



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona



DISSENY I IMPLEMENTACIÓ D'UN SISTEMA D'EXECUCIÓ DE PROVES AUTOMÀTIQUES.

TREBALL DE FI DE GRAU

Grau en Enginyeria Informàtica

MANGO

Autor: Aleix Marro Querol

Director: Francesc Josep Muñoz Lopez

Ponent: Edelmira Pasarella Sanchez

Tutor GEP: Joaquin Deulofeu Aymar

22 de juny de 2021

Resum

L'objectiu d'aquest projecte és agilitzar el procés de validació en el cicle de desenvolupament de programari seguit per l'empresa Mango. Després d'analitzar les carències del procés actual i identificar les potencials millores, el projecte pretén trobar una solució eficient i funcional a llarg plaç per automatitzar aquest procés.

Resumen

El objetivo de este proyecto es agilizar el proceso de validación en el ciclo de desarrollo de software seguido por la empresa Mango. Tras analizar las carencias del proceso actual e identificar las potenciales mejoras, el proyecto pretende encontrar una solución eficiente y funcional a largo plazo para automatizar este proceso.

Abstract

The aim of this project is to speed up the validation process in the software development cycle followed by the company Mango. After analyzing the shortcomings of the current process and identifying potential improvements, the project aims to find an efficient and functional long-term solution to automate this process.

Índex

1	Introducció i context	6
1.1	Introducció	6
1.2	Context	8
1.3	Problema a resoldre	9
2	Abast	11
2.1	Objectius	11
3	Metodologia	13
3.1	Metodologia de treball	13
3.2	Entorns i eines de validació	14
4	Solució plantejada	15
4.1	Sistema d'execució de proves automàtiques (SEPA)	19
4.2	Eines/Software	22
4.2.1	Gherkin	22
4.2.2	Cucumber, Java y Selenium WebDriver	22
4.2.3	Browserstack	22
4.2.4	Jenkins	22
4.2.5	Eines addicionals	23
4.3	Implementació del SEPA	24
4.3.1	Framework	26
4.3.1.1	Estructura i característiques del projecte	26
4.3.1.2	PDP	28
4.3.1.3	PLP	32
4.3.1.4	Email	35
4.3.2	Jenkins	38
4.4	Solucions prèvies i alternatives	49
4.4.1	Cypress vs. Selenium	49
4.4.2	Python vs. Java	49
4.4.3	Kotlin vs. Java	50
4.4.4	RestTemplate vs. Retrofit	50
4.5	Possibles obstacles	51
5	Planificació temporal	52
5.1	Definició de tasques	52
5.2	Plans alternatius i obstacles	56
5.3	Estimacions i Gantt	57

6	Pressupost	60
6.1	Identificació del costos	60
6.1.1	Costos de personal	60
6.1.2	Costos genèrics	61
6.2	Estimació del costos	62
6.3	Control de gestió	63
7	Sostenibilitat	64
7.1	Autoavaluació	64
7.2	Dimensió econòmica	64
7.3	Dimensió ambiental	64
7.4	Dimensió social	65
8	Conclusions	66
9	Agraïments	68
10	Bibliografia	69

Índex de figures

1	Cicle de desenvolupament d'una metodologia en cascada [2]	6
2	Cicle de desenvolupament de funcionalitats i productes seguit per l'empresa	9
3	Cicle de treball seguit per una metodologia àgil [7]	13
4	Cicle de desenvolupament per entorns seguit per l'empresa	15
5	Mòdul "Et pot interessar" visible a la pàgina d'un producte	17
6	Llistat de productes <i>Abrics</i> dins la línia de <i>Dona</i>	17
7	Resposta del servei que ofereix recomanacions per l' <i>Email</i>	18
8	Diagrama a alt nivell de la solució plantejada	19
9	Diagrama representatiu de la solució plantejada	24
10	Estructura del framework d'automatització perso-test-fw	26
11	Procés d'execució de les proves E2E a la PDP	28
12	Plana de <i>BrowserStack</i> amb la informació de totes les execucions recents	31
13	Codi de la prova testWoman de PLP.java	33
14	Resultat de les proves a la PLP sobre la secció <i>prendas</i> de <i>Dona</i>	34
15	Tests d'integració pel servei <i>email-recommendations</i>	36
16	Resultats de l'execució dels tests pel servei <i>email-recommendations</i> . .	37
17	Logo del servidor d'automatització Jenkins	38
18	Selecció de la secció QA dins el servidor Jenkins	38
19	Selecció de la secció QA dins el servidor Jenkins	39
20	Plana prèvia a l'execució del <i>job</i> PERSO_PDP_E2E_Config	40
21	Configuració de l'origen del codi pel <i>job</i> PERSO_PDP_E2E_Config . . .	41
22	Exemple de configuració d'un paràmetre pel <i>job</i> PERSO_PDP_E2E_Config	42
23	Configuració de les variables d'entorn pel <i>job</i> PERSO_PDP_E2E_Config	43
24	Configuració de les credencials pel <i>job</i> PERSO_PDP_E2E_Config	43
25	Configuració del nom d'execució pel <i>job</i> PERSO_PDP_E2E_Config . . .	43
26	Opcions de nom d'execució pel <i>job</i> PERSO_PDP_E2E_Config	44
27	Comanda <i>Maven</i> d'execució pel <i>job</i> PERSO_PDP_E2E_Config	44
28	Comandes posteriors a l'execució del <i>job</i> PERSO_PDP_E2E_Config . .	45
29	Configuració de la connexió amb Teams del <i>job</i> PERSO_PDP_E2E_Config	46
30	Configuració del connector a Teams pel <i>job</i> PERSO_PDP_E2E_Config .	46
31	Notificació a Teams dels resultats de l'execució d'un <i>job</i>	46
32	Configuració de les execucions diàries pel <i>job</i> PERSO_PDP_E2E_Daily_PRE	47
33	Configuració de les execucions diàries pel <i>job</i> PERSO_PDP_E2E_Daily_PRO	48
34	Configuració del <i>job</i> PERSO_EMAIL_PRO	48
35	Primera part del diagrama de Gantt	58
36	Segona part del diagrama de Gantt	59

Índex de taules

1	Taula amb les tasques planificades i les seves estimacions	57
2	Taula amb els rols del projecte i el seu preu de facturació per hora . . .	60
3	Taula amb les despeses addicionals derivades del conveni de pràctiques	60
4	Taula amb els costos genèrics identificats al projecte	61
5	Taula amb totes les despeses identificades i el cost total d'aquestes . . .	62

1 Introducció i context

1.1 Introducció

Segons la definició oferida per l'Institut d'Estudis Catalans [1], la qualitat d'una cosa representa la superioritat d'aquesta en el seu gènere. Sabent això doncs, seria lògic pensar que per tota empresa la qualitat dels seus productes és una prioritat. El cas però, és que no sempre és així i menys encara en l'àmbit del programari.

En l'actualitat, amb la velocitat a la qual es produeix la innovació i els avenços tecnològics, per la majoria d'empreses de programari resulta més rellevant oferir un producte actualitzat amb les tendències que no pas garantir-ne el nivell de seguretat o qualitat que s'exigiria en altres sectors. Aquest fet es produeix principalment pel fet que el control de qualitat en el software ha estat sempre lent i costós i, en general, no permet veure resultats econòmics a curt plaç. Per sintetitzar, si una empresa té un producte amb errors però funcional, molts cops prefereix desenvolupar un nou producte abans que millorar i iterar sobre l'existent. Aquesta decisió ve donada pel fet que els resultats econòmics d'un producte nou són més fàcils d'observar però també per la por a quedar-se desfasat en un mercat constantment canviant i basat en tendències.

Si observem els processos de qualitat en sectors històricament importants com l'automoció o l'aviació, podem observar que el control de qualitat es realitza a tots els nivells i durant tot el procés de producció. En el cas del programari però, no sempre és així. En general, les empreses prefereixen no dedicar gaire esforç en assegurar la qualitat durant els desenvolupament d'un producte. Fan això basant-se en els arguments exposats anteriorment i en el fet que aquest podria ralentitzar i allargar el temps fins que l'usuari final rebés el producte. Es per això que, molts cops el control de qualitat en el programari es realitza un cop el producte ha estat desenvolupat (Vegis la figura 1). Fer-ho en aquest punt permet utilitzar personal menys qualificat tècnicament en un procés molts cops manual i repetitiu.

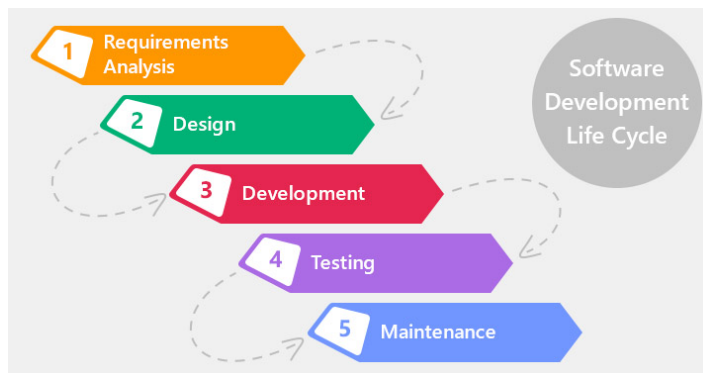


Figura 1: Cicle de desenvolupament d'una metodologia en cascada [2]

Aquest esquema en cascada [2] però, implica què quan es troba un error el cost de corregir-lo sigui molt més elevat que si s'hagués detectat en el moment en que es va produir. En identificar-lo al final del procés, resulta força complicat saber en quin punt s'ha produït i com solucionar-lo. És per això, que molts cops les empreses prefereixen presentar el producte sense corregir aquests errors, esperant que l'impacte d'aquests sigui menor que els beneficis o que no siguin detectats pels usuaris finals. No cal dir, que aquest sistema és insostenible doncs es sacrifica la qualitat vers la velocitat. Sobre aquest aspecte i d'altres força interessants en parla el criptògraf Bruce Schneier [3] al llibre *Click here to kill everybody*.

En els darrers anys, s'ha vist com algunes grans empreses han començat a abandonar aquesta metodologia explicada prèviament. Tot i mantenir-se en la tendència de primar la velocitat vers la qualitat, amb l'avanç de les eines i el coneixement tècnic, han començat a incorporar el control de qualitat al llarg del procés de desenvolupament del programari. D'aquesta incorporació, ha nascut el rol del QA (Quality Assurance) [4], una persona capacitada tècnicament destinada a assegurar la qualitat del producte. El QA s'encarrega de validar que els productes o desenvolupaments realitzats satisfan el criteris d'acceptació, que d'ara en endavant s'anomenaran AC, per les seves sigles en anglès. Els AC defineixen els requisits ha de satisfer el desenvolupament per poder ser presentat. En general el AC són definits pels *stakeholders* o propietaris del producte. Per determinar si es satisfà un AC, el QA realitza una prova on es comprova si s'obté el comportament o resultat esperat. Un exemple molt senzill de prova podria ser registrar-se a una xarxa social i esperar a rebre el correu de benvinguda. Amb aquesta prova s'estarien validant diferents AC, com podrien ser que el registre s'ha realitzat amb èxit o que el correu de benvinguda s'ha enviat correctament.

La diferència entre un QA i el tester tradicional [5] radica en el seu coneixement i la metodologia de treball. A diferència dels perfils més tradicionals, el coneixement tècnic permet al QA automatitzar les proves realitzades per a la validació dels AC, que d'altra forma s'haurien de realitzar manualment. Aquesta automatització permet augmentar l'eficiència de la feina realitzada i, en conseqüència, el benefici econòmic per l'empresa. Addicionalment ofereix robustesa al producte, doncs les proves es poden executar de forma periòdica, prenent consciència de l'estat del desenvolupament. També presenta un altre avantatge i, és que, al tenir coneixements tècnics es poden definir AC de caire tècnic, millorant la qualitat tècnica del producte. Un exemple de AC tècnic podria ser que un servei fos resistent a una petició inesperada.

Sobre l'automatització i creació de les proves necessàries per assegurar la qualitat en l'àmbit web es desenvolupa aquest Treball de Fi de Grau.

1.2 Context

Aquest treball es desenvolupa dins el marc d'un conveni de cooperació educativa per a la realització de pràctiques acadèmiques. En aquest conveni les parts implicades són la Facultat d'Informàtica de Barcelona (FIB), l'empresa Punto Fa, s.l i l'autor d'aquest treball, Aleix Marro Querol, com a estudiant del Grau en Enginyeria Informàtica (GEI). Com a estudiant de GEI, cursant l'especialització de Computació, en el moment de la realització del treball l'autor esta realitzant les pràctiques curriculars a l'empresa Mango, pertanyent a l'esmentada Punto Fa, s.l.

Per posar una mica de context, Mango [6] és una multinacional catalana dedicada al disseny, la fabricació i la comercialització de roba. És la segona empresa de *retail* més gran del país en volum de facturació i té més de 2700 punts de venda i 15000 treballadors repartits en més de 100 països. Essent una de les pioneres dins el sector, Mango va crear la seva pàgina web a Internet Mango.com el 1995 (shop.mango.com).

La venda de productes a través d'Internet era un sector que si bé presentava una tendència creixent durant els últims anys, encara no era majoritària en sectors com el tèxtil. Aquesta tendència però, s'ha accentuat notablement a causa de la pandèmia del Covid-19. Aquest increment en la facturació ha obligat a les empreses a focalitzar els esforços en els canals de venda en línia.

En el cas de Mango, aquests esforços s'han traduït en un augment de la plantilla del departament d'Online - IT, així com del nombre d'iniciatives realitzades per aquest. Les iniciatives estan formades per petits grups de treballadors i busquen descobrir nous punts d'acció per incrementar les ventes i la satisfacció del client. Donats els bons resultats, una d'aquestes iniciatives va acabar resultant en un equip, l'equip de Personalització, cada cop més extens. Aquest equip s'encarrega d'aportar personalització a la web de Mango i es divideix en dues seccions: l'equip de dades, encarregat de la gestió d'aquestes, i l'equip tècnic encarregat de la presentació de la personalització a l'usuari. Per exemple, si es vol oferir un mòdul on apareixen productes personalitzats a la web, l'equip de dades s'encarrega de decidir quin és el contingut d'aquest, mentre que l'equip tècnic s'encarrega de desenvolupar el mòdul per mostrar els productes al client. Dins d'aquest equip tècnic de Personalització és on es desenvolupa aquest treball.

Per últim i abans de prosseguir, es important clarificar per evitar ambigüitats, que al llarg d'aquest document es farà ús dels termes Personalització i personalització. El primer farà referència a l'equip tècnic i el segon al fet d'oferir un producte d'acord amb el perfil d'un usuari.

1.3 Problema a resoldre

Quan ens referim a la personalització, aquesta pot realitzar-se de moltes maneres i en molts àmbits diferents. En el cas d'una web per comprar productes com és el cas Mango, la idea més bàsica pot ser la d'oferir recomanacions en funció de les preferències de l'usuari. Aquestes recomanacions però, poden presentar-se de diverses maneres segons la ubicació on es troben i poden ser canvians al haver de satisfer certs requisits de negoci. El cas és que, el fet de personalitzar els productes oferits a l'usuari durant la seva navegació per la web, esdevé en múltiples desenvolupaments i múltiples requisits a satisfer. Aquest fet, juntament amb el comentat a la introducció sobre la constant adaptació de les empreses a les tendències, acaba resultant en una web sempre canviant. En aquest concepte de web constantment canviant, és necessari poder assegurar el desenvolupament segur de noves funcionalitats així com la preservació d'aquelles ja existents.

Per poder garantir la qualitat, és necessari comprovar en cada cas que els nous desenvolupaments satisfan els AC definits i que, al mateix temps, no deixen de satisfer aquells que es satisfien prèviament. La comprovació d'aquests AC, esdevé una feina tediosa i llarga si es realitza manualment i, això mateix pot contribuir al fet que no es realitzi correctament.

Identificat doncs aquest problema, la idea principal d'aquest treball és proporcionar una solució que permeti automatitzar la comprovació d'aquests AC fent ús de proves específiques per cada desenvolupament.

Per exposar amb més claredat on s'ubica aquest problema dins el context de Mango, s'ofereix a la figura 2 un esquema representatiu del cicle de desenvolupament seguit per l'empresa. Tal com es pot observar, durant el cicle de desenvolupament ja es troba present el rol de QA, tot i que actualment valida tots els desenvolupament de forma manual.

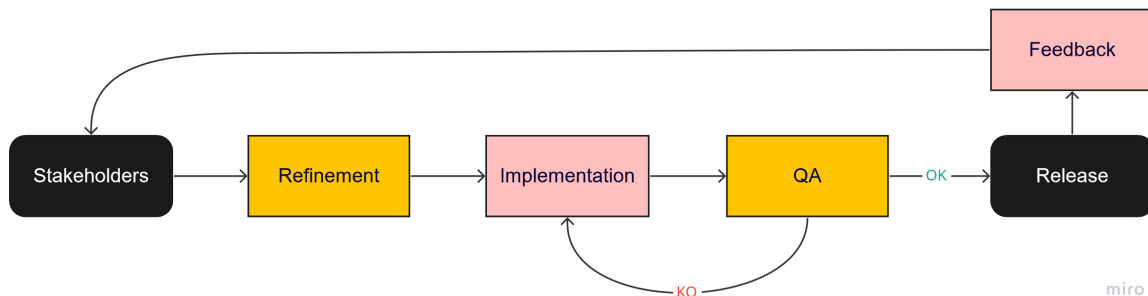


Figura 2: Cicle de desenvolupament de funcionalitats i productes seguit per l'empresa

Desgranant una mica les fases que componen aquest cicle s'obté que:

- **Stakeholders:** són els propietaris del producte. Generalment aquests són de caire econòmic i es troben a la primera fase del projecte, la de presa de decisions. En aquesta fase els *stakeholders* decideixen quin és el següent desenvolupament que cal realitzar. Aquest pot consistir en una nova funcionalitat o senzillament en una iteració sobre un producte o funcionalitat ja existent.
- **Refinement:** durant la fase de refinament l'equip tècnic valora la viabilitat del projecte juntament amb els *stakeholders*. En cas que el resultat sigui positiu es realitza un disseny de la solució a implementar així com dels diferents passos a seguir. És en aquesta fase on es presenten els requisits de negoci que acaben generant els AC que ha de satisfer el desenvolupament.
- **Implementation:** la fase d'implementació consisteix en la creació a nivell tècnic de la solució. Seguint els diferents passos acordats a l'etapa anterior, es van realitzant diferents tasques que permeten assolir l'objectiu marcat. Durant aquesta fase el rol de QA actua validant les tasques realitzades revisant les diferents parts del desenvolupament. En aquest punt aquest projecte farà incís, ja que en alguns desenvolupaments alguns AC no requeriran que tot el desenvolupament estigui acabat per ser comprovats. Si es dona el cas, existirà la possibilitat d'automatitzar-ne les comprovacions abans de la finalització de la implementació. Tot i així, el procés més costós pel QA es troba un cop finalitzada la implementació. Per aquest motiu s'ubica el seu paper després de la implementació.
- **QA:** un cop finalitzada la fase d'implementació, es troba el punt on el paper del rol de QA pren més importància. Un cop el nou desenvolupament està acabat, es necessita fer-ne una validació integral. En concret, cal revisar que compleix amb les expectatives i els objectius marcats, però també que conviu correctament amb la resta del sistema preexistent. Aquí el QA s'assegura que el desenvolupament satisfaci tots els AC definits de forma manual, executant un conjunt de proves regressives. En funció de si la validació és positiva o no, es procedeix a la publicació del desenvolupament o es retorna a la fase d'implementació per fer els canvis pertinents.

En aquest punt és on centra i tindrà més impacte aquest projecte, cercant una solució per automatitzar i agilitzar aquest procediment de validació.
- **Release:** tal com s'ha esmentat en el punt anterior, si el resultat del procés de validació és satisfactori es procedeix a la publicació del desenvolupament.
- **Feedback:** un cop s'ha publicat un nou desenvolupament, cal analitzar la reacció dels clients. La seva resposta i les diferents opinions generades seran utilitzades en la fase de presa de decisions i, en general, marcaran la direcció dels nous desenvolupaments.

