats bàsiques amb la millor qualitat possible.

El desenvolupament d'una aplicació mòbil no es va poder realitzar per falta de temps. Aquest objectiu s'ha mitigat amb una aplicació web adaptable a tots els dispositius, deixant aquest objectiu en una línia futura.

11 CONCLUSIONS

Es va definir l'àmbit del projecte i els objectius que es volien obtenir. La investigació sobre les eines, la viabilitat i les necessitats del projecte van verificar la necessitat de continuar el projecte i seguir investigant per arribar al punt on se satisfan totes les necessitats de l'empresa. A mesura que avançava el projecte, es van trobar més oportunitats per desenvolupar noves funcionalitats per beneficiar tant a l'empresa com als treballadors.

Tot i que els objectius principals s'han aconseguit, la qualitat de l'eina pot millorar molt i es seguirà treballant si rep la suficient acceptació i es continua fent servir en un termini de temps més llarg. Els majors limitadors de la qualitat han sigut:

- Inexperiència en les eines utilitzades. En l'àmbit acadèmic, a la menció d'Enginyeria del Software, no s'expliquen conceptes de les eines que actualment es fan servir al mercat laboral, com *ReactJS o NodeJS*. Per això ha fet falta un període d'adaptació aquestes eines i han fet falta moltes més hores de les esperades. Aquesta decisió de fer servir noves eines, m'ha aportat molts coneixements sobre les mateixes i és una de les millors decisions que vaig prendre a l'hora de desenvolupar el projecte.

- Conflictes amb xarxa R+D d'HP. Es van detectar conflictes de proxy entre l'aplicació i els servidors d'HP, deguts a la infraestructura de l'empresa, la qual en treballar a un sector R+D, està blindada entre proxies. Es va haver de negociar amb el departament de ciberseguretat per trobar una solució i van oferir una màquina virtual sense limitacions de proxies i que permet una comunicació bidireccional amb la xarxa. Aquest problema va endarrerir molt el desenvolupament, ja que es van trigar setmanes a diagnosticar els problemes i arribar a una solució adient.
- Situació personal. El fet d'estudiar per assignatures d'un semestre complet, a la vegada que treballar jornada completa i desenvolupar el TFG ha sigut un factor limitador de temps i d'estres però que ha valgut la pena pels resultats obtinguts.

El desenvolupament d'aquest projecte ha refermat i millorat tots els coneixements adquirits durant la carrera, m'ha permès orientar les meves possibilitats laborals amb l'adquisició de coneixements específics en les tecnologies que més m'interessen i m'ha donat una millor visió global sobre totes les fases que ha de seguir un projecte per desenvolupar-se correctament sota els estàndards de l'enginyeria del software.

AGRAÏMENTS

Agrair el projecte a tots els companys del meu equip de treball per l'ajuda a l'hora de testejar l'aplicació. A Albert Jordà, enginyer de test que ha aportat coneixements i idees en el desenvolupament, als meus tutors, d'empresa, Ignacio Justel i de la universitat, Xavier Otazu, que han aportat maduresa i professionalitat i m'ha guiat sobre les pràctiques correctes per al bon desenvolupament. Finalment a la meva família i parella pel suport emocional i la paciència.

REFERÈNCIES

[1] *Tom, Creating a Single-Page App in React using React Router* 6 Juny de 2019 [online] Disponible a: https://medium.com/swlh/build-your-first-react-single-page-app-ad341f86b920 Consulta: 20 Agost del 2019

[2] *Andrew Gasanoff, Smoke Testing Suite: What it is, Why You Need it, and How to Automate*, 12 Juny del 2019 [online]. Disponible a: https://www.functionize.com/blog/smoke-testing-suite-what-it-is-why-you-need-it-and-how-to-automate/ Consulta: 19 Septembre del 2019.

[3] *Agoi Abel, Pagination in ReactJS*, 6 Març del 2019 [online]. Disponible a: https://medium.com/@agoiabeladeyemi/pagination-in-reactjs-36f4a6a6eb43 Consulta: 11 Desembre de 2019

[4] *Las metodologías Scrum y MVP pueden trabajar juntas* [online]. Disponible a: https://www.heflo.com/es/blog/metodologia-agil/scrum-mvp/ Consulta: 02 Octubre del 2019.

[5] WBS Wikipedia [online]. Disponible a: https://en.wikipedia.org/wiki/Work_breakdown_structure Consulta: 21 Novembre del 2019

[6] *Demir Selmanovica, Beginner's Guide To Managing Software Development with Kanban and Trello* [online]. Disponible a: https://www.toptal.com/agile/guide-managing-development-kanban-trello Consulta: 15 Septembre del 2019.

[7] *Testeando Javascript con Mocha y Chai* 25 Gener del 2018 [online]. Disponible a: https://www.paradigmadigital.com/dev/testeando-javascript-mocha-chai/ Consulta: 15 Octubre del 2019.

[8] *Robert C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship* Agost del 2018 ISBN:978-0-13-235088-4 Lectura: 15 Juliol del 2019.

