Testing d'aplicacions Progress

Adil Asoufi Bniyech

Resum— Davant la complexitat de testejar aplicacions heretades sense accés al codi font, que exigeix metodologies automatitzades específiques, aquest projecte proposa una metodologia per automatitzar el testing d'aplicacions legacy desenvolupades amb Progress OpenEdge, utilitzades a la CCMA (Corporació Catalana de Mitjans Audiovisuals). Davant la dificultat de modificar el codi font d'aquestes aplicacions, s'ha creat un sistema basat en IA, OCR i eines com PyAutoGUI i Pixtral per detectar, testejar i validar el funcionament de components d'interfície gràfica. S'ha desenvolupat un framework amb millores progressives aplicades a aplicacions reals, millorant-ne la fiabilitat i escalabilitat. El sistema permet generar casos de prova automatitzats i informes útils per als equips de QA. Les validacions amb usuaris i tècnics han mostrat bons resultats en aplicacions poc complexes, amb reptes pendents en fluxos més complexos.

Paraules clau— Testing automatitzat, aplicacions legacy, Progress OpenEdge, intel·ligència artificial, OCR, Pixtral, QA, automatització de proves, interfície gràfica, validació de components.

Abstract— Given the complexity of testing legacy applications without access to source code, which requires specific automated methodologies, this project proposes a methodology to automate the testing of legacy applications developed with Progress OpenEdge, used at the CCMA (Catalan Corporation of Audiovisual Media). Due to the difficulty of modifying the source code of these applications, a system based on AI, OCR, and tools such as PyAutoGUI and Pixtral has been created to detect, test, and validate the functionality of graphical user interface components. A framework has been developed with progressive improvements applied to real applications, enhancing reliability and scalability. The system enables the generation of automated test cases and useful reports for QA teams. Validations with users and technicians have shown good results in simple applications, with remaining challenges in more complex workflows.

Index Terms— Automated testing, legacy applications, Progress OpenEdge, artificial intelligence, OCR, Pixtral, QA, test automation, graphical interface, component validation.

1 Introducció - Context del treball

En el desenvolupament de programari, la qualitat del codi i la seva fiabilitat són essencials per garantir aplicacions robustes i mantenibles. No obstant això, moltes empreses encara depenen d'aplicacions heretades que presenten múltiples reptes, especialment en termes de testing i manteniment.

En la Corporació Catalana de Mitjans Audiovisuals (CCMA), estan utilitzant aplicacions amb sistemes de programari que gestionen processos crítics i requereixen de una metodologia de testing més automatitzat per garantir-ne l'estabilitat i evolució sense alterar el seu codi font, com són les aplicacions Progress [1].

Les aplicacions Progress o Progress OpenEdge ABL (Advanced Business Language) són sistemes desenvolupats amb Progress OpenEdge, una plataforma que combina un llenguatge de programació de quarta generació (4GL) amb un sistema de gestió de bases de dades relacional (RDBMS). Són molt utilitzades en l'àmbit empresarial per la seva robustesa i capacitat d'integració. Tot i això, moltes han estat en ús durant dècades sense una modernització significativa, fet que en dificulta la compatibilitat amb eines actuals de testing i manteniment.

La manca de proves automatitzades no només incrementa el risc d'errors en producció, sinó que també suposa un fre important per a l'evolució d'aquests sistemes. Automatitzar el testing en aquest tipus d'entorns permetria reduir costos de manteniment, facilitar la introducció de canvis i garantir una major fiabilitat del programari.

Aquest projecte té com a objectiu desenvolupar una metodologia efectiva per al testing automatitzat d'aplicacions Progress, especialment pensada per a entorns legacy com els de la CCMA. L'objectiu és facilitar la validació contínua del programari amb processos eficients, mesurables i automatitzables, aprofitant les eines d'intel·ligència artificial que tenim al nostre abast per optimitzar-ne cada fase.