2 Abast

2.1 Objectius

Sintetitzant l'explicació feta durant el procés d'identificació del problema i plantejament, els objectius principals d'aquest projecte són:

- Poder fer una comprovació eficient i ràpida dels AC d'un desenvolupament mitjançant l'execució de proves de forma àgil. Aquestes proves actualment es realitzen de forma manual fent que el procés sigui molt ineficient.
- Que les proves siguin automàtiques, és a dir, que un cop executades determinen si es compleixen els AC, sense necessitat d'intervenció per part de l'usuari que les executa.
- Poder iniciar l'execució d'aquestes proves de forma ràpida i senzilla per fer accessible la seva execució a qualsevol usuari.
- Poder executar aquestes proves de manera periòdica i automàtica.
- Tenir les proves en qüestió centralitzades en un sol punt, oferint la possibilitat de modificar-les i d'eliminar-ne o afegir-ne de noves si resulta necessari.

Pel que fa als objectius secundaris que es pretenen assolir amb la solució plantejada s'identifiquen:

- Poder definir els AC en alt nivell, amb un llenguatge proper i fàcil d'entendre per l'equip de negoci. Aquesta possibilitat facilitaria l'obtenció dels AC a partir dels requisits imposats per aquests.
- Permetre la configuració i parametrització de les proves realitzades.

Pel que fa al compliment d'aquests objectius, s'observen en dos casos diferents. Aquest projecte es limitarà a la primera esmentada:

- Els desenvolupaments de l'equip de Personalització a la web, en la versió per escriptori i la versió per telèfons mòbils.
- Els desenvolupaments de l'equip Personalització a les aplicacions per telèfons mòbils.

Tot i que els AC, en general, són comuns per ambdós casos, aquest projecte es centrarà en els desenvolupaments realitzats a la web. Tot i això, tal com queda reflectit en aquest document, la solució plantejada s'ha definit tenint en compte futurs desenvolupaments que permetin aplicar el sistema a les aplicacions mòbils també. La decisió de complir els objectius només pel primer cas, s'ha pres tenint en compte els següents punts:

- La implementació de les proves és diferent pels dos casos. Tot i que per un desenvolupament els AC tendeixen a ser el mateixos, el funcionament de les aplicacions és completament diferent al de la web. Aquest fet produeix que el procediment a seguir per realitzar les proves sigui també diferent. Així doncs, la implementació de les proves requereix de tecnologies diferents per les dues plataformes.
- El trànsit i facturació de la web és superior al de les aplicacions mòbils. Si bé és cert que el nombre d'usuaris en dispositius mòbils és superior al d'usuaris d'ordinador, la majoria d'aquests usuaris fan ús de la web en la seva versió per mòbil enlloc de les aplicacions. Aprofitant aquest punt, s'esmenta que a partir d'ara quan es faci referència a la versió de la web per escriptori s'utilitzarà el terme *Desktop* i quan es faci referència a la versió per mòbils s'utilitzarà *Mobile*.
- L'organització de l'empresa afavoreix els desenvolupaments al web, basant-se en el punt anterior. Aquest fet produeix que, en general, la personalització s'apliqui abans en aquest canal. En el moment de realització d'aquest treball, els punts on es preveu realitzar el control de qualitat, només es personalitzen en la versió web.

3 Metodologia

3.1 Metodologia de treball

Pel que fa a l'organització i sistema de treball que es preveu seguir en aquest projecte, es treballarà seguint una metodologia àgil [7] seguint el cicle de la figura 3. Més en concret, és treballarà seguint la metodologia *scrum* [8] fent ús d'equips. Tot i tractar-se d'un treball individual, cal recordar que aquest es realitza dins l'equip de Personalització.

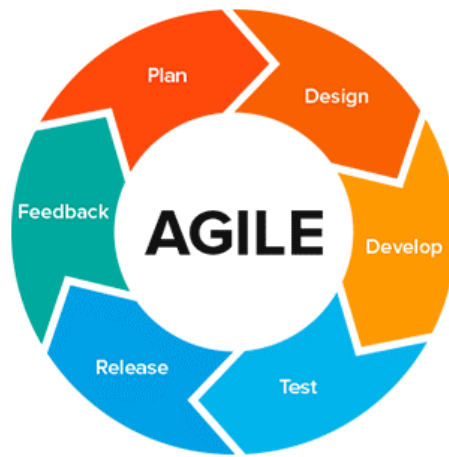


Figura 3: Cicle de treball seguit per una metodologia àgil [7]

El fet de seguir aquest sistema de treball resultarà especialment útil pel projecte tenint en compte els següents punts:

- La planificació es realitzarà a nivell de setmana, revisant-se contínuament i permeten així anticipar-se a endarreriments i bloquejos.
- En tractar-se d'un sistema canviant, el fet de treballar en constant contacte amb els desenvolupadors permetrà tenir constància en tot moment de les modificacions realitzades. Cal tenir en compte això ja que, en la majoria dels casos els nous desenvolupaments son avaluats en funció dels seus resultats (per exemple realitzant un test AB [9]). Aquests resultats no sempre son els esperats i poden acabar esdevenint en nous desenvolupaments o modificacions de requisits i AC durant la fase d'implementació.
- Per últim, el fet de treballar amb tecnologies desconegudes pot resultar en bloquejos, ja sigui per falta de coneixement o per l'arquitectura del sistema actual. El contacte constant permetrà solucionar aquests problemes amb molta més agilitat, aprofitant els coneixements de la resta de l'equip.

3.2 Entorns i eines de validació

Per últim, en referència a l'entorn de treball que s'utilitzarà, es defineixen els entorns i les eines de validació.

Pel que fa als entorns utilitzats, es seguirà el sistema utilitzat per l'empresa, basats en 3 entorns:

- **Cloudtest:** Entorn de desenvolupament on s'implementen els canvis i nous desenvolupaments per primer cop.
- **Preproducció:** Entorn pràcticament idèntic a producció, on es porten els canvis un cop es valida que compleixen amb tots els criteris de qualitat. Aquest entorn serveix com a punt de confluència per tots els nous desenvolupaments fets en entorn Cloudtest.
- **Producció:** L'entorn públic i dedicat a l'usuari final. S'inclouen els canvis en aquest entorn un cop s'ha validat que compleixen amb tots els criteris de qualitat i que son compatibles entre ells.

A priori pot semblar que aquests entorns de treball son independents d'aquest projecte, el cas però és que no es així. Per poder validar correctament la qualitat dels desenvolupaments, el sistema ha de poder funcionar amb independència de l'entorn que se li proporcioni. És per això que, durant el desenvolupament de la solució, es farà la distinció entre les proves executades a preproducció i les proves executades a producció.

En última instància, pel que fa a les eines de validació del projecte, es farà ús de les següents tecnologies:

- **Bitbucket** [10]: Sistema de control de versions basat en Git. Aquí és on s'ubicarà la part principal del projecte.
- **Jira** [11]: Entorn de gestió de projectes i seguiment d'errors. S'utilitzarà aquesta eina per treballar fent ús de la metodologia *scrum*.
- **Confluence** [12]: Base de documentació corporativa. Aquesta eina s'utilitzarà bàsicament per documentar i fer seguiment dels canvis realitzats durant el desenvolupament del projecte.

4 Solució plantejada

Un cop analitzat el problema, es planteja com a solució el disseny d'un sistema que permeti executar proves automàticament. Aquest sistema servirà per executar les proves definides sota demanda, fent així possible comprovar ràpidament i amb seguretat la satisfacció dels AC cada cop que es realitzi una modificació a la web. El fet de tenir-ho automatitzat permetrà també comprovar regularment l'estabilitat i el correcte funcionament del sistema amb un cost molt reduït, doncs no requerirà de supervisió humana. En aquest aspecte, es planteja l'automatització de les proves per a la seva execució en horaris de poc trànsit a la web. Pel que fa a les proves executades sota demanda, aquelles que han de permetre assegurar la qualitat dels nous desenvolupaments, és necessari que puguin ser executades en els diferents entorns explicats a la secció 3.2, doncs tots intervenen en la fase de desenvolupament.

Per detallar amb claredat com s'utilitzen aquest diferents entorns, s'observa a la figura 4 el cicle seguit quan es realitzen desenvolupaments. Aquest esquema és el mateix vist a la secció 1.3 a la figura 2, incloent ara el detall dels diferents entorns utilitzats. També inclou aquells punts on actua el rol de QA i on ho haurà de fer la solució dissenyada en aquest projecte que, a partir d'aquest punt s'anomenarà SEPA (Sistema d'Execució de Proves Automàtiques).

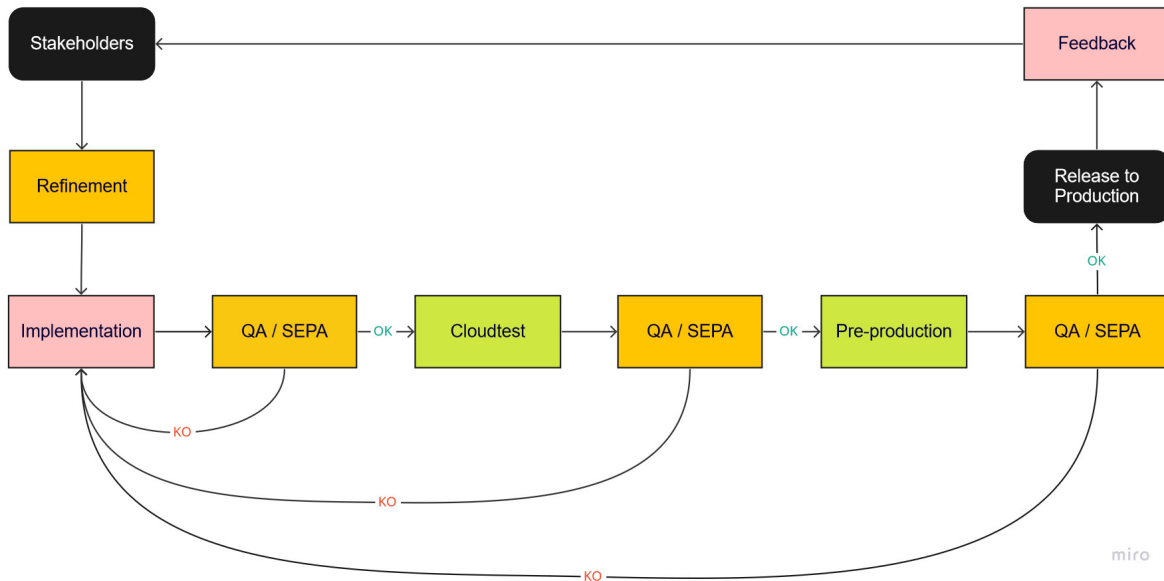


Figura 4: Cicle de desenvolupament per entorns seguit per l'empresa

Tal com s'observa a l'esquema, les proves s'hauran de poder executar en els següents moments:

- **Implementation:** durant la realització de les tasques que compondran cada desenvolupament. Tot i que no sempre serà possible en el cas dels nous desenvolupaments, per aquells ja existents es podran validar els AC durant iteracions posteriors.
- **Cloudtest:** un cop finalitzada la implementació, caldrà revisar si el nou desenvolupament és consistent i funciona correctament amb la resta de components del web.
- **Pre-production:** havent assegurat que el desenvolupament és compatible amb la resta de la web, caldrà assegurar que també ho sigui amb la resta de nous desenvolupaments. Totes les noves modificacions que estiguin pendents de ser publicades s'uniran en aquest entorn, on es validaran conjuntament.
- **Production:** per últim, si els resultats han estat positius en els entorns previs, és desplegarà a producció. També en aquest entorn caldrà assegurar que tot funciona correctament. D'altra banda, serà aquí especialment important, assegurar l'estabilitat d'alguns desenvolupaments, executant proves de manera continuada en el temps.

Abans de prosseguir, cal matisar que, fins aquest punt sempre de desenvolupaments i proves en termes generals, sense especificar-ne cap en concret. S'ha fet això ja que un dels objectius principals identificat a la secció 2.1 fa referència a la creació, modificació o eliminació de proves. És a dir, el que es busca amb aquest objectiu és que el sistema pensat sigui genèric. En el moment de realització d'aquest projecte, es preveu incloure dins el sistema aquelles proves necessàries per assegurar la qualitat en 3 punts clau de la web de l'empresa. Tot i això, aquest ha d'estar preparat per poder executar futures proves que assegurin la qualitat de nous desenvolupaments.

Sabent això, i abans de detallar els diferents tipus de proves incloses al sistema, és important introduir el 3 punts d'actuació d'aquestes:

- **PDP:** la Product Detail Page o pàgina de producte és aquella plana on es pot apreciar tota la informació de qualsevol producte que estigui a la venda a la web de Mango. En aquesta plana, actualment hi ha un mòdul de recomanacions anomenat "Et pot interessar" el que es pot veure a la figura 5. Aquest mòdul ha estat desenvolupat per l'equip de Personalització i marca el punt de partida d'aquest equip. Vist que els productes que ofereix són recomanacions dirigides, el benefici que reporta són molt superiors a la de la resta de mòduls. Així doncs resulta rellevant assegurar-ne el correcte funcionament i estabilitat al llarg del temps. Per aquest mòdul es crearan proves que n'hi comprovin la presència però també els diferents components i comportaments.

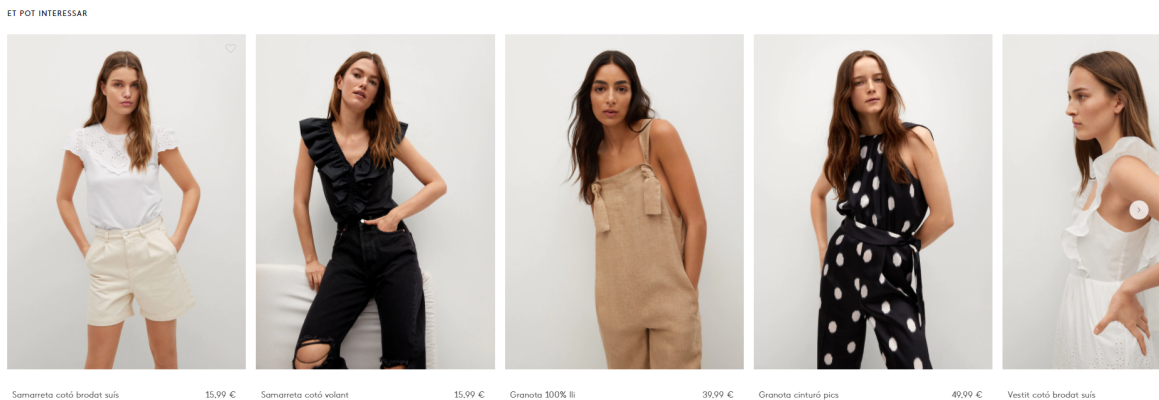


Figura 5: Mòdul "Et pot interessar" visible a la pàgina d'un producte

- **PLP:** la Products List Page o llistat de productes és aquella plana on es poden observar els múltiples productes disponibles a la venda d'una secció en concret. Des del punt de vista d'un usuari és el lloc des d'on accedeix a una pàgina de producte fent-hi *click* a sobre. A la web de Mango hi ha múltiples llistats, els més importants però es corresponen amb les diferents famílies de productes dins les línies que ofereix la marca. Un exemple de llistat seria per exemple el llistat d'*Abrics* dins la línia de *Dona* com s'observa a la figura 6. En aquests llistats esmentats, l'empresa esta començant a oferir personalització, mostrant en primera posició 4 productes personalitzats. Així doncs, el que es buscarà es crear proves que comprovin si aquests 4 productes son o no personalitzats.

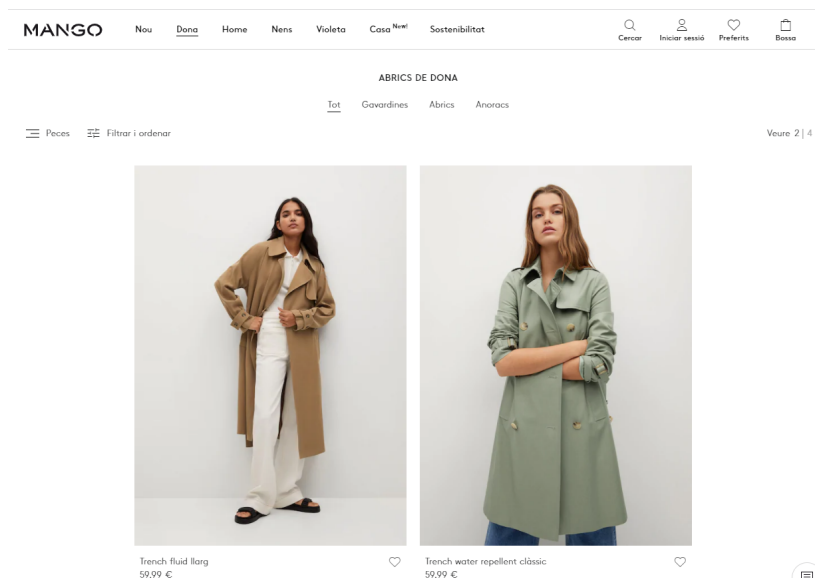


Figura 6: Llistat de productes *Abrics* dins la línia de *Dona*

- **Email:** aquest serà el tercer punt d'actuació de les proves creades en aquest sistema. A diferència dels productes anteriors, en aquest cas no es tracta d'un producte com a tal del qual el client pugui fer-ne ús directament, sinó d'una part d'aquest. Com a part de l'estratègia de *marketing*, l'empresa envia múltiples correus als usuaris. La idea és que progressivament aquests correus continguin cada cop més contingut personalitzat. Per aportar aquesta personalització, l'equip encarregat de la seva gestió fa ús d'un servei pertanyent a l'equip de Personalització. En tractar-se d'una iniciativa recent, les comprovacions en aquest cas seran més senzilles. La idea però, és poder assegurar que el servei funciona correctament i ofereix una resposta desitjada. Es pot veure en aquest cas, una imatge de la resposta del servei per una petició en concret a la figura 7.

```
{
  "recommendationsType": "complementary",
  "recommendedProducts": [
    {
      "recommendationType": "complementary",
      "id": "17050193",
      "colorId": "99",
      "name": "Mono fluido cinturón",
      "price": {
        "price": 29.99,
        "currency": "EUR",
        "salePrice": "29,99 €"
      },
      "url": "/es/mujer/vestidos-y-monos-monos/mono-fluido-cinturon_17050193.html?c=99",
      "image": {
        "relativeUrl": "/T1/fotos/S20/17050193_99.jpg?ts=1619507120326",
        "altText": "Mono fluido cinturón - Plano medio"
      }
    },
    { ...
  ]
}
```

Figura 7: Resposta del servei que ofereix recomanacions per l'*Email*

Un cop s'han detallat els diferents punts i situacions en que actuarà la solució adoptada, es procedeix a detallar-ne la seva composició i funcionament. Abans de començar però, és important explicar els diferents punts que compondran aquest apartat.

En aquest apartat s'exposarà inicialment la solució tècnica adoptada, especificant primer tots els components d'aquesta (apartat 4.1), així com les eines emprades per a la seva implementació (apartat 4.2). Posteriorment, s'especificarà detalladament com s'han utilitzant aquestes eines en la implementació d'aquesta solució, explicant en detall el funcionament i implementació de cada component (apartat 4.3). Per últim, es farà una justificació de les eleccions preses pel que fa a les eines emprades. Durant la fase de disseny, s'han valorat múltiples opcions per implementar cada component de l'estructura del projecte. S'avaluaran doncs, les diferents opcions plantejades i els motius pels quals s'ha escollit la solució detallada a continuació (apartat 4.4).

4.1 Sistema d'execució de proves automàtiques (SEPA)

Després d'haver plantejat la solució a grans trets i havent explicat els punts concrets on haurà d'actuar, ja es pot decidir una arquitectura per a la solució desitjada. Per l'elecció d'aquesta s'han tingut en compte els següents punts:

- L'objectiu més important del sistema es poder executar proves que assegurin la qualitat d'un desenvolupament. Això requereix d'una plataforma que permeti executar les diferents proves disponibles.
- L'altre objectiu indispensable és que les proves disponibles, puguin ser executades automàticament de forma periòdica o bé sota demanda. El més important però, és que la seva execució resulti fàcil i pràctica. Així doncs, es requereix d'un sistema que, fent ús de la plataforma esmentada, permeti executar les proves en aquestes modalitats.
- Per últim, s'han tingut en compte també altres solucions adoptades a l'empresa per resoldre problemes de caire similar.