[9] *Moshe Ezderman, How to set up CI/CD Pipeline for a node.js app with Jenkins*, 28 Febrer del 2018 [online]. Disponible a: https://medium.com/@mosheezderman/how-to-set-up-ci-cd-pipeline-for-a-node-js-app-with-jenkins-c51581cc783c Consulta: 05 Novembre del 2019.

[10] Jenkins oficial [online] Disponible a: https://jenkins.io/ Consulta: 4 Gener del 2020

[11] Mongoose oficial [online] Disponible a: https://mongoosejs.com/docs/ Consulta: 4 Octubre del 2019

[12] *Sampol, ¿Que es MERN ?* 22 Gener del 2017 [online] Disponible a: https://platzi.com/blog/que-es-mern-stack-javascript/ Consulta: 9 Septembre del 2019.

[13] MongoDB Wikipedia [online]. Disponible a: https://es.wikipedia.org/wiki/MongoDB Consulta: 21 Septembre del 2019.

[14] Express oficial [online]. Disponible a: https://expressjs.com/es/ Consulta: 17 Septembre del 2019.

[15] *Hamza Mahmood, Advantages of Developing Modern Web apps with React.js* 27 Maig del 2018 [online] Disponible a: https://medium.com/@hamzamahmood/advantages-of-developing-modern-web-apps-with-react-js-8504c571db71 Consulta: 15 Septembre del 2019.

[16] Node.js oficial [online]. Disponible a: https://nodejs.org/en/ Consulta: 17 Septembre del 2019.

APÈNDIX

A.1 Planificació del Projecte Inicial

Objectius	Q1		Q2			Q3	
	Agost	Septembre	Octubre	Novembre	Desembre	Gener	Febrer
Documentació	Elicitació de requisits Reunió Stakeholders 1 Anàlis de mercat	Mapa d'Objectius Elicitació de requisits Mapa de Pantalles	Reunió Stakeholders 2 Correcció Requisits Correcció Mapa de Pantalles	Manuals d'usuari Diagrames UML	Manuals d'usuari Correcció de Documents Revisió d'especificacions	Manuals d'usuari	Informe Final Presentació
Scheduler	Disseny inicial Desenvolupament	Desenvolupament Entrega Alpha al Client	Correcció d'errors Refactorització			Refactorització	
Reporter		Disseny	Disseny Desenvolupament	Desenvolupament Refactorització	Correcció d'errors Aplicar correccions Alpha	Refactorització	
Reviewer			Disseny	Disseny Desenvolupament	Disseny Desenvolupament	Desenvolupament Refactorització Correcció d'errors	
Testing	UTest API Scheduler	UTest API Scheduler UTest FrontEnd Scheduler	UTest API Reporter Test d'Integració	UTest FrontEnd Reporter Test d'Integració	UTest FrontEnd Reporter Test d'Integració	ontEnd Reviewer Test d'In	UTest Global Test d'Integració
Sistema	Setup entorn NodeJS React Setup entorn de Testing Setup Logging API	Setup Docker Setup Client per entrega	Setup CI/CD			Preparació entregable	
Informes		Preparar Informe Inicial TFG	Inicial TFG Informe de Pr	Preparar Informe Final Informe de Progrés 2	Preparar Informe Final Preparar Presentació Informe de Progrés 2	Preparar Informe Final Preparar Presentació	Informe Final PPresentació
Entregues		Entrega Alpha Scheduler		Entrega Alpha Reporter Entrega Scheduler Millorat		Entrega Beta Aplicació	Entrega Aplicació

Fig. 17: Taula que mostra la planificació inicial del projecte amb els objectius i tasques a assolir durant cada fase.

A.2 Planificació del Projecte Després de la Milestone Q2

Objectius	Q1		Q2		Q3		
	Agost	Septembre	Octubre	Novembre	Desembre	Gener	Febrer
Documentació	Elicitació de requisits Reunió Stakeholders 1 Anàlis de mercat	Mapa d'Objectius Elicitació de requisits Mapa de Pantalles	Reunió Stakeholders 2 Correcció Requisits Correcció Mapa de Pantalles	Manuals d'usuari Diagrames UML		Manuals d'usuari Correcció de Documents Revisió d'especificacions	Informe Final Presentació
Scheduler	Disseny inicial Desenvolupament	Desenvolupament Entrega Alpha al Client	Correcció d'errors Refactorització		Correcció d'errors Aplicar correccions Alpha		Refactorització
Reporter		Disseny	Disseny Desenvolupament	Desenvolupament Refactorització	Correcció d'errors Aplicar correccions Alpha		Refactorització
Reviewer			Disseny	Disseny Desenvolupament	Correcció d'errors Aplicar correccions Alpha	Desenvolupament Refactorització Correcció d'errors	
Testing	UTest API Scheduler	UTest API Scheduler UTest FrontEnd Scheduler	UTest API Reporter Test d'Integració	UTest FrontEnd Reporter Test d'Integració	Redefinició de tots els UTest Code Coverage	UTest FrontEnd Reviewer Test d'Integració	UTest Global Test d'Integració
Sistema	Setup entorn NodeJS React Setup entorn de Testing Setup Logging API	Setup Docker Setup Client per entrega	Setup CI/CD			Preparació entregable	
Informes		Preparar Informe Inicial TFG	Informe Inicial TFG Informe de Progrés 1	Preparar Informe Final Informe de Progrés 2	Informe de Progrés 2	Preparar Informe Final	Informe Final Presentació
Entregues		Entrega Alpha Scheduler		Entrega Alpha Reporter Entrega Scheduler Millorat		Entrega Beta Aplicació	Entrega Aplicació