Per assolir aquest repte, es farà primer una revisió de l'estat de l'art en testing automatitzat, amb especial atenció a tècniques aplicables a sistemes antics o amb dificultats d'integració. Posteriorment, s'analitzaran eines existents al mercat per identificar aquelles més adequades al cas concret de Progress. També es definirà un conjunt de mètriques per mesurar la cobertura i l'eficàcia de les proves, i es crearà un dataset específic de proves per validar la metodologia proposada. A més, es durà a terme una anàlisi detallada de les aplicacions de la CCMA per entendre millor els seus requisits i limitacions, garantint així una solució alineada amb la realitat del seu ecosistema tecnològic.

El treball es desenvoluparà en diverses fases, que s'aniran explicant a les següents parts, amb la implementació d'una aplicació per testejar i validar la metodologia aplicada i aproximar-se als resultats que es volen presentar.

A continuació, es procedirà a explicar els objectius del projecte, l'estat de l'art, la metodologia utilitzada, la planificació duta a terme, el desenvolupament del projecte, un anàlisi i discussió dels resultats, i les conclusions extretes.

- *E-mail de contacte: adilasoufi@gmail.com*
- *Menció realitzada: Tecnologies de la Informació*
- *Treball tutoritzat per: Ramon Martí Escalé (DEIC)*
- *Curs 2024/25*

2 Objectius del projecte

L'objectiu principal d'aquest treball és **desenvolupar una metodologia efectiva per testejar aplicacions Progress sense la necessitat de modificar el codi font original**. Aquest repte sorgeix del fet que moltes aplicacions Progress són sistemes antics, sovint sense una estratègia de testing automatitzat, cosa que dificulta la seva evolució i manteniment.

A continuació es llistaran tots aquests objectius en ordre de prioritat:

- Es vol **analitzar a fons les dificultats específiques del testing en aplicacions legacy**, concretament tres aplicacions de la CCMA: una per al manteniment d'idiomes, que gestiona la informació lingüística usada per les aplicacions; una altra per al manteniment de països, que gestiona les dades dels països; i finalment, una aplicació per a l'alta de sol·licitud de compres de serveis, que gestiona les compres de serveis a la CCMA.
- **Investigar metodologies i eines utilitzades en altres entorns legacy**, valorant-ne l'efectivitat i la possibilitat d'adaptació a sistemes Progress. L'objectiu és construir una base de coneixement que serveixi per dissenyar una metodologia aplicable sense alterar el codi original.requereixi modificacions en el codi existent.
- **Establir mètriques clares per avaluar la metodologia**. Aquestes mètriques permetran mesurar l'eficàcia i la qualitat de la metodologia en entorns reals.
- **Dissenyar casos de prova específics per a aplicacions Progress**, centrats en escenaris representatius i crítics del seu funcionament. Aquest pas és essencial per validar la capacitat de la metodologia per identificar errors als punts més sensibles de l'aplicació.
- **Validar la metodologia en aplicacions reals de la CCMA**, aplicant-la en entorns pràctics per comprovar-ne la viabilitat i extreure conclusions útils. Això permetrà ajustar i millorar la proposta segons els resultats obtinguts en situacions reals.
- **Proposar una solució final pràctica,** un **software**, en el que incorporant totes les funcionalitats necessàries, es millori la qualitat i seguretat de les aplicacions Progress, sense incrementar la seva complexitat ni comprometre l'estabilitat. L'objectiu és facilitar el manteniment i l'evolució dels sistemes a llarg termini.

3 Estat de l'art

El testing automatitzat és molt important per assegurar que el programari funcioni bé i es pugui mantenir amb facilitat. Tot i això, quan parlem d'aplicacions legacy, com les que estan desenvolupades en Progress, apareixen molts problemes per poder aplicar les tècniques modernes de testing. Aquestes aplicacions solen tenir una arquitectura molt antiga i monolítica, el codi no està gaire modularitzat i la documentació que tenen és insuficient o està desactualitzada. Abans, el testing es feia gairebé tot manualment, cosa que era molt lent i podia portar a errors humans. Amb el temps, s'han creat eines d'automatització que fan que el testing sigui més ràpid i econòmic, però encara hi ha moltes dificultats per integrar-les en aquests sistemes antics. Per això, és necessari buscar noves formes i estratègies que permetin adaptar i fer evolucionar aquestes aplicacions legacy.