S'introdueix aquesta secció amb un esquema a alt nivell de l'arquitectura que es planteja disponible a la figura 8. Aquest servirà com a punt de partida, oferint una visió general del sistema i permetent introduir les diferents parts que el composaran.

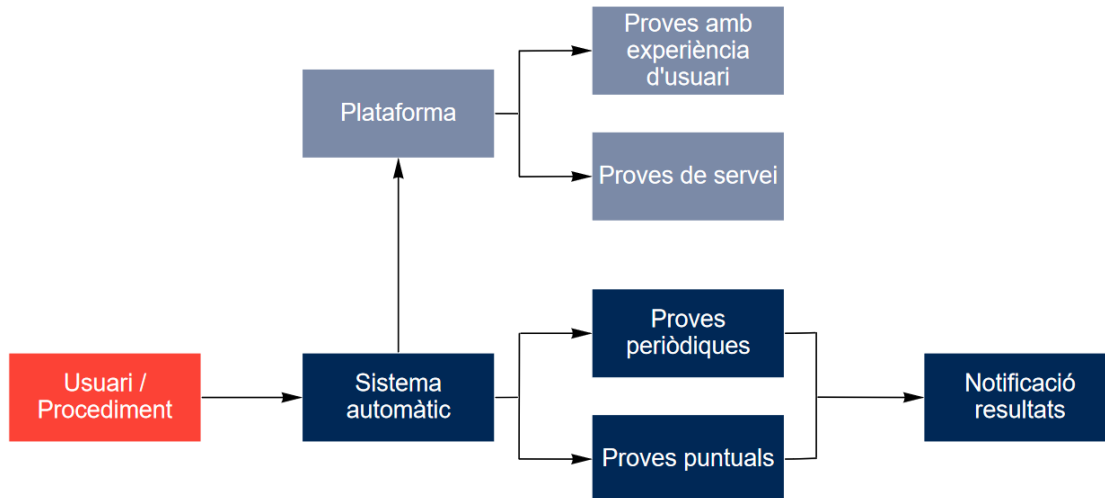


Figura 8: Diagrama a alt nivell de la solució plantejada

Es procedeix identificant i explicant els diferents components del sistema:

- **Usuari / Procediment:** En primera instància estarà l'usuari, agent o procediment que farà ús del sistema, utilitzant l'eina o servei escollits per a realitzar l'automatització. El sistema podrà ser utilitzat per tot aquell tingui accés al servei d'automatització. L'objectiu principal es que pugui ser utilitzat fàcilment pels rols de QA però també pels desenvolupadors o qualsevol usuari que tingui necessitat de realitzar comprovacions.

D'altra banda, es parla de procediment ja que es pot donar el cas en que sigui un disparador o un altre procediment el que requereixi d'executar les proves. Vegis per exemple el cas en que es vulgui posar com a requisit per afegir una nova funcionalitat, que el resultat d'unes proves sigui positiu. En aquest cas es podria posar un pas al procés encarregat d'afegir aquesta nova funcionalitat, que utilitzes el sistema per llençar les proves pertinents.

- **Sistema automàtic:** Es tracta d'aquella eina o servei que permetrà llençar i automatitzar les proves que es definiran durant la implementació de la Plataforma.
- **Plataforma:** Aquest component representarà la part principal d'aquest treball. Serà el programa que contindrà i executarà les proves i en generarà els resultats. El punt més important d'aquest component és que ha de ser una estructura genèrica, és a dir, ha de permetre afegir i implementar noves proves més enllà de les que es desenvoluparan en aquesta primera versió del sistema.
- **Tipus de proves:** la plataforma ha de permetre llençar tot tipus de proves que es vulguin definir. Tot i això, en aquesta primera versió es definiran 2 tipus diferents de proves, en funció de la metodologia utilitzada per assegurar la qualitat dels desenvolupaments actuals: PDP, PLP i Email.
 - **Proves d'experiència d'usuari:** També anomenades proves d'extrem a extrem, seran aquelles proves que permetran avaluar la qualitat del producte simulant el comportament d'un usuari. Consistiran bàsicament en executar un navegador en un dispositiu de proves, i seguir els mateixos passos i navegació que seguiria un usuari final. Utilitzant aquesta metodologia, es simularan tots aquells comportaments que es volen comprovar. Si l'execució finalitza, voldrà dir que els resultats han estat els esperats i, en conseqüència, que el nivell de qualitat és l'esperat. Aplicades al cas de Mango, s'utilitzaran aquestes proves per assegurar la qualitat a la PDP. Aquestes proves són molt fiables, ja que repliquen un comportament del desenvolupament en les mateixes condicions que es donen a producció. D'altra banda però, tenen un cost en recursos i temps més elevat que altres tipus de proves. El fet de requerir d'un dispositiu real i haver d'executar de manera seqüencial tots els passos per comprovar un punt concret, fa augmentar substancialment aquests costos.

- **Proves de servei:** Aquestes proves, permetran comprovar el correcte funcionament d'aquells serveis que utilitza un producte. Seran especialment útils ja que tenen una càrrega en recursos i temps d'execució molt inferiors a les proves d'extrem a extrem. En el cas d'aquest sistema serviran pels desenvolupaments de la PLP i Email.
- **Moments d'execució de les proves:** tal com s'ha especificat a l'apartat d'objectius, els diferents tipus de proves definits poden haver d'executar-se per diferents motius. En funció de quant s'hagin d'executar aquestes proves, es distingirà entre dos modalitats d'execució.
 - **Proves periòdiques:** Pels dos tipus de proves que s'han identificat, hi haurà execucions que es realitzaran de manera automàtica cada determinat temps. Aquestes execucions permetran assegurar l'estabilitat del producte, així com el seu correcte funcionament al llarg del temps.
 - **Proves puntuals:** Aquestes proves, executables sota demanda, permetran avaluar el producte en un moment determinat. Aquesta opció permetrà fer comprovacions quan es facin modificacions al producte per assegurar que segueix funcionant correctament.
- **Notificació resultats:** Per últim és configurarà una connexió del sistema automàtic per notificar dels resultats a les proves.

4.2 Eines/Software

Un cop identificats els diferents components del sistema, es procedeix a explicar les eines que s'utilitzaran per a la implementació d'aquests. L'elecció de les tecnologies no ha estat arbitrària, en alguns casos s'han escollit aquelles imposades per la companyia, i en altres s'han valorat múltiples opcions. Es veurà amb més detall a l'apartat 4.4 com s'ha realitzat aquesta presa de decisions.

4.2.1 Gherkin

Gherkin [13] és un DSL o Llenguatge Específic de Domini que serveix per aproximar les visions de negoci i desenvolupament. Bàsicament serveix per a la definició d'escenaris utilitzant un llenguatge d'alt nivell els quals es corresponen amb el diferents AC que es volen satisfer. S'utilitzarà per les proves E2E implementades per la PDP.

4.2.2 Cucumber, Java y Selenium WebDriver

La combinació d'aquestes eines permet implementar els diferents passos que es seguiran per comprovar els escenaris definits prèviament amb Gherkin. Pel que fa a Cucumber [14] és l'eina que llegirà els escenaris escrits en alt nivell i en farà la validació executant els passos que hi corresponguin en cada cas. Després de l'execució, generarà un informe amb els resultats dels tests executats. Java [15] és el llenguatge de programació que s'utilitzarà per a la implementació dels passos que seguiran les proves escrites. Junament amb Java, es farà ús Selenium WebDriver [16], un framework que servirà per la definició d'aquelles proves que simulin una navegació com ho faria un usuari real. Aquest últim, fa que resulti molt senzill programar un comportament en una web real ja que utilitza comandes enviades directament a un navegador real.

4.2.3 Browserstack

BrowserStack [17] [18] és una plataforma de proves web i mòbil al núvol (també anomenada granja de dispositius) que ofereix als desenvolupadors la possibilitat de provar els seus llocs web i aplicacions mòbils a través de navegadors, sistemes operatius i dispositius mòbils reals a la carta. Serà especialment útil ja que permetrà executar les proves definides en dispositius reals repartits per tot el món, reproduint així amb gran fidelitat el comportament que podria un usuari qualsevol.

4.2.4 Jenkins

Jenkins [19] [20] és un servidor d'automatització de codi obert i gratuït. La possibilitat d'automatitzar tasques, permet aproximar-se a un procés de desenvolupament d'integració contínua desitjat per la majoria d'empreses. En l'àmbit d'aquest projecte, servirà per crear dos tipus de tasques:

- **Execucions de proves periòdiques:** aquestes tasques ens permetran que les proves definides s'executin automàticament de forma periòdica. En el nostre cas, tal com es pot apreciar a la figura adjuntada a l'inici d'aquest apartat, automatitzarem les proves definides perquè s'executin cada nit en els dos entorns de treballs que tindrem: producció i preproducció.
- **Execucions de proves parametritzades:** aquestes tasques ens permetran executar les proves definides amb una configuració concreta. Dins aquesta configuració podrem definir aspectes com l'entorn sobre el qual volem provar, el tipus de dispositiu i sistema operatiu o, per exemple, el navegador web amb el que es realitzaran les proves.

4.2.5 Eines addicionals

Les eines esmentades anteriorment definiran l'estructura principal d'aquest sistema. Tot i així, durant la implementació d'aquest es farà ús de més tecnologies i conceptes, entre les quals es troben les següents:

- **Retrofit** [21]: Client que permet realitzar crides a serveis utilitzant Java.
- **Maven** [22]: Eina que permet organitzar projectes i gestionar les dependències que aquests puguin tenir.
- **Teams** [23]: Plataforma de comunicació desenvolupada per Microsoft que ens servirà per rebre avisos sobre els resultats de les proves.
- **Mòduls:** Durant el desenvolupament del projecte, algunes de les parts es desenvoluparan de forma mòdular. Utilitzar aquesta estructura permetrà ubicar-les en llibreries, afavorint així el seu reaprofitament en altres projectes de l'empresa.

4.3 Implementació del SEPA

Amb la solució plantejada a l'apartat 4.1 i un cop escollides les eines a l'apartat 4.2, es procedeix a detallar-ne la seva implementació tècnica. Així doncs, després de refinar l'esquema oferit a l'apartat 4.1 amb la figura 8, el s'obté com a resultat la figura 9:

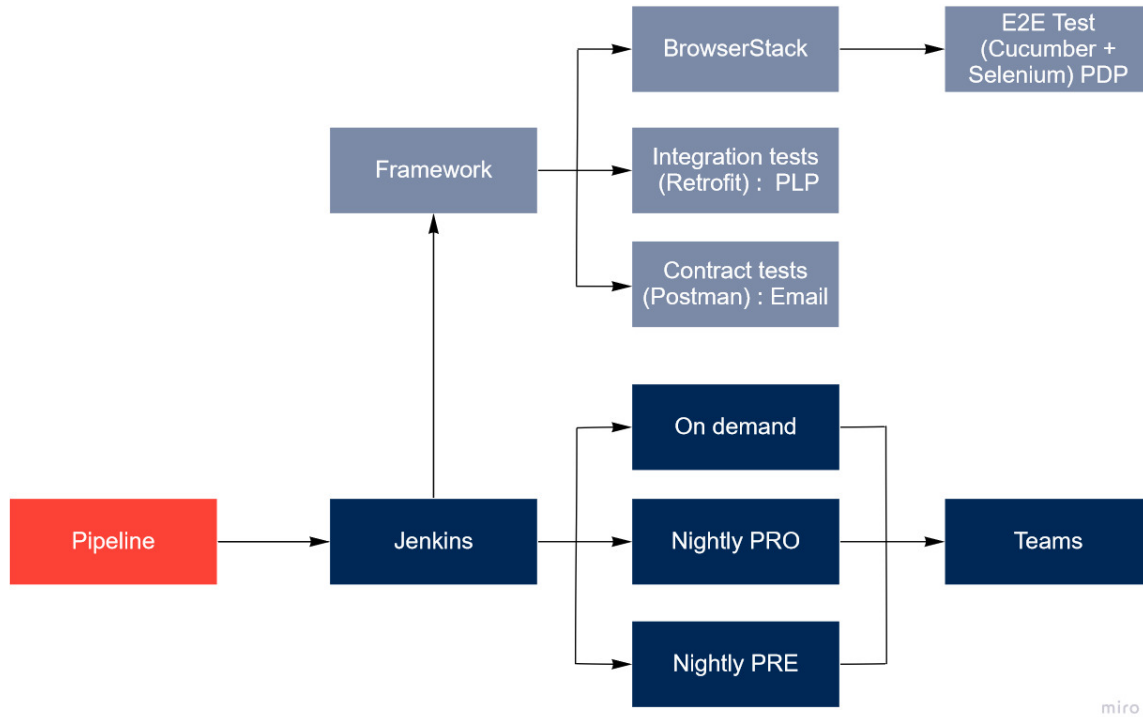


Figura 9: Diagrama representatiu de la solució plantejada

Abans d'entrar al detall, sobre el funcionament i implementació de cada component, s'ofereix una breu explicació de cada un, permetent així obtenir una visió global de com funcionarà el sistema:

- **Jenkins:** En aquest cas, el component rep el nom de la tecnologia emprada per implementar-lo. El Jenkins serà el servidor que permetrà llençar les execucions de les proves així com programar-ne la seva execució. Fent ús de tasques, es podran executar les proves desitjades executant una tasca en concret, programada prèviament.
- **Framework:** Serà la part més important, utilitzant les tecnologies Java i SeleniumWebdriver contindrà i executarà les proves per assegurar la qualitat tant la PDP, PLP com l'Email. També estarà preparat per poder allotjar i executar el codi necessari per futures proves que puguin provar nous desenvolupaments.

- **BrowserStack:** Aquest serà l'intermediari utilitzat per poder executar les proves E2E en dispositius reals.
- **Tipus de proves:**
 - **E2E Tests:** Aquestes proves executaran un navegador i simularan la navegació real com ho faria un usuari. Estaran definides amb Gherkin i Cucumber, i utilitzaran Selenium per fer ús dels navegadors. S'utilitzaran per comprovar els AC de la PDP.
 - **Integration Tests:** Aquestes proves estaran escrites en Java, i fent ús de Retrofit permetran comprovar el servei utilitzat per la PLP.
 - **Contract Tests:** Implementades sobre Postman, aquestes proves permetran comprovar que el servei d'Email compleix amb el requisits pactats.
- **Moments d'execució de les proves:**
 - **On demand:** Opció que permetrà via Jenkins executar proves amb configuració sota demanda.
 - **Nightly PRO:** Execució nocturna de proves per mitjà de Jenkins en l'entorn de producció.
 - **Nightly PRO:** Execució nocturna de proves per mitjà de Jenkins en l'entorn de preproducció.
- **Teams:** Mitjançant un connector, Jenkins ens informará dels resultats de les proves enviant missatges a un canal de Teams.

Un cop formada aquesta imatge general del sistema, es prossegueix a detallar cada un dels components. Es començarà parlant del Framework, doncs ha estat el primer component implementat i la part més important del sistema.

4.3.1 Framework

Tal com s'ha exposat en els apartats anteriors, el framework és la part més important del SEPA. És el lloc on es troben totes les proves el llançament de les quals es vol automatitzar. La idea més important d'aquest és que totes les proves, encara que utilitzen tecnologies diferent, puguin ser accessibles des del mateix punt, compartint recursos i dependències comunes si es dona el cas.

Aquesta secció, començarà amb una explicació sobre tipus de projecte que és el framework i de l'estructura bàsica que segueix. En segona instància, es parlarà del component més important d'aquesta estructura: les proves. En concret s'exposaran els 3 tipus de proves contingudes pel SEPA en aquest projecte, aquelles per assegurar la qualitat de la PDP, la PLP i l'Email.

4.3.1.1 Estructura i característiques del projecte

El framework es troba allotjat a un repositori git, facilitant així el seu accés i permetent portar un control de les versions. En aquest cas es troba ubicat al servidor de Bitbucket utilitzat per l'empresa i rep el nom de **perso-test-fw**. El nom permet identificar ràpidament l'equip (personalització - perso), la finalitat del codi (test) i el tipus de projecte (framework - fw).

Pel que fa a l'estructura de directoris i arxius del projecte, es pot observar a la figura 10. A la imatge es pot observar l'estructura obrint el projecte amb l'entorn de desenvolupament IntelliJ IDEA, que ha estat el més utilitzat per desenvolupar el framework.

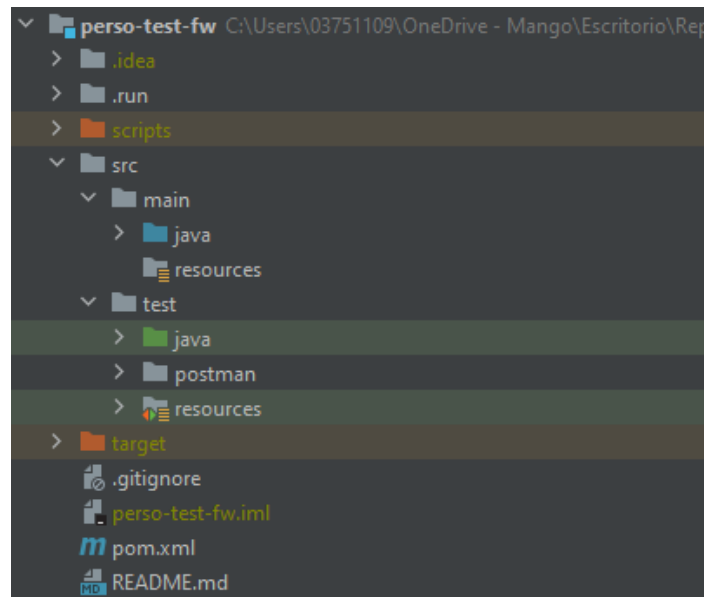


Figura 10: Estructura del framework d'automatització perso-test-fw

Observant la imatge, es poden veure múltiples directoris i arxius. Aquests s'expliquen a continuació, especificant-ne el contingut o la funcionalitat:

- **.idea, .run i perso-test-fw.iml**: aquests directoris o fitxers i el seu contingut són generats automàticament per l'entorn desenvolupament utilitzat. Contenen bàsicament informació de configuració o d'execució i no tenen afectació en el funcionament del projecte.
- **scripts**: aquest directori conté els diferents scripts que han donat suport durant el desenvolupament de les proves. Tot i que el framework és utilitzat per un servidor Jenkins, per poder llençar les proves durant la fase de desenvolupament d'aquestes, s'han generat scripts per agilitzar el procés. Bàsicament serveixen per definir les variables d'entorn utilitzades per les proves en funció del sistema i dispositius utilitzats. Tot i que no són utilitzades pel SEPA, es mantenen al framework, doncs la idea més important d'aquest és que ha de permetre incorporar noves proves en el futur.
- **src**: aquest directori conté el codi principal del projecte. En concret, conté dos directoris: **main** i **test**, que s'expliquen a continuació.
- **src/main**: aquest directori conté, separat en altres directoris en funció del llenguatge de programació, aquell codi que no està relacionat directament amb el procés de test o d'execució de les proves. En el cas d'aquest projecte, només conté fitxers d'utils programats en Java, com poden ser funcions auxiliars d'accés o assignació a les variables d'entorn.
- **src/test**: en aquest directori s'hi troba tot el codi relacionat amb la fase test o d'execució de proves. També conté els recursos necessaris per executar-les. El codi contingut s'organitza també en aquest cas, en funció del llenguatge o tecnologia emprat per implementar les proves. En el cas d'aquest projecte cal distingir entre Java (llenguatge emprat les proves a la PDP i la PLP) i Postman (tecnologia utilitzada per les proves a l'email).
- **target**: aquest directori es genera automàticament quan s'executen les proves sobre la PDP. Conté tota la informació de l'execució així com els informes de resultats generats.
- **.gitignore**: fitxer de configuració que permet definir quins fitxers s'ignoren quan es volen pujar modificacions al repositori. És un fitxer de configuració típic de git, utilitzat generalment per excloure informació pesant i redundant, com els resultats d'una execució o fitxers executables.
- **pom.xml**: fitxer Maven per a la gestió de dependències. Aquest fitxer permet tractar amb facilitat totes aquelles dependències amb recursos externs que puguin tenir les proves escrites amb Java dins el projecte. Resulta especialment útil ja

que permet tenir centralitzades en un punt totes les dependències que puguin requerir-se.