Fig. 18: Taula que mostra la planificació del projecte després de les modificacions per els errors trobats durant la fase Q2.

A.3 Diagrames de Gantt de la planificació inicial i la planificació després de la Milestone Q2

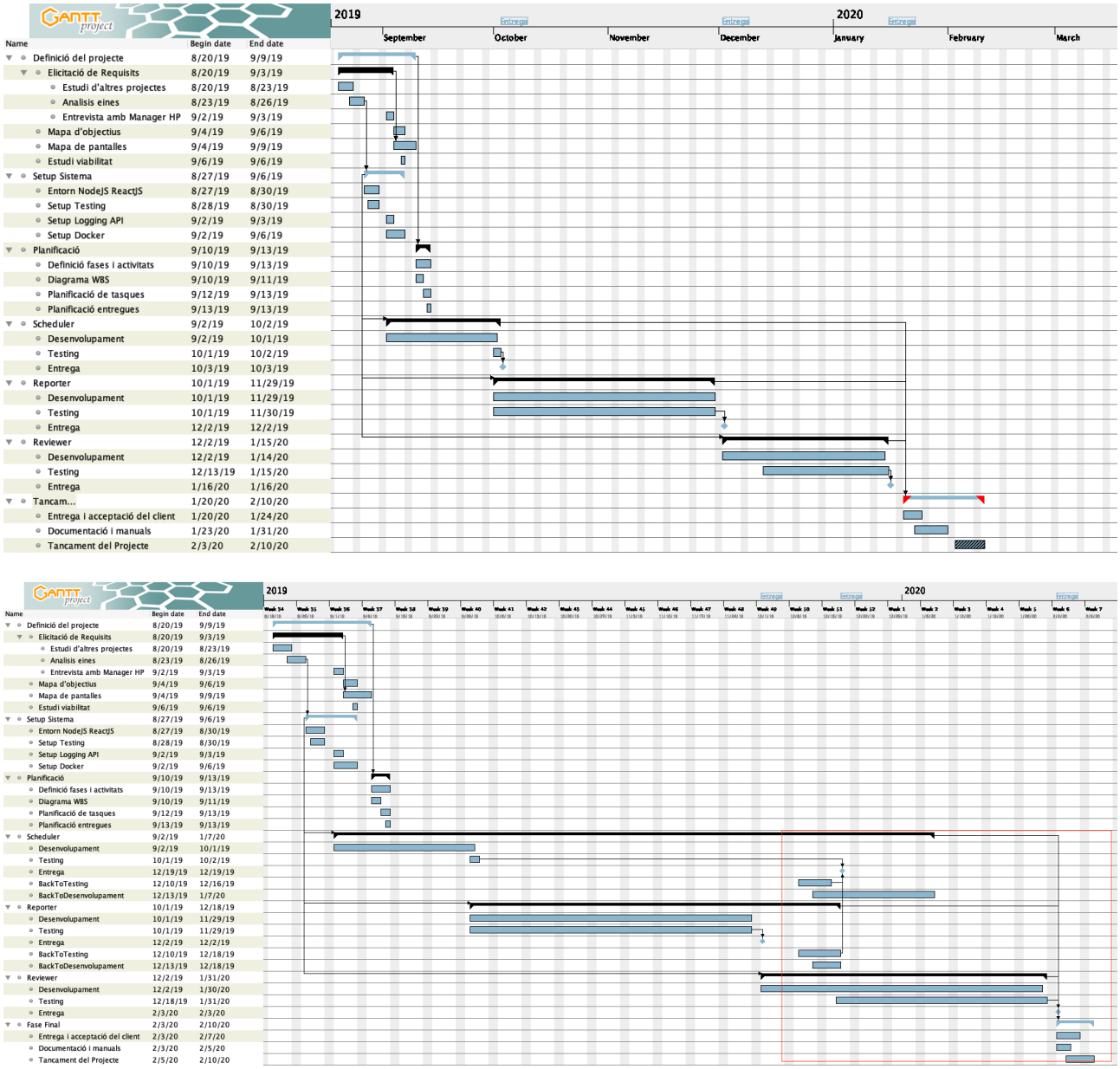

Fig. 19: Diagrama de Gantt que mostra la planificació del projecte abans i després de les modificacions per els errors trobats durant la fase Q2.