3.1 Eines per al Testing Automatitzat en Sistemes Legacy

Avui dia, existeixen diverses eines i enfocaments per a la implementació del testing automatitzat en sistemes legacy. Selenium, per exemple, és una de les eines més àmpliament utilitzades per a l'automatització de proves en aplicacions web, però la seva aplicabilitat en entorns legacy és limitada. Altres solucions, com TestComplete [11] i OmniParser [10] de Microsoft, ofereixen un enfocament més flexible que facilita la integració amb aplicacions més antigues. A més, han sorgit noves tecnologies que exploren la possibilitat d'automatitzar proves en sistemes legacy sense necessitat d'accedir directament al codi font.

3.1.1. OmniParser: Transformació i Processament de Dades

OmniParser, desenvolupat per Microsoft, és una eina de processament i transformació de dades dissenyada per convertir formats estructurats i semi-estructurats en dades útils per a aplicacions modernes. Aquesta eina permet la conversió de dades de documents com logs, arxius CSV, JSON o XML en formats més accessibles per a la seva anàlisi i ús en processos automatitzats. La seva integració en el testing automatitzat facilita la generació de conjunts de dades per a proves, permetent una extracció i validació més eficient de la informació. Això és especialment útil en aplicacions legacy, on les dades solen estar emmagatzemades en formats obsolets o poc estructurats. Amb OmniParser, es pot automatitzar el procés d'extracció de dades per validar el funcionament de l'aplicació en diferents escenaris de prova, assegurant així una cobertura més gran i una millor detecció d'errors en el sistema.

3.1.2. TestComplete: Automatització de proves funcionals

TestComplete, desenvolupada per SmartBear, és una eina d'automatització que permet fer proves d'interfície gràfica en aplicacions web, d'escriptori i mòbils. La seva principal avantatge és que facilita la creació de proves sense coneixements avançats de programació, mitjançant enregistrament i reproducció, i admet llenguatges com Python, JavaScript o VBScript. En aplicacions Progress OpenEdge ABL, s'utilitza per validar la interfície sense modificar el codi font. Les proves són repetibles i escalables, cosa que redueix el temps dedicat a validacions manuals i millora la detecció d'errors. A més, el reconeixement d'objectes i la detecció de canvis permeten adaptar-se a variacions de l'aplicació. Integrar TestComplete dins una estratègia de testing continu ajuda a garantir la fiabilitat i mantenibilitat de les aplicacions legacy de la CCMA.

3.2 Intel·ligència Artificial i Testing Automatitzat

Un dels avenços més prometedors en aquest camp és l'ús

de la intel·ligència artificial per optimitzar el testing automatitzat. Empreses com Anthropic [2] han desenvolupat frameworks innovadors, com Claude 3.5 Computer Use [3], que permeten interactuar amb sistemes sense accedir al codi font. OpenAI ha presentat Operator [7], una plataforma per optimitzar tasques repetitives amb IA, i Google, amb Project Astra dins l'ecosistema Gemini AI [9], ha aplicat models avançats d'aprenentatge automàtic a la prova de programari. Aquestes tecnologies permeten crear proves autònomes, capaces de detectar errors i millorar l'eficiència del procés de verificació.

Una possible integració en el testing d'aplicacions Progress podria incloure l'ús d'OmniParser per estructurar dades de logs i fitxers d'entrada, i agents d'IA com Claude 3.5 o Project Astra per generar i executar proves adaptatives. Aquests agents podrien identificar patrons d'error, suggerir nous casos de prova i adaptar-se a canvis en l'aplicació, millorant la cobertura i reduint la necessitat d'intervenció manual.

3.4 Reptes i Futur del Testing Automatitzat en Aplicacions Legacy

Tot i els avenços recents, encara persisteixen reptes importants que dificulten l'automatització de proves en aplicacions legacy. La manca de compatibilitat amb eines modernes continua sent un obstacle rellevant, així com la dificultat per generar conjunts de dades adequats que permetin validar correctament aquests sistemes. A més, malgrat el progrés en agents d'intel·ligència artificial, la seva capacitat per interpretar i executar proves en entorns tancats o específics encara és limitada, fet que evidencia la necessitat de seguir desenvolupant metodologies adaptades a les particularitats d'aquests entorns.