- **README.md**: fitxer informatiu. Conté tota la informació sobre les variables d'entorn utilitzades al framework així com el contingut de cada una d'elles.

4.3.1.2 PDP

Un cop explicada amb detall l'estructura del projecte, es poden ja introduir les diferents proves que s'han inclòs en aquesta primera versió del SEPA. Per començar, es detallaran les proves dissenyades per assegurar la qualitat de la personalització a la PDP. Es comença per aquestes proves ja que van ser les primeres en ser dissenyades, però també perquè són les de major complexitat.

Per entendre bé la implementació d'aquestes proves, cal entendre bé el seu funcionament i els components que hi intervenen. Cal recordar que les proves que es volen realitzar per la PDP, són proves E2E, és a dir, que simulen el comportament d'un usuari en un dispositiu i entorn reals. Més en concret, aquestes proves simulen accions i determinen si el comportament és l'esperat o no. Un exemple de prova E2E en aquest cas podria ser comprovar que estant a una PDP i fent *click* a un producte del mòdul "Et pot interessar", la web et porta la PDP d'aquest producte. Aquest tipus de comprovacions són les que s'implementaran per assegurar que el mòdul funciona correctament.

A la figura 11 es pot observar el procés seguit des de l'execució de la comanda per executar les proves fins l'execució d'aquestes en un dispositiu real. Els diferents números identifiquen els diferents passos seguits i elements involucrats en cada pas. És important esmentar que la totalitat de les classes que figuren a l'esquema han estat completament implementades per aquest projecte.

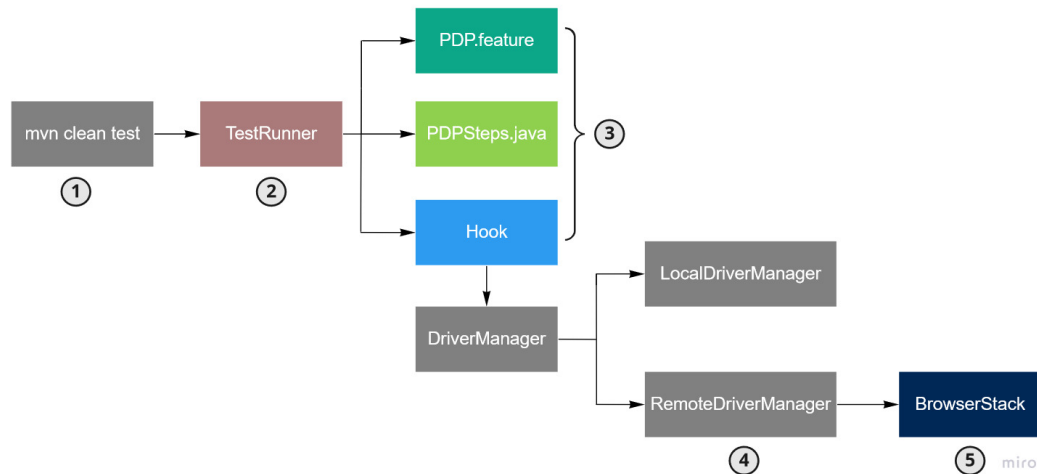


Figura 11: Procés d'execució de les proves E2E a la PDP

Tenint una visió general, s'expliquen a continuació els diferents passos i les classes i arxius:

1. Execució la comanda maven: `mvn clean test`. Aquesta comanda fa ús de l'eina Maven, i realitzar bàsicament els següents passos: neteja i cerca de tests. La neteja, que s'executa primer, és opcional, ja que l'únic que fa és netejar els resultats d'execucions prèvies si és que les hi ha. En el cas d'aquest projecte no obstant, s'executarà sempre per evitar possibles interferències entre execucions. El segon pas realitzat, el que fa la comanda test, consisteix en la cerca de fitxers que permetin executar tests unitaris. Aquests fitxers o *runners* són els encarregats de cercar i executar les proves. En aquest projecte, n'hi ha 2: el **TestRunner** i el **TestRunnerRetry**. El primer és l'encarregat d'executar les proves en primera instància, i el segon és l'encarregat de tornar a executar aquelles proves que fallin a la primera execució.
2. Cerca de proves: quan s'executen els *runners*, aquests cerquen allí on tenen indicat les *features* i els *steps* necessaris per executar-les. Una *feature* és un fitxer on es troben definits un conjunt de escenaris que es volen comprovar. Aquests escenaris representen situacions reals en les que es pot trobar un usuari. Cada escenari es divideix en 2 o 3 *steps* definits en alt nivell:
 - **Given:** *step* que defineix el punt de partida de la situació. Per exemple, trobar-se a una PDP, podria ser el punt de partida inicial d'un escenari i per tant d'una prova. Donat que un escenari no necessàriament ha de començar en un punt concret, aquesta part no sempre és necessària.
 - **When:** *step* que defineix l'acció que és vol comprovar i que, per tant, s'executarà en la simulació real. Un exemple podria ser accedir a una PDP des d'un llistat de productes.
 - **Then:** és la darrera part i defineix l'estat esperat un cop s'ha realitzat l'acció definida al *step When*. Generalment consisteix en una descripció de la situació en que es trobaria l'usuari.

Així doncs, per cada *feature* s'executaran els diferents escenaris que contingui. Per cada un d'ells, es reproduirà la situació descrita al *Given*, si en té, i s'intentarà realitzar l'acció especificada al *When*. Si aquesta és produeix correctament i es satisfà la situació descrita al *Then*, llavors l'execució de l'escenari haurà satisfactòria i per tant la prova serà correcta.

Una *feature* pot estar composta per molts escenaris diferents i el resultat d'aquests no té perquè ser el mateix. També es pot donar el cas en que diversos escenaris tinguin *steps* comuns. És per aquest motiu que es separa la definició dels escenaris de la seva implementació. Per poder fer aquesta separació s'utilitza *Cucumber* i *Gherkin*. *Gherkin* és l'eina que permet definir els escenaris utilitzant els passos esmentats en un llenguatge en alt nivell. Un cop definit un *step*, es pot

definir la seva implementació en un altre fitxer. *Cucumber* s'encarrega d'assignar a cada *step* la seva implementació fent ús del nom. D'aquesta manera si dos escenaris tenen un *step* comú només cal definir-ne un cop la seva implementació.

Havent vist el funcionament s'entén doncs per quin motiu els *runners* cerquen tant els fitxers amb les *features* (PDP.feature a la figura 11) com els fitxers amb la implementació dels *steps* que les formen (PDPSteps a la figura 11).

Per últim, els *runners* cercaran el fitxer *Hook*. Aquest fitxer serà l'encarregat de preparar l'entorn per executar les proves, i de tancar les connexions un cop aquestes hagin finalitzat.

3. Selecció de l'entorn: un cop el *runner* troba la classe Java *Hook* en crea una instància. La classe *Hook*, és l'encarregada d'iniciar el *driver* necessari per executar les proves. També s'encarrega d'aturar-lo un cop s'han finalitzat aquestes, evitant així que cap procés es quedi obert a la màquina.

El *driver* s'instancia fent ús d'una classe Java anomenada *DriverManager*. Aquesta classe bàsicament determina quin tipus de *driver* s'utilitzarà en funció d'on es volen executar les proves. El framework està preparat per poder executar proves en local, per això existeixen dues classes Java que instancien el dos tipus de *drivers*: *LocalDriverManager* per executar les proves en local, i *RemoteDriverManager* per executar les proves en remot. La decisió entre instanciar un tipus de *driver* o un altre es pren en funció de la variable d'entorn SUT_REMOTE, que indica on es volen executar les proves. En el cas de les proves a la PDP però, sempre s'executaran en un dispositiu real remot. Per aquest motiu, el *driver* sempre serà una instància de la classe *RemoteDriverManager*.

4. Connexió i preparació del dispositiu: quan s'instancia el *RemoteDriverManager* aquest retorna una estructura del tipus *RemoteWebDriver*. Aquesta estructura té unes *capabilities*, decidides en el moment en que s'instancia, que defineixen com es comportarà el *driver*. En aquestes *capabilities* el *RemoteWebDriver* defineix el nom que rebran les execucions dels tests, que aquests s'hauran de realitzar en remot a *BrowserStack* i en quin tipus de dispositiu i navegador s'hauran de realitzar. La connexió amb la granja de dispositius *BrowserStack* es realitza creant una estructura de tipus *Local* passant-li les *Keys* d'accés pertinents.
5. Execució de les proves: un cop el *Hook* ha instanciat el *driver* i iniciat la connexió, s'inicia l'execució dels *steps* que componen cada escenari. Cal esmentar que tant el fitxer PDP.feature com el fitxer PDPSteps tenen accés al *driver* i per tant a la connexió. Aquest dos fitxers comparteixen l'accés amb el *Hook* fent ús d'una classe Java anomenada *BaseUtil*, que injecta el *driver* com una dependència.

Un cop la connexió ha estat establerta i es comencen a executar les proves, es genera a la web de *BrowserStack* una sessió on es poden veure les diferents proves executades. Per cada escenari provat s'enregistra una gravació que permet a l'usuari veure com ha anat la prova. A la figura 12 es pot veure un exemple de la plana que es pot veure en accedir a *BrowserStack*.

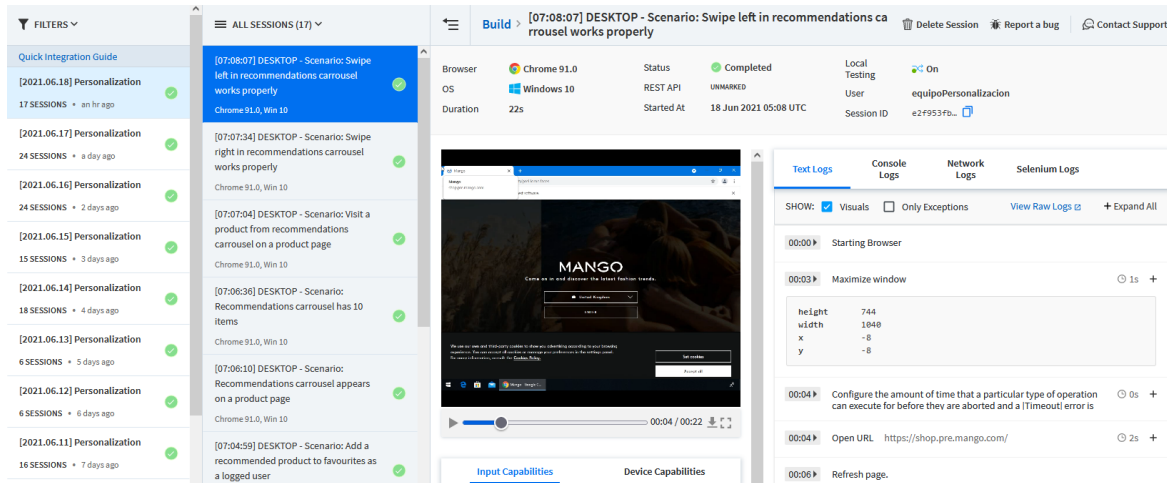


Figura 12: Plana de *BrowserStack* amb la informació de totes les execucions recents

Començant per l'esquerra, a la figura 12 es poden veure els següents elements:

- Primerament es poden veure els diferents blocs d'execucions que s'han produït durant els darrers dies. Tal com es pot observar, cada dia es crea un bloc nou. Es decideix fer-ho així per identificar ràpidament les proves quan es vulguin consultar, evitant acumular-les en un sol bloc.
- A la segona columna, es poden observar les diferents proves realitzades dins de cada bloc. Tal es pot veure al títol de cada prova, cada una d'elles es correspon amb un escenari i un dispositiu. En el cas de la figura 12 tots els escenaris han estat executats sobre *Desktop*.
- Al tercer component es pot veure reproductor de vídeo. Aquest ofereix una gravació de les proves que s'han realitzat, i permet descarregar-ne els resultats.
- Per últim, a la dreta de tot, es pot veure una secció on apareixen els *Logs* que informen sobre els passos seguits per l'execució de les proves. Aquests bàsicament informen de com s'ha produït la navegació per la web i els resultats de les comprovacions realitzades.

4.3.1.3 PLP

Les segones proves que s'han inclòs en aquest *framework* són aquelles que busquen assegurar el segon desenvolupament realitzat per l'equip de Personalització, el fet a la PLP. En aquest desenvolupament l'equip no va desenvolupar cap component nou (com si ho havien fet a la PDP amb el mòdul nou) sinó que va modificar el llistat preexistent per afegir-hi la personalització als 4 primers llocs.

És per això que aquestes proves són força més senzilles i directes que les de la PDP doncs es realitzen sobre un servei, és a dir, no són d'extrem a extrem. El seu funcionament es basa bàsicament en analitzar si els 4 primers productes que retorna el servei que proporciona els productes de la PLP són personalitzats o no.

En aquest cas, les proves estan implementades íntegrament amb Java, i es poden executar fàcilment executant la classe Java que les conté.

Abans d'explicar amb detall com estan implementades les proves i el seu funcionament, cal parlar de *mango-services*. *mango-services* és una llibreria implementada en aquest projecte que permet utilitzar serveis fent ús de *Retrofit*. Dins d'aquesta llibreria s'ha inclòs una API per cada servei que es vol utilitzar. Aquestes APIs permeten fer una crida a un servei i tenir accés a la seva resposta com si es tracés d'una estructura en Java. Per fer això ha estat necessari definir amb classes Java l'estructura que segueix la resposta que dona el servei en format Json. S'ha fet doncs el procés de *parsing* de la totalitat de la resposta. En aquest punt el treball ha estat notable, doncs els serveis emprats per aquestes proves tenen estructures molt grans i complexes. Sense fer més incís, el 2 serveis inclosos per treballar amb la PLP han estat:

- **headers**: és el servei que proporciona les capçaleres dels diferents llistats disponibles a la web. Aquestes capçaleres són necessàries per poder fer ús del servei que proporciona els productes en cada llistat. Certament aquestes capçaleres s'haguessin pogut introduir una a una a les proves, per tots els llistats on es volen executar les proves. Tot i així s'ha optat per automatitzar el procés, ja que el nombre de llistats disponibles és extens i aquests poden canviar.
- **productList**: aquest és el servei que proporciona els diferents productes que es poden veure en un llistat de la web. Per obtenir els productes d'una secció en concret, per exemple abrics de dona, es tan fàcil com realitzar una crida al servei amb els paràmetres pertinents. El més important d'aquests paràmetres es la capçalera, que indica la família (dona) i la secció (abric). Aquest paràmetre i d'altres com la família són els que s'obtenen amb el servei de *headers*.

Un cop disponibles els dos serveis a utilitzar la implementació de les proves ha estat molt més senzilla. Aquesta s'ha realitzat en una classe Java anomenada `PLP.java`, disponible dins el directori `ServiceTesting` ubicat a l'adreça `perso-test-fw/src/test/Java/`.

Per facilitar la realització de comprovacions de forma granular, s'ha implementat una funció per cada secció disponible a la web. Així doncs, s'han implementat les següents funcions: `testWoman`, `testMan`, `testTeenGirl`, `testTeenBoy`, `testGirl`, `testBoy`, `testBabyGirl`, `testBabyBoy` i `testNewborn`. A la figura 13 es pot observar l'exemple més senzill, el de `testWoman`.

```
32      @Test
33      public void testWoman() {
34          initializeTest();
35          brand = "she";
36          SubMenuEntity menuShe = menus.get(1).getMenus();
37
38          List<SubMenuEntity> submenuPrendas = menuShe.getSubMenu2();
39          testProductList(submenuPrendas, "familia");
40
41          submenuPrendas = menuShe.getSubMenu3();
42          testProductList(submenuPrendas, "accesorio");
43      }
```

Figura 13: Codi de la prova `testWoman` de `PLP.java`

Cada una d'aquestes funcions segueix el següent procediment:

1. Primerament s'inicia el test fent ús de la funció `initializeTest`. Aquesta funció realitza una crida al servei de `headers` i l'assigna a un atribut privat de la classe. Tot seguit es dona valor a l'atribut `brand`.
2. Un cop obtinguda la resposta, es cerquen i obtenen les capçaleres de la secció dins d'aquesta (es pot veure a la línia 36 de la figura 13).
3. Amb les capçaleres disponibles, es realitza una crida a la funció `testProductList` indicant també el tipus de la secció que es vol revisar. Normalment es tenen 2 seccions per cada `brand`: la secció `prendas` de tipus `familia`, i la secció `accesorios` de tipus `accesorio`. Algunes famílies tenen seccions addicionals, com pot ser `sastreteria` dins la `brand` d'home.

La funció `testProductList` és una mica extensa i per aquest motiu no s'exposa amb detall. Tot i així, el que fa és un bucle on en cada iteració comprova una família diferent, com pot ser abrics de dona. Durant cada iteració es desglossa la capçalera d'aquella família, s'obtenen tots els paràmetres necessaris i posteriorment es crida al servei `productList` utilitzant l'API de `mango-services`. Un cop

realitzada la crida, es consulta a la resposta el valor del camp *recommendationsType*. Aquest camp indica el tipus de recomanació de les 4 primeres peces del llistat en cas que siguin personalitzades. Si no ho són el seu valor és *no-personalized*. Un cop obtingut el resultat, si ha estat personalitzat, es comptabilitza la prova com satisfactòria. En cas contrari es comptabilitza com a fallada. Quan finalitza el bucle s'informa dels resultats totals de la secció, indicant el nombre de proves correctes i de proves fallades.

A la figura 14 es pot observar el resultat d'executar les proves sobre la *brand* Dona.

```
-----
Country: ES
Brand: she
prendas_she
-----
vestidos_she: no-personalized
camisas_she: no-personalized
camisetas_she: no-personalized
cardigans_she: no-personalized
sudaderas_she: no-personalized
chaquetas_she: no-personalized
abrigos_she: no-personalized
pantalones_she: no-personalized
vaqueros_she: no-personalized
faldas_she: no-personalized
shorts_she: no-personalized
bano_she: no-personalized
sport_she: no-personalized
bodies_she: no-personalized
pijamas_she: no-personalized
-----
Successful tests: 0
Failed tests: 15
-----
```

Figura 14: Resultat de les proves a la PLP sobre la secció *prendas* de Dona

Tal com es pot observar a la figura 14 tots els tests executats han resultat fallats. Això es produeix pel fet que en el moment d'implementació de les proves aquest desenvolupament encara està en fase de proves i la personalització es troba molts cops desactivada. Per aquest motiu, es decideix no automatitzar l'execució d'aquestes proves ni crear cap *job* a Jenkins que en permeti una execució ràpida. Es sap que les proves funcionen correctament ja que aquestes es limiten a analitzar les respostes dels serveis. El fet que siguin infructuoses és només qüestió del contingut de la resposta que aquests tornen. Així doncs es deixen les proves

llestes per a la seva execució en local i poder ser executades només quan sigui estrictament necessari.

4.3.1.4 Email

Per últim, s'expliquen les darreres proves implementades dins el SEPA, les encarregades de comprovar la qualitat del servei de recomanacions per *email*. Tot i que aquestes proves no estaven contemplades dins el plantejament inicial d'aquest projecte, s'ha decidit incloure-les modificant la planificació i el disseny. S'ha fet això ja que s'ha pogut finalitzar el seu desenvolupament amb temps suficient. En part, el temps ha estat suficient gràcies al fet que les proves sobre la PLP no s'han automatitzat amb el servidor Jenkins tal com estava previst.

A nivell d'estructura, aquestes proves són les més senzilles, ja que consten d'un sol fitxer executable amb una comanda. En concret, es troben al fitxer *EmailPostmanCollection.json* ubicat dins la direcció `perso-test-fw/src/test/postman/Email`.

A diferència de les proves descrites anteriorment, les proves de servei implementades per a l'Email no estan implementades amb Java. Aquestes estan escrites en un fitxer de format Json i utilitzen dues eines completament noves:

- **Postman**: per a l'escriptura de les proves i generació del fitxer .json que les conté, el qual a partir d'ara s'anomenarà col·lecció.
- **Newman**: per a l'execució de la col·lecció per mitja d'una comanda.