El panorama actual reflecteix una clara transició cap a enfocaments més sofisticats, on la intel·ligència artificial i l'automatització de processos tenen un paper clau. No obstant això, adaptar aquestes solucions a sistemes com Progress segueix sent un repte pendent. Es necessiten metodologies que permetin una implementació efectiva sense comprometre l'estabilitat ni la funcionalitat del programari existent. En aquest context, la recerca i l'exploració de noves eines són fonamentals per garantir l'evolució d'aquests sistemes i la seva adaptació als requisits actuals del mercat.

4 Metodologia

La metodologia que s'ha seguit per aquest projecte es basa en un enfocament SCRUM adaptat a l'estil de treball de la CCMA i a les necessitats específiques del desenvolupament del treball. El procés ha estat organitzat en 4 sprints de 1 mes cadascun, amb el sprint final al juny, on s'han presentat els resultats finals i l'informe complet. Aquesta estructura va permetre garantir una evolució constant i una metodologia robusta, amb una planificació flexible que s'adapta als canvis i necessitats que poguessin sorgir durant el desenvolupament del projecte.

Cada sprint ha tingut tasques concretes, però el contingut s'ha pogut ajustar segons els resultats obtinguts en el sprint anterior. El primer sprint ha començat amb la investigació de les eines de testing i la definició de la metodologia, mentre que en els següents sprints s'han desenvolupat versions progressives, s'han mesurat els resultats i fet ajustos. Finalment, en el sprint 4, s'han fet les proves finals, s'ha validat la metodologia proposada i s'ha lliurat l'informe final.

El desenvolupament ha sigut de caràcter experimental, implicant la creació i prova de l'aplicació per automatitzar els tests en les aplicacions Progress. En aquest cas, l'experimentació ha consistit en la creació de versions progressives de proves automatitzades, amb els que es volia arribar a una solució final pas a pas i els quals s'han validat al llarg dels sprints a partir de proves amb els datasets que s'han anat dissenyant i utilitzant.

A mesura que s'avançava, les versions de l'aplicació s'han fet més complexes i ajustades a les necessitats concretes de les aplicacions Progress que s'han utilitzat, amb les 3 aplicacions que s'han esmentat anteriorment, per cada una s'anirà adaptant cada versió a cadascun d'aquests programes que ha oferit la CCMA per aplicar al projecte. Aquest procés de prova i ajust s'ha basat en la innovació en cada versió i en aproximacions al resultat final, ha estat iteratiu , per tant, en cada versió es volia millorar els resultats de l'anterior. L'objectiu d'aquesta part ha sigut utilitzar una metodologia efectiva per a aquest projecte, per facilitar els processos que s'han utilitzat de manera que es poguès arribar a una solució que no presenti problemes de consistència des de etapes inicials, validant les eines seleccionades i assegurant que els resultats siguin útils i pràctics per poder utilitzar-los a futur.

5 Planificació

La planificació del projecte ha seguit un enfocament flexible i iteratiu, estructurat en 4 sprints de tres setmanes a un mes cadascun. Aquest plantejament ha permès una evolució progressiva del treball, ajustant-se als resultats obtinguts en cada fase i garantint una adaptació constant a les necessitats del projecte.

Aquest procés ha culminat amb l'últim sprint, en el qual s'han consolidat els resultats finals i s'ha realitzat una validació completa de la metodologia proposada. Aquesta fase ha inclòs una anàlisi detallada de l'efectivitat de les proves automatitzades i la seva aplicabilitat en entorns reals.

A continuació, es farà una descripció de les fases seguides durant la planificació del projecte.

Estudis del projecte: s'ha realitzat una investigació de les eines de testing disponibles.

- Estudi del problema: s'ha fet un anàlisi detallada de les dificultats en el testing de les aplicacions Progress, identificant els principals reptes, investigant quins son els problemes que ens estan donant aquestes aplicacions antigues i com es podrien millorar a partir de la pròpia solució.
- Estudi d'estat de l'art (eines): s'han invetigat les eines i metodologies existents en testing automatitzat per identificar les més adequades per a aquest projecte, i s'ha fet una cerca de les eines que s'aplicarien al nostre projecte i com es podrien integrar.

Anàlisi, definicions i prototipatge: s'ha definit la metodologia que es volia implementar.

- Anàlisi d'alternatives: S'han comparat diferents eines i metodologies per seleccionar les que millor s'adaptaven a les necessitats del projecte, assegurant-ne la viabilitat sense modificar el codi font de les aplicacions.
- Definició de mètriques: va permetre establir indicadors clau per mesurar l'eficàcia de les proves automatitzades, com ara la cobertura de codi, el temps d'execució de les proves i la reducció d'errors detectats.
- Preparació dataset de proves: s'ha dissenyat un conjunt de dades amb l'objectiu de garantir que les proves cobrissin una àmplia varietat de situacions i casos d'ús reals, millorant la fiabilitat dels resultats i la seva aplicabilitat en entorns productius.

Desenvolupament i evolució del producte: s'ha començat el desenvolupament de versions d'aplicació de proves automatitzades, aplicats a les aplicacions Progress de la CCMA.

- Versió inicial (implementació i mètriques): desenvolupament i implementació de la primera versió de l'aplicació de testing automatitzat, avaluant la seva eficàcia amb les mètriques definides. Aquesta versió ha sigut la més senzilla, intentant aplicar-la sobre l'aplicació de manteniment d'idiomes, la que té menys complexitat.
- Versió optimitzada (implementació i mètriques): desenvolupament i implementació de la segona versió de l'aplicació de testing automatitzat, avaluant la seva eficàcia amb les mètriques definides. S'ha millorat la primera versió, per poder aplicar-la sobre l'aplicació de manteniment de països, i s'ha començat a pensar en l'opció de generar els casos de prova de manera automàtica.
- Versió final (implementació i mètriques): desenvolupament i implementació de la versió final de l'aplicació de testing automatitzat, avaluant la seva eficàcia amb les mètriques definides. En aquesta versió s'ha volgut aconseguir una solució més robusta, per aplicar-la sobre l'aplicació d'Alta de Solicituds de Compres de Serveis, amb la idea de poder cobrir aquesta aplicació i també el seu codi font.

Anàlisi i finalització: a mesura que avançaven els sprints, s'han validat els resultats i s'han realitzat ajustos per millorar l'eficàcia del sistema, a més, s'ha redactat l'informe final amb les conclusions extretes del projecte i s'han presentat i discutit els resultats, destacant-ne l'impacte i les possibles millores futures en el testing automatitzat d'aplicacions Progress.

- Testing i validació amb usuaris finals: s'ha fet la divulgació de l'aplicació cap a possibles usuaris finals per que donessin el feedback de les proves que es feien sobre l'aplicació i els resultats que aquesta ens donava.
- Anàlisi i discussió dels resultats: ha consistit en una avaluació exhaustiva dels resultats obtinguts durant el desenvolupament de les versions, examinant l'eficàcia de les proves automatitzades en termes de detecció d'errors, estabilitat del sistema i millora en els temps d'execució. S'han comparat els resultats amb les mètriques establertes prèviament per validar la metodologia proposada i determinar-ne la viabilitat en un entorn real.

Aquesta planificació s'ha fet tenint en compte els temps que se'ns ha donat per fer totes les entregues i seguiments, de manera que si hi hagués qualsevol contratemps, es tingui temps de reacció fins a l'útim sprint, i per això, la planificació acabava a meitats de maig i es tenia marge fins a finals de maig.

Finalment, s'ha planificat i dissenyat l'informe i la presentació final per defensar el projecte davant del tribunal en les dates establertes al juliol.

6 DESENVOLUPAMENT

Com s'ha pogut veure a la planificació, primer s'ha començat fent un estudi del projecte, posteriorment una anàlisi, definicions i prototipatge del projecte, per procedir a desenvolupar el projecte, per, finalment, analitzar i discutir els resultats obtinguts.

6.1 Estudi del projecte

Primerament, es comença a estudiar el problema que tenim endavant, com solucionar-ho i quines eines tenim per ferho actualment.