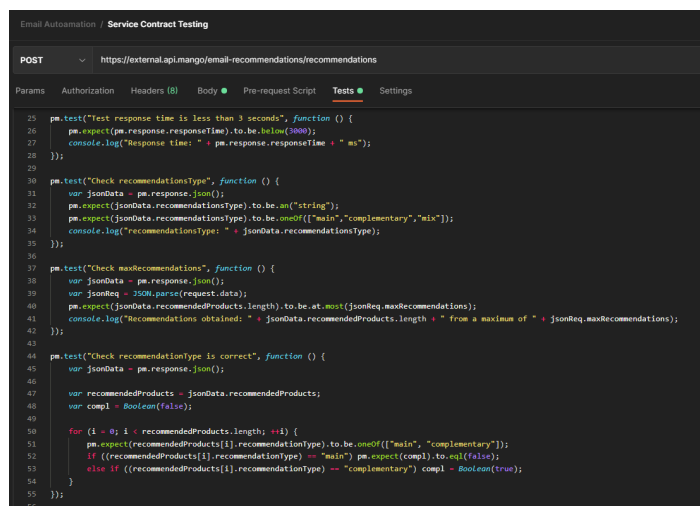
La implementació de les proves s'ha realitzat directament sobre l'aplicació d'escriptori de Postman. Aquesta aplicació permet fer crides a serveis i analitzar-ne la resposta. Addicionalment, permet definir comprovacions que s'executen sobre la resposta d'un servei cada cop que es realitza una crida. I per últim, el que és més interessant, permet exportar aquestes comprovacions en un fitxer .json. Just això doncs és el que s'ha fet per la creació d'aquestes proves.

Abans de prosseguir a definir amb més detall com s'han implementat aquestes proves és important explicar el seu objectiu i tipus. Les proves que es volen executar en aquest cas, són proves de contracte sobre un servei. En concret, en aquest cas, sobre el servei ***email-recommendations*** que proporciona les recomanacions que cal incloure als correus comercials enviats.

Les proves de contracte, són aquelles que s'encarreguen d'assegurar que un servei compleix amb tots els requisits que s'havien definit sobre el contracte pactat amb el client o consumidor. En aquest cas, l'equip de Personalització és el responsable del servei esmentat anteriorment. El que es vol doncs és assegurar que en tot moment,

s'esta oferint el servei que s'havia pactat amb l'equip que el consumeix, el d'Email.

La implementació dels test es realitza amb Javascript i és bastant intuïtiva. Dins l'aplicació de Postman, un cop introduïda la crida al servei que es vol provar, hi ha una opció que es diu *Tests*. En aquest mateix apartat es poden escriure tots els tests que es vulguin executar sobre la resposta, i s'executaran automàticament cada cop que es realitzi una crida al servei. A la figura 15 es poden observar alguns exemples dels tests inclosos dins els SEPA.



```
25 pm.test("Test response time is less than 3 seconds", function () {
26   pm.expect(pm.response.responseTime).to.be.below(3000);
27   console.log("Response time: " + pm.response.responseTime + " ms");
28 });
29
30 pm.test("Check recommendationsType", function () {
31   var jsonData = pm.response.json();
32   pm.expect(jsonData.recommendationType).to.be.oneOf(["main", "complementary", "mix"]);
33   console.log("recommendationType: " + jsonData.recommendationType);
34 });
35
36 pm.test("Check maxRecommendations", function () {
37   var jsonData = pm.response.json();
38   var jsonReq = JSON.parse(request.data);
39   pm.expect(jsonData.recommendedProducts.length).to.be.at.most(jsonReq.maxRecommendations);
40   console.log("Recommendations obtained: " + jsonData.recommendedProducts.length + " from a maximum of " + jsonReq.maxRecommendations);
41 });
42
43 pm.test("Check recommendationType is correct", function () {
44   var jsonData = pm.response.json();
45   var recommendedProducts = jsonData.recommendedProducts;
46   var compl = Boolean(false);
47
48   for (i = 0; i < recommendedProducts.length; i++) {
49     pm.expect(recommendedProducts[i].recommendationType).to.be.oneOf(["main", "complementary"]);
50     if ((recommendedProducts[i].recommendationType == "main") pm.expect(compl).to.eql(false);
51     else if ((recommendedProducts[i].recommendationType == "complementary") compl = Boolean(true);
52   }
53 });
54
55 }
```

Figura 15: Tests d'integració pel servei *email-recommendations*

Per entendre millor com funcionen les proves de contracte, s'expliquen a continuació els tests d'exemple, visibles a la figura 15:

- *Test response time is less than 3 seconds*: aquest és un dels test més bàsics, senzillament informa sobre si el servei tarda més de 3 segons en respondre o no. 3 segons és un temps inadmissible en els entorns en que es treballa, i si alguna cosa ralentitza el servei cal corregir-ho encara que la resposta sigui l'esperada.
- *Check recommendationsType*: aquest test comprova que el camp de la resposta que informa sobre el tipus de recomanacions rebudes tingui un valor correcte.
- *Check maxRecommendations*: quan es realitza una crida al servei s'especifica a la petició el nombre màxim de recomanacions, per no carregar en excés el servei. Aquest test comprova que el nombre de recomanacions rebudes no sigui superior al màxim sol·licitat.
- *Check recommendationsType is correct*: quan es reben les recomanacions, s'especifiquen els algorismes de recomanació que es volen utilitzar, demanant un algorisme principal i un de complementari. El servei intenta assolir les recomanacions

màximes amb l'algoritme principal i, si no pot assolir-lo, intenta oferir recomanacions basant-se en l'algoritme complementari. Aquest test comprova que totes les recomanacions rebudes siguin d'algun dels 2 tipus demanats. També comprova que un cop s'ha obtingut un producte recomanat amb l'algoritme complementari, no es torni a recomanar amb el principal. Si això es produís voldria dir que no s'esta prioritant correctament l'algoritme principal.

Després d'haver vist com funcionen els test i haver-ne explicat alguns exemples és interessant veure com s'executen aquests al Postman. A la figura 16 es pot veure el resultat d'executar els tests, veient que en aquest cas 1 test ha estat fallat i els altres han resultat satisfactoris.

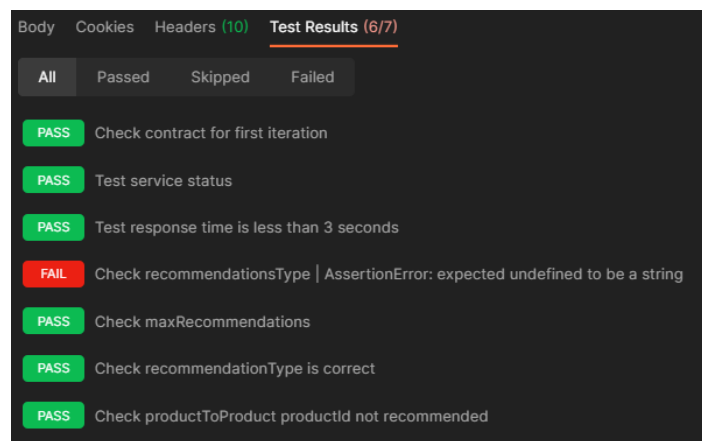


Figura 16: Resultats de l'execució dels tests pel servei *email-recommendations*

Tot i que el programari de Postman per escriptori es realment útil i permet executar els tests amb facilitat, no permet assolir els objectius que s'han definit en aquest projecte, pels següents motius:

- En primera instància crea dependència amb el programari, és a dir, requereix que tot usuari que vulgui executar els tests disposi de Postman.
- No permet automatitzar el llançament de proves fent ús de Jenkins, tal com s'ha fet amb les proves de la PDP.
- Impedeix la centralització de totes les proves del SEPA dins el *framework* creat.

Per aquests motius es decideix fer ús de *Newman* per executar els tests. *Newman* permet executar col·leccions de proves a través de comandes. D'altra banda, Postman permet exportar els tests definits com una col·lecció en format json. Així doncs, per aquest projecte, s'han exportat els tests en un fitxer anomenat *EmailPostman-Collection.json* que s'ha inclòs dins el *framework* perso-test-fw. Fent això s'ha pogut automatitzar el llançament de les proves fent ús de *Jenkins*. Aquesta part però, s'explica al següent apartat, amb l'explicació del sistema d'automatització.

4.3.2 Jenkins

Tal com s'ha explicat a l'apartat d'eines, Jenkins [17] és un servidor d'automatització de codi obert escrit en Java. En el cas de Mango, es disposen de múltiples servidors d'aquest tipus per automatitzar tot tipus de processos o tasques. En general permet automatitzar qualsevol procés informàtic que sigui executable mitjançant línia de comandes. Per automatitzar una tasca o procés, és fa ús de *jobs*. Un *job* és bàsicament una tasca que executa sempre el mateix procediment i que pot ser programada per ser executada periòdicament.



Figura 17: Logo del servidor d'automatització Jenkins

En el cas d'aquest projecte, com el que es vol executar és un projecte Maven (per la PDP) i una col·lecció amb *Newman* (per Email), ambdós executables amb una comanda, fer ús d'aquests servidors és una molt bona opció. El que es farà doncs, és crear *jobs* que executin les proves ubicades al codi del *framework* perso-test-fw.

En el cas concret d'aquest sistema, es fa ús del Jenkins utilitzat per l'equip de sistemes, disponible a la següent URL: <https://jenkins.sys.mango/jenkins/>. Aquest Jenkins, conté una secció dedicada al rol de QA 18, és aquí on s'allotjaran els *jobs* creats i exposats en aquest apartat.

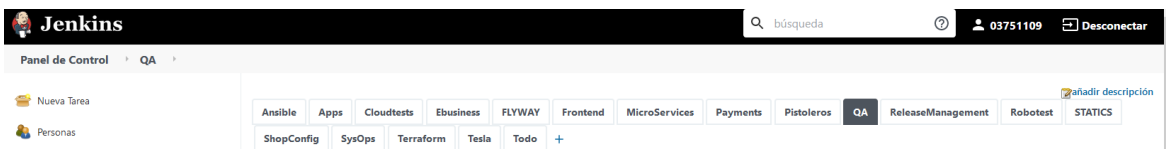


Figura 18: Selecció de la secció QA dins el servidor Jenkins

Un cop vist on s'ubiquen els *jobs* utilitzats, es procedeix a explicar-los detalladament. En total, es disposarà dels 4 *jobs* (Vegis la figura 19) següents, a l'abast de qualsevol usuari amb accés al Jenkins i amb permisos d'execució. Com es pot apreciar a la imatge, actualment no hi ha cap *job* que permeti l'execució de les proves sobre la PLP. Cal recordar que actualment no es requereix de la comprovació d'aquest producte ja que encara es troba en fase de proves.

		PERSO_EMAIL_PRO	10 días - #43: Email	N/D	10 Seg			Freestyle
		PERSO_PDP_E2E_Config	3 días 6 Hor - #339: Desktop	11 días - #336: Desktop	6 Min 52 Seg			Freestyle
		PERSO_PDP_E2E_Daily_PRE	3 días 11 Hor - #266: Desktop	2 días 14 Hor - #267: Desktop	10 Min			Freestyle
		PERSO_PDP_E2E_Daily_PRO	15 Hor - #241: Desktop	5 días 11 Hor - #235: Desktop	8 Min 22 Seg			Freestyle

Figura 19: Selecció de la secció QA dins el servidor Jenkins

Com es pot apreciar a la imatge, per cada *job* es té disponible la següent informació i opcions, d'esquerra a dreta:

- **Resultat de la darrera execució:** aquest indica si al darrera execució ha estat satisfactòria o no. Per exemple, a la imatge, només en el cas del tercer *job* va fallar la darrera execució. Una execució pot fallar per exemple si el resultat de les proves és negatiu, però també si el servidor Jenkins produeix un error. És per aquest motiu que serà important mirar el resultat de l'execució i no només si ha estat positiva o no. En conclusió, una execució positiva implica que l'execució de les proves ha estat satisfactòria però una negativa no implica necessàriament que hagi estat negativa.
- **Estabilitat de les darreres execucions:** aquest camp fa referència a les darreres 5 execucions. Indica amb icones meteorològiques com han estat les darreres execucions. Els diferents símbols representen diferents nivells d'estabilitat. Començant per la pluja (el sistema es troba molt inestable) on han fallat les darreres 5 execucions, fins al sol sense núvols, on les 5 darreres execucions han estat positives. A la imatge es poden observar 3 nivells diferents d'estabilitat.
- **Nom del *job*:** aquest camp senzillament ofereix el nom que se li ha donat al *job*. Fent-hi *click* a sobre s'accedeix a la seva pàgina.
- **Últim èxit:** aquest camp indica el temps que ha passat des de la darrera execució del *job* amb resultats positius. El valor és N/D si no s'ha produït encara cap execució amb èxit.
- **Última fallada:** aquest camp indica el temps que ha passat des de la darrera execució del *job* amb resultats negatius. El valor és N/D si no s'ha produït encara cap execució fallada.

- **Última duració:** aquest camp indica el temps va trigar la darrera execució del *job*. El valor es N/D si no s'ha produït encara cap execució del *job*.
- **Execució del *job*:** aquesta opció permet executar el *job* ràpidament. Fent-hi *click* s'executa el *job* sense necessitat d'accedir a la seva pàgina.
- **Consola:** aquesta opció permet accedir a una consola on es mostra tota la informació sobre la darrera execució del *job*.
- **Tipus:** indica el tipus de *job* si se n'hi ha assignat cap. Aquesta opció permet ordenar els diferents *jobs* en funció del seu valor.

Donat que el nombre d'opcions que ofereix Jenkins és molt elevat, s'explicarà amb detall només un dels *jobs* i el procés seguit per crear-lo. Per la resta de *jobs* s'esmentaran les seves particularitats o configuracions característiques.

- **PERSO_PDP_E2E_Config:** aquest *job* permet llençar les proves E2E definides per comprovar la qualitat del mòdul "Et pot interessar" a la PDP. En el moment de llençar l'execució ofereix diverses opcions de configuració, seguint els paràmetres introduïts al *framework*. Així doncs la plana per executar el *job* es pot veure a la figura 20.

Esta ejecución requiere parámetros adicionales:

SHOP_ENV

This var contains the URL part defining the SHOP environment.
It's value should be "https://shop.pre.mango.com/" or "https://shop.mango.com/".
Alternatively, it could define a cloudtest environment. In this case the cloudtest URL must use Akamai to access the environment.

SUT_PRODUCT_REFERENCE

DEVICE_ENV

BROWSER_ENV

This parameter only has effect in case DEVICE_ENV=Desktop.
When DEVICE_ENV=Mobile default browser will be executed.

OPTIMIZE_PREVIEW

This parameter is optional and it's for activate a Optimize preview if desired.

Ejecución

Figura 20: Plana prèvia a l'execució del *job* PERSO_PDP_E2E_Config

Les opcions que es poden apreciar a la imatge ofereixen el valor per defecte per cada camp. Veient que permet configurar cada paràmetre:

- **SHOP_ENV**: permet definir en quin entorn es volen executar les proves. Accepta tres opcions possibles: l'entorn de producció, l'entorn de preproducció i qualsevol entorn de *cloudtest*.
- **SUT_PRODUCT_REFERENCE**: permet definir sobre quina PDP es volen executar les proves. Accepta la URL (eliminant l'entorn) de qualsevol pàgina de producte visible a la web.
- **DEVICE_ENV**: permet definir sobre quin tipus de dispositiu es volen executar les proves. Les opcions disponibles són *Desktop* per la versió d'escriptori i *Mobile* per la versió mòbil.
- **BROWSER_ENV**: permet definir sobre quin navegador es volen executar les proves. Aquest valor només es té en compte quan **DEVICE_ENV** = *Desktop*. En cas que les proves s'executin sobre un dispositiu mòbil, el navegador utilitzat és el predeterminat pel dispositiu.
- **OPTIMIZE_PREVIEW**: aquest paràmetre permet executar les proves sobre una vista prèvia generada per *Google Optimize*. Aquesta funcionalitat s'ha exclòs d'aquest projecte, doncs encara no s'ha pogut comprovar amb qualitat.

Un cop vistes les opcions d'execució que ofereix el *job*, es procedeix a veure'n la seva configuració. Per fer-ho s'identifiquen els següents punts seguits en el moment de la seva creació:

- **Origen del codi font**: per que el *job* pugui executar el codi, cal proporcionar-li l'origen d'aquest, com es fa a la figura 21. En aquest cas l'origen és el repositori **perso-test-fw.git** on esta allotjat el framework. També cal proporcionar-li credencials. En aquest cas l'empresa té un usuari amb permisos, així s'evita utilitzar usuaris personals.



Figura 21: Configuració de l'origen del codi pel *job* **PERSO_PDP_E2E_Config**

- **Configuració de paràmetres:** cal definir quins paràmetres es volen poder configurar. Per fer-ho Jenkins ofereix l'opció d'afegir paràmetres amb una opció per defecte i una descripció d'aquests. Si es vol que l'usuari pugui definir ell mateix el valor del paràmetre es pot fer ús d'una paràmetre en cadena. En cas de que es vulgui fer escollir l'usuari entre múltiples opcions hi ha una opció de tipus "Elecció". És important recordar que aquests paràmetres corresponen a variables d'entorn, que el framework utilitzarà. Per tant, el nom assignat a cada variable, s'ha de correspondre amb les variables d'entorn utilitzades pel framework. Es poden configurar els paràmetres fent ús de la opció visible a la figura 22.

The image shows a Jenkins configuration page for a 'Parámetro de cadena' (String Parameter). At the top, there is a checkbox labeled 'Esta ejecución debe parametrizarse' which is checked. The form has a title 'Parámetro de cadena' and a red 'X' icon. It contains three main sections: 'Nombre' (Name) with the value 'SHOP_ENV', 'Valor por defecto' (Default Value) with the value 'https://shop.pre.mango.com/', and 'Descripción' (Description) with a text area containing: 'This var contains the URL part defining the SHOP environment. It's value should be 'https://shop.pre.mango.com/' or 'https://shop.mango.com/'. Alternatively, it could define a cloudfest environment. In this case the cloudfest URL must use Akamai to access the environment.' Below the description is a '[Plain text] Visualizar' link. At the bottom, there is a checkbox 'Trim the string' which is unchecked.

Figura 22: Exemple de configuració d'un paràmetre pel *job* PERSO_PDP_E2E_Config

- **Configuració d'entorn:** cal recordar que el framework treballa amb més variables d'entorn d'aquelles disponibles com a paràmetres per configurar. Aquestes són principalment aquelles que es corresponen amb les credencials i que, en principi, no es modifiquen entre execucions. Les variables definides són doncs: SUT_REMOTE (per executar amb BrowserStack) i els usuaris i credencials pel compte de proves, de BrowserStack i per passar el CAS (autenticació quan l'entorn estigui protegit). Es defineixen a l'opció que permet injectar variables d'entorn dins al *job*, com a la figura 23.

☒ Inject environment variables to the build process

Properties File Path

Properties Content

```
SUT_REMOTE=true

SUT_USER_MAIL=mng_test_perso_auto@mango.com
SUT_USER_PASS=$SUT_USER_PASS

BS_AUTOMATE_USER=equipopersonaliz1
BS_AUTOMATE_KEY=$BS_AUTOMATE_KEY

CAS_USER=$CAS_USER
CAS_PASS=$CAS_PASS
```

Figura 23: Configuració de les variables d'entorn pel *job* PERSO_PDP_E2E_Config

Com algunes són credencials, és important no posar-les en cap cas al descobert. Per això, s'utilitza una altra opció de Jenkins que permet definir-les en ocult. Vegis l'exemple de la figura 24.

Name/Password Pairs

Name

SUT_USER_PASS

Password

Concealed

Change Password

Name

BS_AUTOMATE_KEY

Password

Concealed

Change Password

Figura 24: Configuració de les credencials pel *job* PERSO_PDP_E2E_Config

- **Nom de l'execució:** Jenkins ofereix la possibilitat de donar un nom personalitzat a cada execució (Com a l'exemple de la figura 25). Aquesta configuració és opcional, però en aquest cas s'ha utilitzat per afegir el tipus de dispositiu al nom.

☒ Set Build Name

Build Name

#\$(BUILD_NUMBER): \$DEVICE_ENV

Avanzado...