6.1.1 Estudi del problema

Abans d'iniciar el desenvolupament del projecte, es va considerar essencial dur a terme un estudi detallat de les aplicacions corporatives de la CCMA amb les quals es va decidir treballar. Aquestes aplicacions comprenien sistemes de manteniment d'idiomes per les aplicacions, manteniment de països i una altra més complexa per a la gestió de sol·licituds d'alta de compres de serveis. L'objectiu d'aquesta fase d'anàlisi va ser entendre les funcionalitats clau de cadascuna d'elles, identificar els components de la seva interfície i determinar quines eines eren necessàries per automatitzar les interaccions per testejar la aplicació de manera eficient. Les funcionalitats a provar eren diverses i depenien de cada aplicació, entre les quals es trobaven:

- Manteniment d'idiomes: alta, baixa i modificació d'idiomes, i exportació del llistat d'idiomes a format excel. [Fig. 5]
- Manteniment de països: alta, baixa i modificació de països, filtrat per país, impressió i exportació del llistat de països i auditoria dels canvis fets al país seleccionat. [Fig. 6]
- Alta de sol·licitud de compres de serveis: Dropdowns, textboxes i buttons on s'havia de seleccionar els serveis a comprar, persones de contacte, proveïdors… [Fig. 7]

Una vegada estudiades aquestes aplicacions, es va començar a pensar en quina solució donaríem al problema que teníem per testejar aquestes aplicacions, la qual es va

dur a terme en la següent fase,

6.1.2 Estudi de l'estat de l'art (eines)

Un cop es va tenir una visió clara de les aplicacions objecte d'estudi, es va decidir explorar les opcions tecnològiques disponibles per implementar una solució viable al problema presentat, i a continuació, cercar les tecnologies disponibles per a la seva realització.

Es va decidir que les funcionalitats finals del projecte serien les següents:

1. Capturar imatges de la UI: fent iteracions contínues rere cada interacció amb l'aplicació en moments clau.
2. Extreure components de la interfície a testejar, per aconseguir saber quins són els components més importants que hauríem de provar.
3. Analitzar l'estructura de la UI, per veure quines són les funcionalitats importants a testejar del resultat dels components de la interfície extrets, a partir d'intel·ligència artificial.
4. Interactuar automàticament amb l'aplicació, de manera que no requereixi cap esforç manual de l'usuari.
5. Comparar els resultats esperats amb els obtinguts per detectar errors.
6. Automatitzar l'execució de les proves en cada nova versió de l'aplicació.

En aquest sentit, es va dur a terme una recerca sobre diferents sistemes de reconeixement òptic de caràcters (OCR), ja que resulta fonamental poder interpretar els elements de la interfície de manera automàtica, per aconseguir fer l'extracció dels components a testejar. Entre les opcions analitzades, es van valorar solucions com Tesseract OCR, una eina coneguda per la seva capacitat de reconèixer text en imatges, i Omniparser, que ja es va valorar prèviament com una alternativa viable.

A més de l'OCR, també es va contemplar la possibilitat d'integrar un framework per a la interacció amb les aplicacions, de manera que poguem fer les captures esmentades a l'estructura de la solució. Inicialment, es van valorar opcions com Selenium [13] que està més enfocat en interaccions amb aplicacions web o mòbils, SikuliX [14], que està bastant enfocat en aplicacions d'escriptori, i AutoIt[15], centrat en automatitzar acci-ons ens finestres in controls de UI (interfície d'usuari) però després d'analitzar-ne les capacitats i limitacions, es va decidir utilitzar PyAutoGUI[16] com a eina prin-cipal per a la simulació d'interaccions amb la interfície. Aquesta elecció es va basar en la seva flexibilitat i facili-tat d'ús per manipular aplicacions gràfiques, a més de la comoditat que ens donava al haver escollit Python com a llenguatge per desenvolupar la nostra aplicació, ja que aquest framework està especialitzat i té més faci-litats per aplicar-ho en entorns Python.

6.2 Anàlisi, definicions i prototipatge

En aquesta etapa, un cop decidit com es faria l'aplicació i quines eines s'utilitzarien, s'ha començat a analitzar les alternatives i les bases que es volien seguir.