Figura 25: Configuració del nom d'execució pel *job* PERSO_PDP_E2E_Config

Fent això es pot veure ràpidament en quin dispositiu s'ha llençat l'execució com a la figura 26.

✓ #339: Desktop	2 MB
03-jun-2021 14:57	
✓ #338: Mobile	2 MB
03-jun-2021 12:44	

Figura 26: Opcions de nom d'execució pel *job* PERSO_PDP_E2E_Config

- **Comanda d'execució:** tal com s'ha explicat a la fase d'implementació, per a l'execució de les proves sobre la PDP es fa ús de l'eina *Maven*. Per permetre al *job* executar les proves doncs, cal configurar la comanda de la figura 27 a la secció pertinent.

Ejecutar tareas 'maven' de nivel superior

Version de Maven

Maven 3.6.2

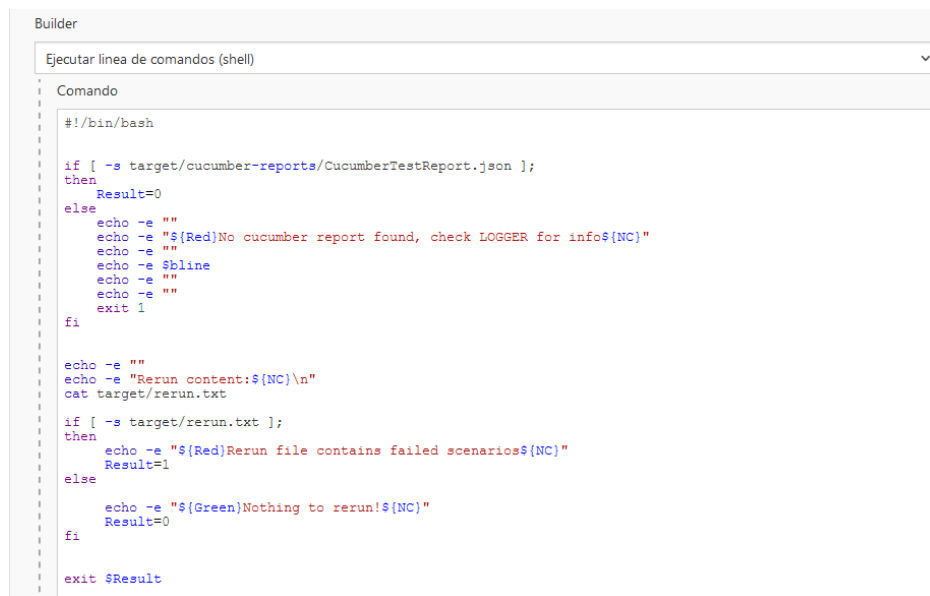
Goals

-fn clean test -Dcucumber.filter.tags="@\${DEVICE_ENV}"

Figura 27: Comanda *Maven* d'execució pel *job* PERSO_PDP_E2E_Config

Com es pot observar, la comanda inclou la variable *DEVICE_ENV* com a filtre. Aquest filtre permet definir quines proves s'executen en funció del dispositiu on es volen executar.

- **Comandes post execució:** Un cop executades les proves cal que el *job* analitzi els resultats obtinguts per saber si l'execució ha estat correcta o fallida. Amb l'opció d'executar comandes després de l'execució, el *job* realitzarà les comprovacions de la figura 28.



```
Builder
Ejecutar línea de comandos (shell)
Comando
#!/bin/bash

if [ -s target/cucumber-reports/CucumberTestReport.json ];
then
    Result=0
else
    echo -e ""
    echo -e "${Red}No cucumber report found, check LOGGER for info${NC}"
    echo -e ""
    echo -e $bline
    echo -e ""
    echo -e ""
    exit 1
fi

echo -e ""
echo -e "Rerun content:${NC}\n"
cat target/rerun.txt

if [ -s target/rerun.txt ];
then
    echo -e "${Red}Rerun file contains failed scenarios${NC}"
    Result=1
else
    echo -e "${Green}Nothing to rerun!${NC}"
    Result=0
fi

exit $Result
```

Figura 28: Comandes posteriors a l'execució del *job* PERSO_PDP_E2E_Config

Seguint l'ordre de la imatge, després de l'execució, es realitzen 2 accions:

- * En primera instancia es comprova si existeix un *Cucumber Report*. Cal recordar que si l'execució finalitza correctament, es genera un informe que n'indica com ha anat l'execució. Si l'informe hi és, s'accepta, de moment, l'execució com a correcta. Si no hi és, es finalitza l'execució del *job* i es marca com a fallida, doncs l'execució de les proves no ha estat satisfactòria.
 - * La segona comprovació que es fa, es revisar si hi ha tests que hagin fallat. Cal recordar, que com algunes proves poden ser inestables, quan es realitza una execució, les proves fallades, es guarden al fitxer *rerun.txt* i es tornen a executar. Després de la 2na execució només queden registrades al fitxer aquelles proves que han fallat 2 cops. Per fer la comprovació doncs, es mira el contingut final del fitxer *rerun.txt*. Si el fitxer es buit es finalitza l'execució i es marca aquesta com a finalitzada amb èxit. En cas contrari, contrari es declara l'execució del *job* fallida, ja que algunes proves han fallat dos cops.
- **Connexió amb Teams:** per últim, un cop el *job* ja funciona, es pot connectar amb Teams per obtenir un informe sobre els resultats d'aquest. Per fer-ho es fa ús d'una opció disponible amb l'extensió de Jenkins *Office 365 Connector*. La configuració és força senzilla, només cal indicar als camps pertinents la URL de la connexió i el nom que se li vol donar, tal com s'ha fet a la figura 29. Dins les opcions avançades es pot configurar quines alertes es volen rebre. Es pot triar per exemple, que es notifiqui sempre l'usuari, independentment del resultat o bé només quan es doni un resultat concret,

per exemple, una fallada. Aquesta opció és especialment útil ja que, en el cas dels *jobs* executats diàriament per assegurar l'estabilitat, només es vol saber-ne el resultat quan no és satisfactori.



Figura 29: Configuració de la connexió amb Teams del *job* `PERSO_PDP_E2E_Config`

Per obtenir la URL del canal on es volen rebre les alertes s'ha de fer directament des de Teams. Dins el mateix canal, fent ús de l'opció *Connectors*, es pot configurar una connexió amb Jenkins, com a l'exemple de la figura 30. Després de donar-li el nom desitjat, s'obtindrà la URL que cal posar a la configuració del *job*.

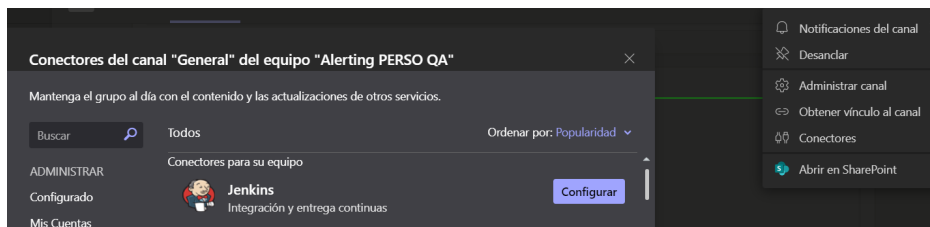


Figura 30: Configuració del connector a Teams pel *job* `PERSO_PDP_E2E_Config`

Finalment, un cop finalitzada la configuració amb Teams, si es realitza una execució del *job* s'obtindrà un avís com el de la figura 31.

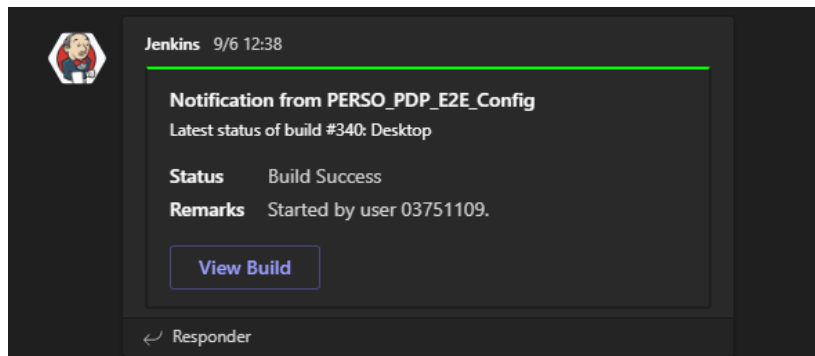


Figura 31: Notificació a Teams dels resultats de l'execució d'un *job*

- **PERSO_PDP_E2E_Daily_PRE**: aquest *job* esta pensat per assegurar l'estabilitat dels nous desenvolupaments a la PDP, executant-ne les proves periòdicament. S'executa cada matinada i informa dels resultats a través de Teams. A nivell de configuració es pràcticament idèntic al *job* PERSO_PDP_E2E_Config, amb la diferència que aquest no permet configuració. Com que el seu objectiu es comprovar que els nous desenvolupaments no afecten a l'estabilitat de la PDP, l'entorn d'execució de les proves esta fixat en preproducció. D'altra banda, el dispositiu de proves és sempre *Desktop*, ja que és on les proves són més completes, i el navegador és Chrome, ja que és el que té més trànsit. Per últim, té una configuració addicional respecte el *job* amb configuració, i són els disparadors automàtics que executen el *job* cada dia. Amb l'opció de la figura 32, es poden escollir en quins minuts, hores, dies, mesos i anys es vol executar el *job*. Per indicar que es vol executar en totes les opcions possibles s'utilitza el símbol *. Així doncs amb la programació de la figura 32, el *job* s'executarà durant qualsevol moment entre les 7 i les 8 del matí, de dilluns a divendres i durant tots el dies de l'any. L'indicador H en el cas dels minuts, indica que el Jenkins té llibertat per executar el *job* quan sigui més adient durant la hora establerta, sense definir cap minut en concret. S'escull aquesta hora ja que és quan la web té poc trànsit i es propera a l'inici de la jornada laboral, permeten així obtenir un *feedback* ràpid en començar aquesta. Pel que fa als dies, no s'executa durant el cap de setmana ja que l'entorn de preproducció no esta disponible. En tractar-se d'un entorn privat, s'apaga durant els cap de setmana.

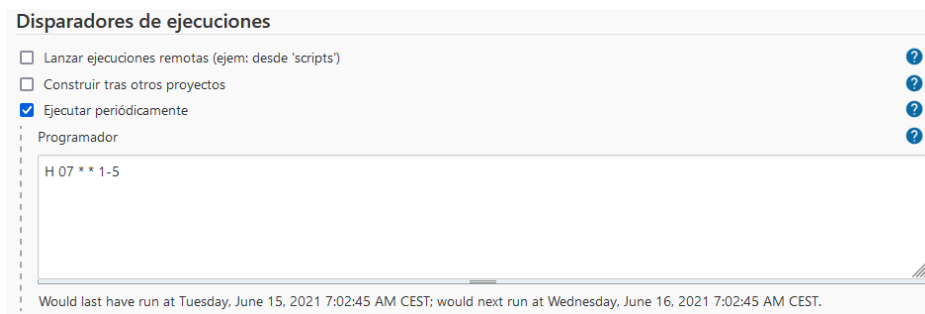


Figura 32: Configuració de les execucions diàries pel *job* PERSO_PDP_E2E_Daily_PRE

- **PERSO_PDP_E2E_Daily_PRO**: aquest *job* té com a objectiu informar sobre l'estabilitat de la PDP en l'entorn de producció. Per aquest motiu, és pràcticament idèntic al *job* anterior, amb dues petites diferències. Primerament, l'entorn d'execució és diferent, en aquest cas esta fixat a producció. D'altra banda, la seva configuració periòdica, visible a la figura 33, difereix lleugerament. El *job* en aquest cas, s'executa 1h abans, per no coincidir amb l'execució de l'anterior. Per últim, aquest si que s'executa tots els dies del any, inclosos els caps de setmana, doncs l'entorn de producció és el d'accés pels clients i sempre es troba actiu.

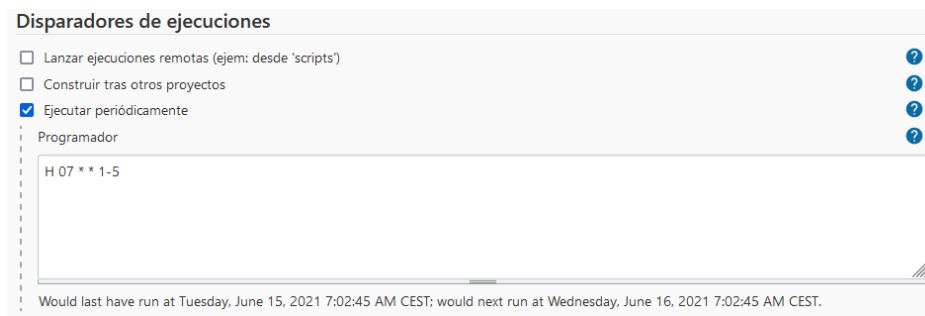


Figura 33: Configuració de les execucions diàries pel *job* PERSO_PDP_E2E_Daily_PRO

- **PERSO_EMAIL_PRO**: aquest *job* permet llençar les proves d'integració definides per comprovar el servei d'Email a l'entorn de producció. Aquest és el *job* més senzill, ja que senzillament accedeix al repositori perso-test-fw on es troba la col·lecció necessària per executar les proves i l'executa fent ús de *Newman*. Es pot veure a la figura 34 l'única configuració característica d'aquest *job*, que executa sempre la mateixa comanda.



Figura 34: Configuració del *job* PERSO_EMAIL_PRO

4.4 Solucions prèvies i alternatives

Donat que Personalització és només una de les múltiples iniciatives realitzades a l'empresa, algunes de les eines detallades en aquesta solució ja han estat utilitzades a l'empresa. Tot i això, per cada eina s'ha realitzat un estudi de les diferents possibilitats valorant els aspectes positius de cada una. Les alternatives analitzades per cada cas s'exposen a continuació.

4.4.1 Cypress vs. Selenium

Quan es parla del framework utilitzat per la definició de proves, s'ha valorat com a alternativa a Selenium la possibilitat d'utilitzar Cypress [24]. Cypress és una opció força atractiva i utilitzada a l'empresa en altres iniciatives, tot i així s'ha descartat pels següents motius [25]:

- **Llenguatge JavaScript:** Limitació del llenguatge utilitzat a JavaScript [26]. La limitació de treballar aquest llenguatge limitava la compatibilitat amb la resta d'eines utilitzades ja que impedia treballar amb Java.
- **Difícil implementació amb Cucumber:** Després de fer una recerca i basant-se en l'experiència de l'empresa, s'observa que la implementació amb Cucumber és més costosa utilitzant Cypress que utilitzant Selenium.
- **Suport oficial per aplicacions mòbils:** Tot i quedar fora l'abast definit per aquest treball, l'objectiu final d'aquest projecte és permetre assegurar també la qualitat en les aplicacions mòbils. En el moment de definició del projecte Cypress no ofereix un suport per aplicacions mòbils de forma oficial.

4.4.2 Python vs. Java

Pel que fa al llenguatge utilitzat per treballar, la proposta inicial ha estat utilitzar Python [27] ja que també és compatible amb Selenium. Tot i així, després de realitzar la recerca pertinent i dialogar amb altres membres de l'empresa es decideix utilitzar Java pels motius exposats a continuació:

- **Modularitat:** Tal com s'ha explicat en el plantejament de la solució, uns dels aspectes positius d'aquest és que ha de permetre la compartició de codi. Després d'analitzar els possibles clients amb els quals es compartiran els mòduls creats com *mango-services*, s'observa que gairebé en la seva totalitat utilitzen Java.
- **Comunitat existent:** Durant la recerca a la web, s'observa que la major part del sector utilitza Java. Aquest fet proporciona una comunitat ampla i forta i, en conseqüència, una major font de coneixement en cas de problemes durant la implementació.

- **Llenguatge base Selenium:** Tot i ser compatible amb ambdós llenguatges, el framework de Selenium WebDriver està escrit principalment amb Java.
- **Stack tecnològic i temps:** Després de veure que els temps d'execució son similars per Python i Java, s'accepta el segon ja que es l'opció per defecte que imposa l'empresa si hi ha la possibilitat.

4.4.3 Kotlin vs. Java

Tot i la desconexió personal del llenguatge, es valora també la possibilitat d'utilitzar Kotlin [28], en ser aquest més actual que Java. A priori es satisfan tots els requisits demanats i també es utilitza en altres iniciatives de l'empresa. Finalment però, es descarta l'idea en veure que Cucumber no ofereix suport oficialment per aquest llenguatge en el moment de realització del treball.

4.4.4 RestTemplate vs. Retrofit

Com a alternativa a Retrofit es planteja utilitzar RestTemplate [29]. A efectes pràctics ambdues tecnologies son similars i completament vàlides per les proves que es volen realitzar. Finalment es decideix utilitzar Retrofit en tractar-se d'una tecnologia més actual l'us de la qual esta bastant estès.

4.5 Possibles obstacles

Tot i fer un plantejament de la solució basat en un anàlisi de les diferents possibles solucions això no implica que durant el desenvolupament del projecte no puguin sorgir imprevistos o desviacions del pla inicial. En temps i entorns VUCA com els actuals, és molt important identificar els riscos ja que aquests poden afectar a la viabilitat del projecte però també a la planificació temporal i pressupost que es definiran en els següents capítols d'aquest document. És per això que s'identifiquen a continuació els principals riscos que cal tenir en compte durant el desenvolupament del projecte:

- Pel que fa a l'arquitectura de la solució s'identifica només un risc referent al servei utilitzat per a l'execució de les proves. Pel que fa a la granja de dispositius de *BrowserStack*, en ser utilitzat actualment a l'empresa, podria donar-se el cas en que sorgissin problemes de compatibilitat amb les proves definides. En aquest de que es produïssin problemes caldria valorar una modificació de les proves o l'afectació fos generalitzada, la cerca d'una alternativa al servei. No s'ha oferit cap alternativa en aquesta planificació ja que aquestes són força abundants i la presa d'aquesta decisió seria de caire econòmic i quedaria fora de l'àmbit del treball. Pel que fa a la resta de tecnologies emprades al projecte, es sap gràcies a l'experiència de l'empresa que totes les funcions requerides es poden complir sense problemes. Tot i això caldrà tenir en compte possible desviacions causades per una desconexió de les tecnologies.
- Un segon punt que pot esdevenir problemàtic durant el desenvolupament del projecte és la creació de mòduls per afavorir la compartició de codi. Cal dir que tot i ser inclòs dins l'abast del projecte aquest objectiu es opcional. És a dir, la viabilitat del projecte no depèn de la satisfacció d'aquest requisit. L'únic que busca aconseguir amb la creació de llibreries es afavorir la reutilització de codi per part d'altres membres de l'empresa. Per evitar possibles afectacions al projecte, s'intentarà assolir una solució funcional primer, i si la planificació ho permet, es modificarà aquesta primera versió ubicant el codi utilitzat en llibreries.
- Per últim, un altre punt força complicat de controlar: la configuració i parametrització. En aquest cas, es plantejarà una solució similar al punt anterior. Un cop es tinguin automatitzades les proves per la configuració predominant, s'intentarà parametritzar-les per la seva execució en qualsevol circumstància. En aquest punt el punt més important és la seva compatibilitat amb les dues versions i els dos entorns de treball. Un cop assolit aquest objectiu, es buscarà una configuració major, que permeti a l'usuari escollir coses com el navegador utilitzat per fer les proves.

5 Planificació temporal

La durada estimada d'aquest projecte és de 578 hores repartides en 126 dies, començant l'1 de febrer del 2021 i acabant el 6 de juny de 2021. En el moment de redacció d'aquest document encara no estan definides amb exactitud les dates per la seva defensa, no obstant, es posa com a límit la primera setmana de juny, fent previsió de defensar-lo en el primer torn disponible d'aquest mes.

Tenint en compte que aquest treball es realitza amb un conveni de pràctiques, es preveu una dedicació de 3 hores per cada dia laborable, representant això un 50% de les hores del conveni. Addicionalment, es planifica una dedicació de 2.5 hores diàries fora dels horaris establerts dins el conveni. D'aquesta manera, es compleix amb la planificació, tenint 88 dies laborables (264 hores en total) i 126 dies amb dedicació individual (315 hores). El resultat és doncs de 579 hores que concorda amb el nombre d'hores aportat inicialment.

L'estimació presentada es detalla a continuació, tot i així, seguint el criteri de la facultat que estima les hores de feina per crèdit entre 25 i 30 hores aquesta estimació no hauria de ser inferior a 450 hores doncs el TFG consta de 18 crèdits. Tenint en compte això, l'estimació presentada pareix raonable amb els tipus de projecte esperat.

5.1 Definició de tasques

L'organització del projecte es basarà en tasques, definides a continuació. Tot i això, abans de començar, es necessari fer-ne la distinció respecte les tasques utilitzades en la metodologia àgil de treball *scrum*. L'objectiu d'aquestes tasques és identificar els principals passos que es seguiran per un correcte desenvolupament del treball, sense entrar en detalls tècnics i fent incís en els aspectes organitzatius i de planificació com poden ser les reunions.

D'altra banda, seguint la metodologia de treball utilitzada en el projecte, es crearan tasques per a la correcta organització de la feina en un termini de temps més curt, en concret, a nivell de *sprint* (en aquest cas, una setmana). Així doncs, les tasques definides a continuació, poden correspondre's amb les tasques que es crearan durant el transcurs del treball.

Per començar, es separaran les tasques en 4 blocs:

- **Gestió del projecte:** les tasques corresponents a aquest grup corresponen a les fases de refinament i definició del projecte i, en general, de gestió i planificació d'aquest.
- **PDP:** aquestes tasques corresponen al desenvolupament del sistema de proves per la personalització a la PDP.
- **PLP:** aquestes tasques corresponen al desenvolupament del sistema de proves per la personalització a la PLP.
- **Documentació i defensa:** en última instància, aquestes tasques correspondran al procés de documentació, redacció i defensa del projecte.

S'ha decidit fer aquesta separació en blocs per separar les tasques tècniques d'aquelles de gestió i per separar els dos punts d'actuació, la PDP i la PLP. És important fer aquesta separació ja que l'estratègia tècnica que es seguirà diferirà d'un punt a l'altre, fent que un pugui tenir endarreriments sense que aquests hagin d'afectar a l'altre. D'altra banda, si es compleix amb la planificació, hi haurà moments de paral·lelització entre els dos blocs.

Començant amb el detall, pel que fa al primer bloc, el de gestió de projectes, es tenen les tasques següents:

- **B1.1 Refinament amb l'empresa:** aquesta tasca fa referència al primer contacte amb el projecte. Representa les múltiples reunions realitzades per a la definició i refinament del projecte. En aquest punt es pacta amb l'empresa el contingut de les tasques següents. Es produeixen múltiples reunions durant una setmana amb duració aproximada de 16 hores.
- **B1.2 Definició i contextualització:** aquesta tasca correspon a la definició del projecte i el context en que aquest es desenvoluparà. Seguint el criteri marcat per la FIB es dediquen 20 hores a aquesta tasca.
- **B1.3 Justificació i alternatives:** en aquesta tasca s'exposa la solució plantejada i es valoren els potencials problemes així com les solucions alternatives al problema a resoldre. Seguint el criteri marcat per la FIB es dediquen 10 hores a aquesta tasca.
- **B1.4 Abast i metodologia:** aquí es defineix amb detall quins aspectes del problema es tractaran dins del projecte. També s'explica la metodologia que seguirà i les eines utilitzades per un correcte seguiment d'aquesta. Seguint el criteri marcat per la FIB es dediquen 8 hores a aquesta tasca.

- **B1.5 Planificació temporal:** aquesta tasca es correspon amb la identificació de les tasques que corresponen aquest projecte. També s'inclou aquí la redacció d'aquest mateix document. Seguint el criteri marcat per la FIB es dediquen 16 hores a aquesta tasca.
- **B1.6 Pressupost i sostenibilitat:** aquesta tasca es correspon amb l'elaboració d'un pressupost pel projecte així com d'un informe de sostenibilitat sobre aquest. Seguint el criteri marcat per la FIB es dediquen 10 hores a aquesta tasca.
- **B1.7 Definició final del projecte:** redacció del document final referent a la planificació del projecte. Tot i estar la gestió molt treballada en aquest punt, es reserven 10 hores per una revisió exhaustiva i millora del treball.

Seguint amb el segon bloc, el de la PDP, el componen les següents tasques:

- **B2.1 Familiarització amb l'entorn i formació:** aquesta tasca fa referència al primer contacte amb l'entorn de treball de l'empresa. També s'inclou aquí la formació inicial en les tecnologies emprades durant el desenvolupament del projecte. Es dedicaran 20 hores a la formació i familiarització en l'entorn durant les dues primeres setmanes de dedicació al projecte.
- **B2.2 Definició de requisits:** corresponent al refinament i definició d'aquells requisits que haurà d'assegurar el sistema implementat. Bàsicament s'inclouran aquí totes les reunions realitzades amb la part de negoci i la resta de l'equip. També s'inclouen aquí les reunions de seguiment doncs alguns requisits poden ser canviants o no estar definits inicialment. Es preveuen unes 8 reunions d'una duració aproximada d'1 hora i mitja.
- **B2.3 Creació i estructuració del projecte:** aquesta tasca es correspon amb el disseny i implementació de l'estructura que seguirà el projecte. Per aquesta tasca es s'espera una dedicació aproximada de 30 hores repartides en dos *sprints*.
- **B2.4 Implementació de les proves:** un cop estigui el framework operatiu, s'implementaran les proves que comprovaran els requisits definits prèviament. Aquesta es probablement la tasca més llarga i difícil d'estimar, en tractar-se de les primeres proves es reserva un temps estimat de 60 hores.
- **B2.5 Parametrització del sistema:** un cop estiguin les proves llestes, es treballarà per permetre'n el seu llançament amb la configuració sol·licitada per l'usuari. Per aquesta tasca es reserven 20 hores.
- **B2.6 Automatització:** un cop estigui el sistema operatiu pel seu ús en local, se n'automatitzara l'ús utilitzant Jenkins. Donades les múltiples opcions que s'espera oferir es dedicaran 50 hores a aquesta tasca.

- **B2.7 Extracció i creació de pàgines:** un cop el sistema sigui funcional, se n'extreuran alguns components en forma de pàgines per afavorir-ne la reutilització. Aquesta tasca es relativament curta, amb una duració aproximada de 14 hores.

Pel que fa al tercer bloc, el de la PLP, es distingeixen les tasques següents:

- **B3.1 Definició de requisits:** corresponent al refinament i definició d'aquells requisits que haurà d'assegurar el sistema implementat. Bàsicament s'inclouran aquí totes les reunions realitzades amb la part de negoci i la resta de l'equip. Un cop més, s'inclouen aquí aquelles reunions de seguiment doncs alguns requisits poden ser canviants o no estar definits inicialment. Els requisits en aquest cas son força nombrosos per això es reserven 22 hores per aquesta tasca.
- **B3.2 Creació mòduls d'APIs sobre serveis :** com l'estratègia que es seguirà en aquest bloc es basarà en l'ús de serveis; s'implementaran utilitzant mòduls totes les APIs necessàries per a la realització de les proves. Un cop més, el fet d'utilitzar aquesta estructura, n'afavorirà la seva reutilització. Representa el punt més important del bloc, amb una dedicació d'unes 60 hores, subjectes a imprevistos.
- **B3.3 Implementació de les proves:** com el framework ja serà funcional del bloc anterior, un cop estiguin les APIs operatives s'implementaran directament les proves que comprovaran els requisits definits prèviament. Si la B3.2 es realitza correctament, la durada d'aquesta tasca no serà superior a 40 hores.
- **B3.4 Parametrització del sistema:** un cop més, quan estiguin les proves llestes, es treballarà per permetre'n el seu llançament amb la configuració sol·licitada per l'usuari. Subjecte a modificacions, es reserven inicialment 20 hores a aquesta tasca.
- **B3.5 Automatització:** un cop estigui el sistema operatiu pel seu ús en local, se n'automatitzarà l'ús utilitzant Jenkins. Donades les múltiples opcions que s'espera oferir es dedicaran 40 hores a aquesta tasca.

Per últim, el darrer bloc, referent a la documentació i defensa del projecte es compondrà de:

- **B4.1 Documentació del projecte:** aquesta tasca inclourà tot el procés de creació del document resultant de la realització del projecte.
- **B4.2 Preparació de la defensa:** per últim, es treballarà en l'exposició i l'estratègia que seguirà per la defensa del treball davant el tribunal corresponent. S'inclouen aquí el conjunt de reunions que es realitzaran amb el director i ponent del treball.

5.2 Plans alternatius i obstacles

Pel que fa a la previsió d'obstacles no es valora la possibilitat de canviar les tasques. Com les tasques generades no tenen un caràcter tècnic, es poden produir imprevistos que afectin a la planificació temporal, però no es requeriria de tasques alternatives. Tot i això però, si que s'han caracteritzat algunes tasques com a optatives, aquests casos especials, s'expliquen a continuació:

- **B2.7:** aquesta tasca fa referència a una millora en benefici de la compartició de codi. Ja que es tracta d'una iniciativa individual, sense gaires precedents a l'empresa és una tasca que pot generar problemes. Si es donés el cas que la seva implementació fos inviable es podria prescindir d'ella, doncs no afecta estrictament al resultat del projecte.
- **B3.2:** aquesta tasca guarda força similitud amb la tasca 2.7, amb la diferència però que en aquest cas no es tracta d'una millora sinó d'obtenir directament una solució intel·ligent. Si el fet voler aportar modularitat suposés un problema, s'optaria per fer ús dels serveis directament al projecte principal. La tasca es realitzaria igual, amb un cost temporal inferior doncs no es generarien llibreries compartides, però propiciant la duplicitat de codi.
- **B2.5 i B3.4:** aquestes tasques també poden ser objecte de problemes, doncs involucren molts aspectes de l'entorn de treball. Si bé la seva idea és només aportar polivalència a les proves, en el moment de planificació no es sap quins aspectes seran parametritzables, es per això, que la tasca tant podria allargar-se tant si s'obtinguessin els resultats esperats com no. Un cop realitzada la B2.5 es valorarà si l'estimació realitzada per la B3.4 es adequada o cal modificar-la.

Tot i això, aquesta planificació s'ha realitzat tenint en compte un marge de 15 dies en relació a la part tècnica, doncs és on es poden produir els problemes esmentats.

5.3 Estimacions i Gantt

A la taula 1 es poden apreciar resumides totes las tasques identificades així com l'estimació temporal per cada una d'elles:

Tasca	Codi	Hores	Dependències
Gestió del projecte	B1	90	
Refinament amb l'empresa	B1.1	16	
Definició i contextualització	B1.2	20	B1.1
Justificació i alternatives	B1.3	10	B1.1
Abast i metodologia	B1.4	8	B1.1
Planificació temporal	B1.5	16	B1.1
Pressupost i sostenibilitat	B1.6	10	B1.1
Definició final del projecte	B1.7	10	B1.1, B1.2, B1.3, B1.4, B1.5, B1.6
PDP	B2	206	
Formació i familiarització amb l'entorn	B2.1	20	
Definició de requisits	B2.2	12	
Creació i estructuració del projecte	B2.3	30	B2.1
Implementació de les proves	B2.4	60	B2.2, B2.3
Parametrització del sistema	B2.5	20	B2.4
Automatització	B2.6	50	B2.4
Extracció i creació de pàgines	B2.7	14	B2.4
PLP	B3	182	
Definició de requisits	B3.1	22	
Creació de mòduls d'APIs sobre serveis	B3.2	60	B3.1
Implementació de les proves	B3.3	40	B2.3, B3.2
Parametrització del sistema	B3.4	20	B3.3
Automatització	B3.5	40	B3.3
Documentació i defensa	B4	100	
Documentació del projecte	B4.1	80	B1.7
Preparació de la defensa	B4.2	20	B1.7
TOTAL		578	

Taula 1: Taula amb les tasques planificades i les seves estimacions

A les figures 35 i 36 es pot veure Gantt resultant de la planificació realitzada.

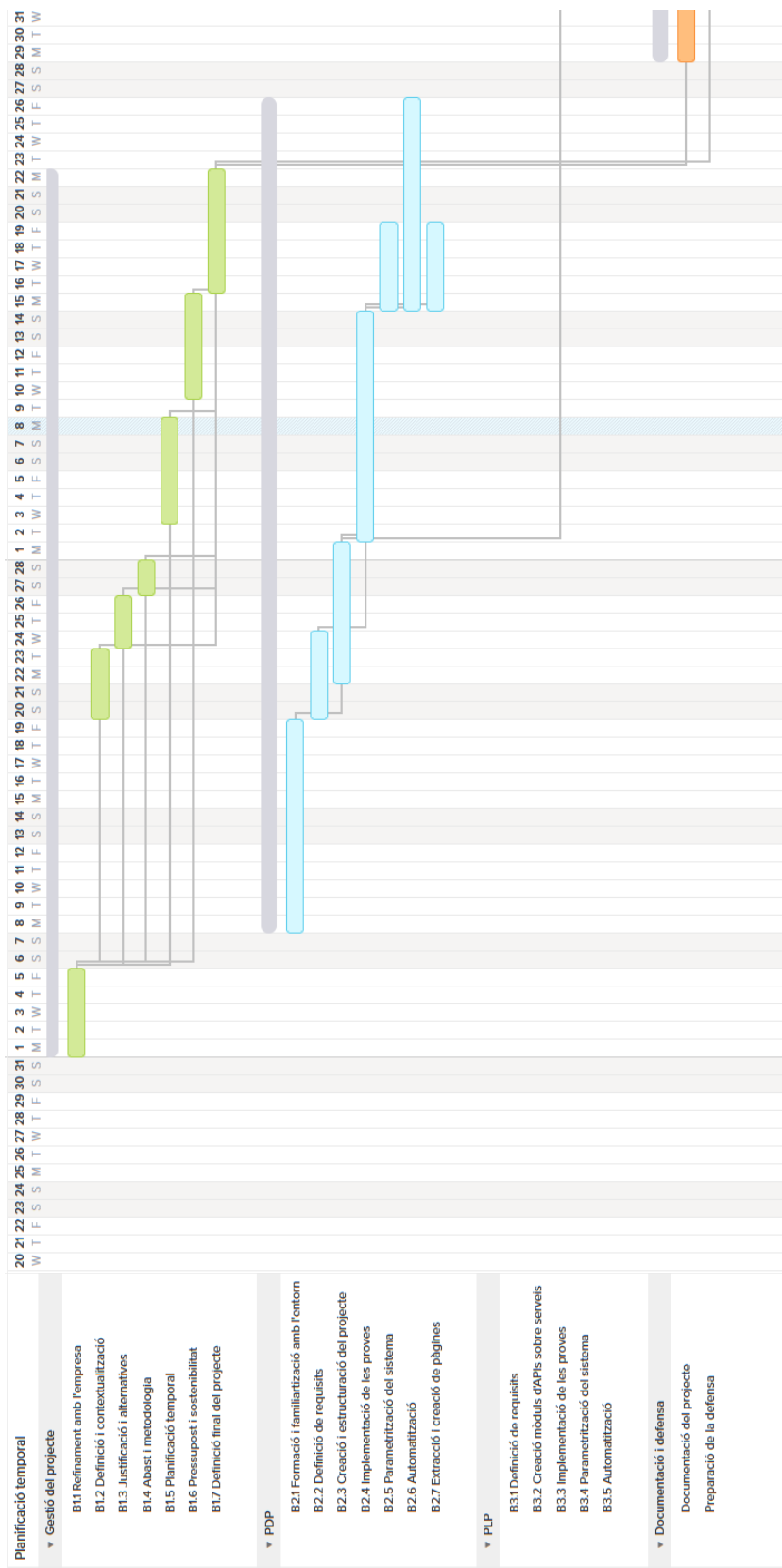


Figura 35: Primera part del diagrama de Gantt

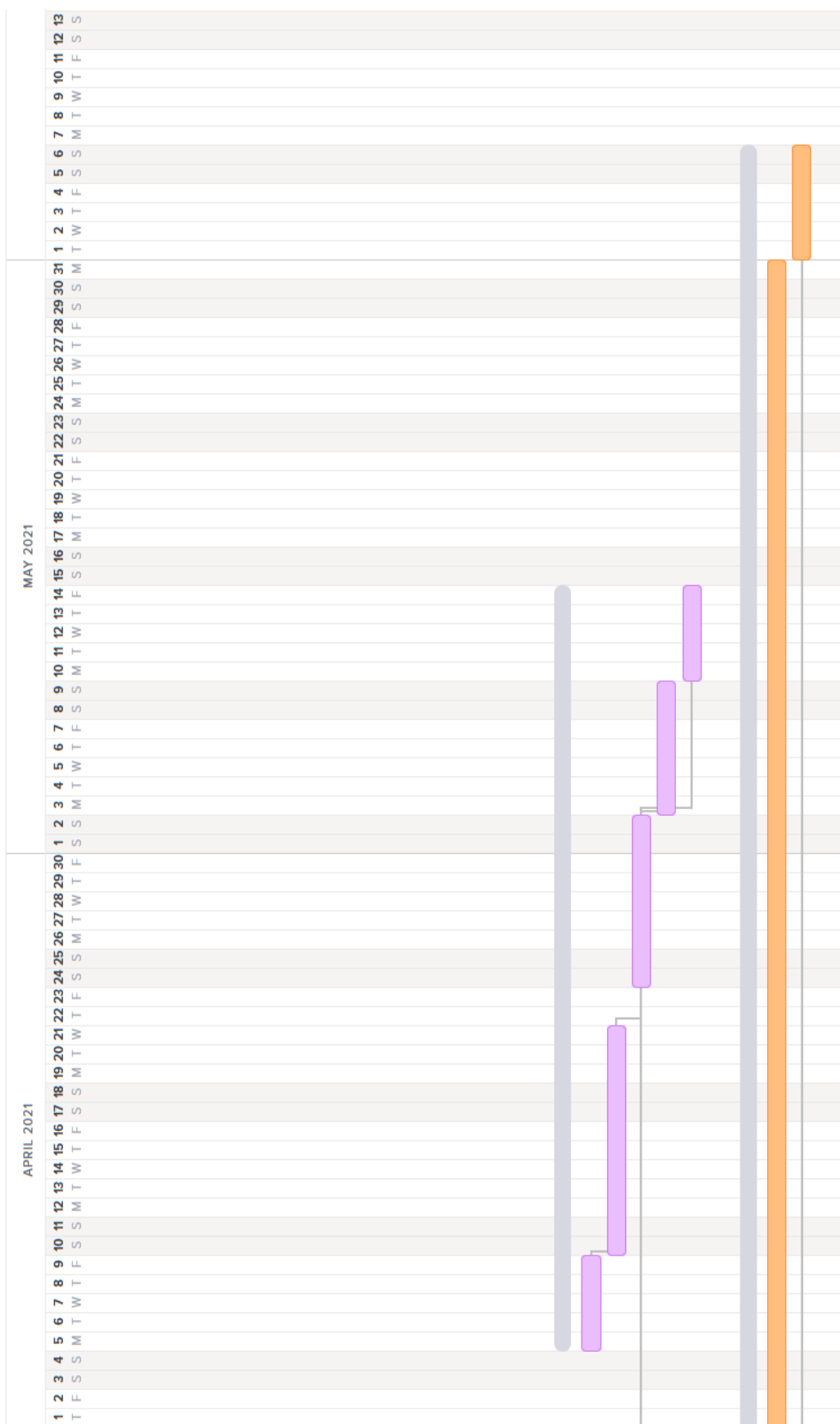


Figura 36: Segona part del diagrama de Gantt

6 Pressupost

6.1 Identificació del costos

Amb la planificació temporal feta ja es poden calcular els costos associats al projecte. Començant amb els costos econòmics, per calcular-los cal distingir entre les següents categories de costos: costos de personal, eines (programari) i dispositius utilitzats. En aquest cas en concret, no es fa incís en els costos de desplaçament o d'espai de treball, doncs en realitzar-se en una empresa tant gran l'impacte es mínim i força complicat d'estimar. El cost real de l'espai es limitaria al consum d'un ordinador, doncs la resta de costos són independents del projecte. Pel que fa al transport, no suposa cap tipus d'increment a l'alumne.