6.2.1 Anàlisi d'alternatives

Amb tota la informació recollida durant la fase d'anàlisi, es va decidir que es passaria a provar una alternativa senzilla, per començar a provar com funcionarien totes aquestes eines que es van descobrir aplicades al projecte, i posteriorment, es definiria l'estratègia de disseny del projecte.

En primer lloc, es va considerar necessari realitzar proves inicials amb una petita aplicació sense interfície pròpia, un petit projecte que seguia una seqüència de passos per validar el funcionament bàsic de l'OCR i la interacció amb la interfície de l'aplicació a provar. Per aquest motiu, es va decidir començar amb el Bloc de notes (Notepad) com a alternativa més bàsica per testejar, i provar les seves funcionalitats que eren 4 botons (Fitxer, format, edició, visualització i ajuda), ja que la seva simplicitat permet fer proves controlades sense la complexitat d'aplicacions més avançades.

Un cop es van començar a obtenir resultats amb

Fig. 2: Previsualització del que seria una versió preliminar del software desenvolupat per a les proves

Notepad, es va decidir traslladar les proves a un entorn més realista [Fig. 2]. Per fer-ho, es van seleccionar aplicacions internes de l'empresa desenvolupades amb Progress, que tenen una interfície més complexa i s'ajusten millor als requeriments del projecte. Aquesta transició va resultar fonamental per validar si el sistema desenvolupat és aplicable en entorns de producció real.

Com a alternativa i seguint el fluxe indicat, es va plantejar el desenvolupament d'un software que funcionava iniciant un executable que iniciava la nostra aplicació. Es suggeria el Notepad, com aplicació d'exemple per executar les proves, però es podia modificar canviant l'input, i quan es llançava l'aplicació, es donava l'opció de fer una nova captura de la pantalla o detectar els components sobre la captura que es feia automàticament al iniciar-la.

El software i la seva interfície es van desenvolupar amb el framework de Python Tkinter [12], ja que és una llibreria estàndard inclosa amb Python que permet crear interfícies gràfiques d'usuari de manera senzilla i ràpida sense dependències externes. Tkinter ofereix una bona integració amb el sistema operatiu, és multiplataforma, i resulta especialment adequada per a aplicacions d'escriptori lleugeres o prototips.

Un cop fet tot aquest procés, s'hauria de seleccionar un component per executar-hi les proves i donant-li a "Ejecutar Pruebas" es començaria a fer tot el procés de prova sobre el component seleccionat.

La interfície inicialment està formada per els botons que et permeten connectar-te o executar l'aplicació seleccionada, a més de capturar la pantalla per fer les proves i buscar els components de l'aplicació.

A més, es va incorporar la funcionalitat de seleccionar un o més components per executar-hi les proves. Un cop seleccionats, es va implementar el botó "Ejecutar Pruebas" per iniciar el procés de test sobre els components escollits. També es va incloure la possibilitat d'eliminar components detectats per error i de generar informes de les proves en format HTML, els quals es podien visualitzar immediatament després de l'execució de les proves. [Fig.3].

En el disseny inicial del sistema, es va definir un flux de treball on Tesseract OCR s'encarregava de detectar els components visibles de la interfície. Aquests components es processaven a través d'un prompt dirigit a un agent d'intel·ligència artificial (GPT-4o), el qual suggeria com interactuar amb cadascun d'ells. Finalment, PyAutoGUI executava les accions proposades de forma automatitzada. No obstant això, aquesta aproximació inicial va demostrar tenir una fiabilitat limitada, especialment en la interpretació del context visual i funcional dels elements detectats.

A mesura que es van començar a provar les aplicacions desenvolupades en Progress, es va detectar un altre problema crític: les proves executades no eren prou fiables. Els components es detectaven de manera inconsistent, els resultats de les proves sovint no eren els esperats i hi havia una dificultat constant per garantir que les interaccions fossin precises i repetibles. Aquestes deficiències van portar a replantejar el procés i a introduir millores significatives en la manera com es gestionaven les proves i la interpretació dels components.