Per fer una correcta justificació del pressupost plantejat, s'exposen primer els costos de personal ja que són els més rellevants. En segona instància, s'ofereix un detall de la resta de categories i finalment una estimació total fent ús del preus i tarifes obtinguts.

6.1.1 Costos de personal

Per calcular els costos de personal, cal identificar els rols implicats en el projecte així com el seu preu per hora i els costos addicionals en impostos si n'hi ha. En el aquest cas, hi ha 2 rols presents en el projecte, el de QA Manager (que fa també la funció de Product Manager) i el de l'estudiant com a QA Stager. Els preus mostrats a la taula 2 inclouen tots els costos en impostos derivats de la contractació del rol.

Rol	Cost (€ / h)
QA Manager	22.5
QA Stager	9

Taula 2: Taula amb els rols del projecte i el seu preu de facturació per hora

Pel que fa al cost del QA Stager, al tractar-se d'un becari en contracte de pràctiques cal afegir costos per l'empresa exposats a la taula 3.

Motiu	Cost
Gestió UPC	452
Aportació S.S Empresa	48.41

Taula 3: Taula amb les despeses addicionals derivades del conveni de pràctiques

6.1.2 Costos genèrics

Tal com s'ha indicat al inici d'aquest apart estan també costos associats als recursos utilitzats durant el desenvolupament del producte.

Pel que fa al programari utilitzat l'únic recurs emprat que té un cost calculable és l'eina BrowserStack. La factura per aquesta eina és de 13776 USD anuals, que repartits en 8 usuaris i en 12 mesos (tenint en compte que el projecte només dura 4 mesos) suposa un cost de 430.5 USD o 361€ (1€ = 1.19 USD en el moment d'escriptura d'aquest document). Pel que fa a la resta d'eines emprades en el desenvolupament, la creació del projecte no suposa un cost addicional per l'empresa. Les eines emprades per la implementació del framework són gratuïtes i el servidor Jenkins emprat per l'automatització és compartit amb múltiples equips de l'empresa. Pel que fa al programari de seguiment i control de versions també es fa ús de llicències empresarials amb un gran nombre d'usuaris.

D'altra banda hi ha els costos relatius als bens materials utilitzats en el desenvolupament del projecte. En aquest punt només cal calcular l'amortització per l'ordinador portàtil aportat per l'empresa. Es tracta d'un Lenovo ThinkPad L390 - 20NR001ESP amb un cost de 1.139 € i un temps de vida útil de 5 anys. Així doncs el seu cost d'amortització resultant és de 75.9 € per aquest projecte.

A la taula 4 s'ofereix una visió resumida dels 2 costos identificats en aquest apartat.

Eina	Cost / Amortització
BrowserStack license	461
Lenovo ThinkPad L390 - 20NR001ESP	75.9

Taula 4: Taula amb els costos genèrics identificats al projecte

Resulta rellevant esmentar que no es comptabilitzen costos derivats de l'entorn de treball com podrien ser la connexió o electricitat ja que el treball es realitza principalment a les oficines de l'empresa. Al final d'aquest apartat es fa una justificació més extensa de les decisions preses respecte aquest punt.

6.2 Estimació del costos

Un cop identificats tots els costos del projecte, ja es pot elaborar un pressupost total tenint en compte els temps per aquells costos per hora o mensuals. Així doncs, es resumeix a taula 5 el total de costos on les unitats fan referència al nombre de mensualitats pels costos addicionals i a les hores i dies pels rols:

Cost	Unitats	Preu	Total
QA Manager	30	22.5	675
QA Stager	3 x 88	9	2376
Gestió UPC	1	452	452
Aportació S.S Empresa	4	48.41	193.64
BrowserStack license	1	461	461
Lenovo ThinkPad L390 - 20NR001ESP	1	75.9	75.9
TOTAL:			4233.54

Taula 5: Taula amb totes les despeses identificades i el cost total d'aquestes

En el següents punts s'aclareixen els possibles dubtes resultants de la taula:

- S'ha definit en 30 hores el nombre d'hores facturades del rol de QA Manager basant-se en una dedicació de 12 hores de formació inicial i 1 hora de seguiment setmanal.
- En tractar-se d'un pressupost per l'empresa només es facturen aquelles hores de QA Stager dins el conveni de cooperació dedicades estrictament a la creació del projecte. Basant-se en aquesta premissa tampoc s'han compatibilitzat els costos derivats de les hores de treball fora del conveni laboral. En el moment de realització d'aquest document i en plena pandèmia del COVID-19, encara no hi ha una regulació que requereixi d'una compensació per l'amortització de l'entorn de treball dins el marc del teletreball. Així doncs tractant-se d'un pressupost dedicat a l'empresa i produït dins el conveni de cooperació esmentat no es poden compatibilitzar costos relatius a aquest aspecte. Tot i així, si es volgués fer un pressupost detallat i no dedicat a l'empresa, caldria afegir en concepte de costos addicionals les següents despeses:
 - 80€ relatius al cost d'una T-Jove per justificar el desplaçament al lloc de treball.
 - 4 mensualitats de 30€ relatius a la connexió a Internet requerida per l'estudiant.
 - Addicionalment caldria calcular els costos d'electricitat i amortització corresponents al domicili de l'estudiant.
 - També caldria incloure els costos esmentats dins l'àmbit de l'empresa. En aquest cas resulten més complicats d'estimar, doncs en tractar-se d'unes ofícines centrals el nombre d'usuaris és molt gran (al voltant de 600 persones).

- L'aportació a la Seguretat Social es factura mensualment a part dels costos del rol.
- El cost de Gestió UPC fa referència a l'aportació que realitza l'empresa a la UPC amb motiu del conveni de cooperació. En aquest cas s'ha tret el percentatge corresponent tenint en compte que el conveni té un duració total de 8 mesos.

6.3 Control de gestió

Pel que fa al control de gestió cal tenir en compte que els següents punts:

- La viabilitat del projecte s'ha estudiat amb detall, així doncs no es plantegen costos addicionals associats al programari utilitzat. S'afirma això ja que no es planteja una alternativa pel que fa a l'arquitectura d'aquelles tecnologies de pagament utilitzades. Si que és cert que es poden presentar desviacions amb tecnologies com el Retrofit o Cucumber, però en ser les alternatives a aquestes també gratuïtes no es produiria un increment dels costos en programari. L'única eina que suposa un cost es BrowserStack i ja s'utilitza actualment a l'empresa assegurant així la viabilitat del seu ús pel projecte.
- Tot i no tenir implicacions en costos per recursos, qualsevol desviació pot resultar en un increment de les hores dedicades per qualsevol dels rols assignats. Aquestes desviacions estan plantejades i senzillament es corregirien amb una major dedicació temporal dins el conveni de cooperació.
- Per últim, sempre existeix la possibilitat de fallada de l'ordinador emprat. En aquest cas seria inevitable que l'empresa es fes càrrec de la substitució del dispositiu.

Havent vist aquests punts doncs, en cas de desviació només caldria recalcular els costos per rol tenint en compte les hores de dedicació reals. La desviació respecte el pressupost original seria doncs el resultat de realitzar la següent operació:

$$(Hores\ dedicació\ reals - Hores\ estimades) \times Preu\ per\ hora\ rol$$

7 Sostenibilitat

Per poder fer un anàlisi correcte sobre l'impacte i sostenibilitat del projecte es realitza l'enquesta anònima del projecte EDINSOST aportada per la FIB. Posteriorment es realitza un anàlisi de l'impacte en les dimensions econòmica, ambiental i social responnent a les preguntes proposades.

7.1 Autoavaluació

Després de la realització de l'enquesta m'he adonat d'alguns punts rellevants. En primera instància, crec que, he confirmat la manca de coneixement que tinc per valorar l'impacte mediambiental d'un projecte. També m'he adonat de la necessitat de millorar alguns aspectes per poder valorar millorar l'impacte econòmic. Tot i així, en el moment de realització d'aquest treball estic començant a cursar l'assignatura *Aspectes Socials i Mediambiental de la Informàtica* fet que esta contribuint a suplir aquesta deficiència de coneixements.

7.2 Dimensió econòmica

- **Respecte el Projecte posat en producció (PPP): Reflexió sobre el cost que has estimat per la realització del projecte?** Crec que el cost presupostat es força realista. El fet d'estar actualment treballant en el projecte i fer-ho en una empresa amb coneixements previs contribuirà molt a obtenir una estimació acurada.
- **Respecte la Vida Útil: Com es resolen actualment els aspectes de costos del problema que vols abordar (estat de l'art)?** En l'actualitat els aspectes abordats es resolen manualment. El fet fer-ho automàticament suposarà una disminució molt gran en el temps de dedicació dels professionals.
- **En què millorarà econòmicament (costos...) la teva solució respecte a les existents?** Tal com s'esmenta representarà una reducció notable de la dedicació dels tècnics. Aquesta reducció es traduirà directament en benefici econòmic.

7.3 Dimensió ambiental

- **Respecte el Projecte posat en producció (PPP): ¿Has estimat l'impacte ambiental que tindrà la realització del projecte?** Resulta complicat fer una estimació detallada. El fet de que el projecte es produeixi en una empresa tant gran fa que valorar-lo individualment resulti realment difícil.

- **Respecte el Projecte posat en producció (PPP): ¿T'has plantejat minimitzar-ne el impacte, per exemple, reutilitzant recursos?** La majoria de tecnologies emprades ja estan presents a l'empresa, així doncs s'aprofiten la majoria de productes existents.
- **Respecte la Vida Útil: Com es resol actualment el problema que vols abordar (estat de l'art)?, i. ¿En què millorarà ambientalment la teva solució respecte a les existents?** Estrictament parlant millorarà la dedicació en fer les tasques esmentades. Per buscar algun aspecte positiu, el fet de requerir menys atenció humana acaba esdevenint en una disminució dels transports a l'empresa. Aquesta disminució generalment implica una disminució en el consum energètic i el nombre d'emissions.

7.4 Dimensió social

- **Respecte el Projecte posat en producció (PPP): Què creus que t'aportarà a nivell personal la realització d'aquest projecte?** Personalment el projecte m'està introduint en un sector que desconeixia i que m'ha sorprès molt positivament.
- **Respecte la Vida Útil: Com es resol actualment el problema que vols abordar (estat de l'art)? i. ¿En què millorarà socialment (qualitat de vida) la teva solució respecte les existents?** Bàsicament en el fet d'evitar la realització d'una feina repetitiva i monòtona. Evitar aquestes tasques em permetrà invertir millor el temps en feines més productives per mi i per l'empresa.
- **Respecte la Vida Útil: Existeix una necessitat real del projecte?** Realment es podrien seguir realitzar les tasques sense el desenvolupament del projecte. Tot i així, vist els beneficis que reportarà seria una mala decisió no portar-lo a terme.

8 Conclusions

Un cop cobert el temps destinat a aquest projecte, ha arribat el moment de recapitular i veure els resultats i el grau de satisfacció respecte la planificació inicial. Per començar, es fa un breu resum dels passos seguits durant la realització del projecte i d'aquestes memòries.

Per començar va ser necessari identificar i definir el problema. Mentre es feia això calia anar documentant el primer esborrany d'aquestes memòries, el relatiu al GEP. Per poder plasmar bé en quin context es produïa el treball va ser important detallar el rol de QA i les carències o punts de millora en el treball d'aquest.

Després d'haver vist que la potencial millora es trobava en el procés de validació del QA, es van definir els objectius a assolir per millorar aquest procés. També es van definir alguns objectius que modelarien la solució, com la facilitat d'execució de la solució o facilitat de compartició de codi.

Un cop acabada la contextualització va detallar-se el punt d'acció concret on calia aplicar la solució. Havent desgranat com funcionava el procés de desenvolupament de l'empresa i on hauria d'actuar-hi la solució, es va plantejar com a solució la implementació del SEPA (Sistema d'Execució de Proves Automàtiques).

L'explicació del SEPA es pot qualificar de progressiva dins d'aquest document. Començant per un plantejament d'arquitectura en alt nivell, se li van sumar les tecnologies a emprar obtenint així una arquitectura de components més tècnica i detallada. El següent pas va consistir en la implementació dels components principals: el *framework* d'execució de proves i el sistema encarregat de la seva automatització. Aquesta part es veu reflectida en aquest document amb una explicació de la implementació i funcionament del SEPA.

Després d'aquest resum sobre el procés seguit pel projecte, ha arribat el moment de fer una valoració dels resultats. Fent referència als objectius estipulats en aquest document, es pot dir que han estat coberts amb èxit. S'ha aconseguit implementar un sistema que permet executar proves amb un sol *click* per comprovar l'estat d'un producte. Queden coberts doncs aquells objectius referents a l'automatització del procés de validació i també aquells referents a l'execució fàcil de proves. També s'ha aconseguit automatitzar el llançament de proves de forma periòdica, aconseguint així monitoritzar l'estabilitat dels desenvolupaments. Fins i tot ha estat possible incloure proves per l'Email, desenvolupament que havia quedat fora de la planificació inicial d'aquest projecte. Si bé és cert que part ha estat gràcies a que les proves sobre la PLP no han requerit de ser automatitzades, el resultat ha estat molt encoratjador. Així doncs es pot concloure que tècnicament i en l'àmbit de l'empresa el projecte ha estat un èxit.

A nivell acadèmic el resultat ha estat aquest propi document. En aquest aspecte, és important dir que en alguns punts han sorgit dificultats. Sobretot pel que fa a la forma d'abordar els diferents apartats, molts cops ha estat complicat decidir fins a quin punt calia aprofundir. Durant les múltiples revisions fetes, s'han anat clarificant conceptes que podien no quedar clars, però en tot cas la quantitat d'informació ha estat aclaparadora. Per entendre bé el projecte, calia definir bé el context en que es produïa així com tots els conceptes que implicava. El rol de QA, el cicle de l'empresa, la metodologia de validació, el llarg conjunt de tecnologies i metodologies seguides per la empresa, etcètera. Aquest fet ha produït que en alguns aspectes de la implementació el nivell de detall no hagi estat molt extens. S'ha explicat per exemple, l'organització i funcionament dels tests per la PDP a nivell de classe, el comportament dels serveis i funcions per la PLP o alguns exemples de comprovacions pel cas d'Email. En el cas de l'automatització s'ha utilitzat un procés com a referència per explicar-ne la resta. El que es vol concloure amb això és que, en alguns punts ha estat difícil decidir quina informació incloure i quina no, per evitar sobrecarregar el document i, sobretot al lector.

Per acabar, és interessant comentar que en l'actualitat es segueix treballant en el SEPA. Fora de l'abast d'aquest projecte, s'han extret alguns dels components de les proves de la PDP en forma de llibreria per afavorir la compartició de codi. Seguint l'exemple de *mango-services* s'ha creat *mango-pages* per facilitar la creació de proves d'extrem a extrem. D'altra banda es manté pendent l'automatització de les proves per la PLP, tan bon punt aquestes puguin ser utilitzades amb regularitat. Per últim, s'ha començat a treballar amb proves per un nou desenvolupament de l'equip de Personalització ubicat a la *Bossa*. Encara no està decidit quins tipus de proves s'inclouran al *framework*, però la idea inicial és començar amb proves com les aplicades per l'Email, vista la facilitat d'implementació i el ràpid *feedback* que ofereixen.

Ja concloent aquest apartat i en conseqüència aquestes memòries, vull fer unes petites conclusions a títol personal. He de dir que en un principi no estava molt segur amb aquest treball, ja que el fet de no ser estrictament de la meva especialitat em generava dubtes sobre la meua capacitat. Ara puc dir però, que no podia estar més equivocat. Si bé és cert que gairebé tots els coneixements aplicats en aquest treball els he hagut d'aprendre durant les pràctiques, crec que quasi qualsevol estudiant s'hagués vist amb una situació similar. Tampoc dubto però, que se n'hagués sortit. El que vull dir amb això, és que aquest treball ha servit per reafirmar la següent idea: com a enginyers i enginyeres som formats amb coneixements tècnics però sobretot som formats amb capacitats. Amb capacitat d'analitzar, d'investigar, de proposar i de solucionar.

9 Agraïments

Vull començar aquestes paraules amb un especial agraïment al Fran, el meu tutor i guia en les que han estat les meves primeres pràctiques i finalment, la meva primera feina. Sense el seu esforç i dedicació en formar-me i compartir amb mi els seus coneixements aquest treball no hagués estat possible. Tampoc puc oblidar-me aquí de l'equip de Personalització, per fer-me sentit valorat i pel gran treball que fa dia rere dia.

En segon lloc, vull agrair-li a l'Edelmira el fet que hagi acceptat aquest treball, sent la temàtica completament aliena al seu àmbit de treball i recerca. Vull agrair-li el feedback donat i disculpar-me per no haver-lo aplicat en la seva totalitat o amb la celeritat desitjada.

Per últim, vull donar les gràcies a la FIB per permetre'm realitzar aquestes pràctiques amb unes condicions poc freqüents en aquest país. La excel·lent formació acadèmica juntament amb les pràctiques han fet de la transició cap al món laboral una etapa molt més dolça.

10 Bibliografia

- [1] Institut d'Estudis Catalans: <https://dlc.iec.cat/> (Visitat 25-02-21)
- [2] Desenvolupament en cascada: https://es.wikipedia.org/wiki/Desarrollo_en_cascada (V. 25-02-21)
- [3] Bruce Schneier: <https://www.schneier.com/> (V. 25-02-21)
- [4] QA: https://en.wikipedia.org/wiki/Quality_assurance (V. 28-02-21)
- [5] Testing tradicional: <https://marutitech.com/traditional-testing-vs-agile-testing> (V. 28-02-21)
- [6] Mango: [https://es.wikipedia.org/wiki/Mango_\(moda\)](https://es.wikipedia.org/wiki/Mango_(moda)) (V. 28-02-21)
- [7] Agile: <https://agilemanifesto.org/> (V. 28-02-21)
- [8] Scrum: [https://en.wikipedia.org/wiki/Scrum_\(software_development\)](https://en.wikipedia.org/wiki/Scrum_(software_development)) (V. 28-02-21)
- [9] Test A/B: <https://www.cyberclick.es/numerical-blog/que-es-test-a/b> (V. 08-03-21)
- [10] Bitbucket: <https://bitbucket.org/> (V. 08-03-21)
- [11] Jira: <https://www.atlassian.com/software/jira> (V. 08-03-21)
- [12] Confluence: <https://www.atlassian.com/software/confluence> (V. 08-03-21)
- [13] Gherkin: <https://openwebinars.net/blog/que-es-gherkin/> (V. 08-03-21)
- [14] Cucumber: <https://cucumber.io/> (V. 08-03-21)
- [15] Java: https://www.java.com/es/about/whatis_java.jsp (V. 09-03-21)
- [16] Selenium WebDriver: <https://www.selenium.dev/documentation/en/webdriver/> (V. 09-03-21)
- [17] [18] BrowserStack: <https://www.browserstack.com/> (V. 09-03-21)
- <https://en.wikipedia.org/wiki/BrowserStack>
- [19] [20] Jenkins: <https://www.jenkins.io/> (V. 10-06-21)
- [https://en.wikipedia.org/wiki/Jenkins_\(software\)](https://en.wikipedia.org/wiki/Jenkins_(software))
- [21] Retrofit: <https://square.github.io/retrofit/> (V. 10-06-21)
- [22] Maven: <https://maven.apache.org/>
- [23] Teams: <https://www.microsoft.com/es-es/microsoft-teams/group-chat-software>
- [24] [25] Cypress: <https://www.cypress.io/> (V. 17-05-21) <https://www.cypress.io/blog/2020/07/08/end-to-end-testing-mobile-apps-with-ionic-and-cypress/>
- [26] JavaScript: <https://developer.mozilla.org/ca/docs/Web/JavaScript> (V. 22-03-21)
- [27] Python: <https://www.python.org/>
- [28] Kotlin: <https://kotlinlang.org/> (V. 22-03-21)
- [29] RestTemplate: <https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/client/RestTemplate.html> (V. 22-03-21) Bibliografia adicional:
- Miró <https://miro.com/app/dashboard/> (V. 20-06-21)
- El llibre Scrum y XP desde las trincheras <http://www.proyectalis.com/wp-content/uploads/2008/02/scrum-y-xp-desde-las-trincheras.pdf> (V. 20-01-21)