6.2.2 Definició de mètriques

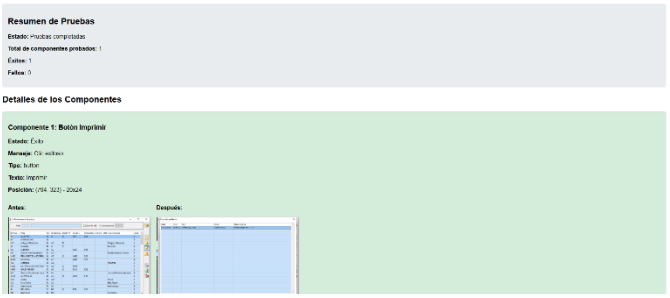

Fig. 3: Exemple simple d'informe generat després dels resultats obtinguts de les proves realitzades

Un cop establertes les eines i la metodologia de test, es va considerar fonamental definir mètriques que permetessin mesurar l'eficàcia del sistema de proves automatitzades. Per això, es van identificar diversos indicadors clau, entre els quals es trobaven:

- La cobertura de codi: quantitat de codi que estem arribant a testejar amb la nostra aplicació.(Importància mitjana ja que volem arribar a provar les funcionalitats més importants)
- Temps d'execució de les proves: quant triga en respondre la nostra aplicació a les proves que es fan. (Importància alta)
- Precisió de detecció dels components(interactius i no): volem arribar a tenir una precisió considerablement alta per no tenir errors a l'hora de provar els components.(Importància alta)
- Reducció d'errors detectats en comparació amb les proves manuals anteriors, ja que un dels objectius principals és aconseguir reduir els errors de manera iterativa (Importància alta)
- Precisió en la posició dels components detectats: volem assegurar la major precisió en on es situen aquests components que detecta la IA.

Aquestes mètriques es van anar integrant al sistema de test de forma progressiva, amb l'objectiu de monitorar i millorar contínuament la qualitat i la fiabilitat dels resultats obtinguts en cada iteració. A més, s'aprofitava la funcionalitat de generació d'informes en format HTML per facilitar la lectura i anàlisi dels resultats obtinguts durant les proves, de manera que teníem una transcripció del que havíem aconseguit fins a aquell moment de manera contínua.

6.2.3 Preparació dataset de proves

Tenint en compte les mètriques esmentades i els objectius d'aconseguir cobertures altes, es va decidir que un punt molt important era cobrir de manera robusta els components més importants de les aplicacions a provar, per tant es va decidir dissenyar un conjunt de dades específic per a la validació del sistema. Aquest data set es va compondre de captures de les interfícies del programari que volíem testejar, aplicació de manteniment d'idiomes [Fig.5], manteniment de països [Fig.6] i alta de sol·licitud de compra de serveis [Fig.7] amb els seus components per obtenir uns resultats reals.

Es van etiquetar els components amb l'OCR (Tesseract),

per entrenar o validar els sistemes d'extracció d'informació visual, assegurant d'aquesta manera una informació real que el sistema pogués reconèixer tots aquests components amb les seves posicions detectades per la lectura òptica d'imatge, i es van procedir a fer simulacions d'anàlisis dels components d'aquestes aplicacions, de les quals s'extreien els resultats segons les mètriques aplicades al data set de proves.

6.3 Desenvolupament i evolució del producte

A continuació s'explicarà com s'ha procedit amb el desenvolupament de cada versió del projecte

6.3.1 Versió inicial

Davant les limitacions detectades en la fase de disseny, per a la primera versió es van decidir introduir canvis substancials en la manera com es processaven les dades. Un dels problemes principals va ser que el sistema depenia massa d'un agent d'IA no especialitzat en OCR, fet que reduïa la precisió en la detecció i classificació dels components de la interfície. Per solucionar-ho, es va iniciar una cerca d'alternatives més robustes i es va descobrir, a partir de les proves que es van realitzar amb l'OCR i les simulacions, que Mistral AI[17] oferia una solució especialitzada en lectura d'imatges anomenada Pixtral[18]. Aquesta eina va demostrar tenir una capacitat molt superior en l'extracció d'informació visual i la interpretació d'interfícies gràfiques, per la qual cosa es va decidir adoptar-la com a nova eina principal per a la detecció de components.

Amb la incorporació de Pixtral, el sistema va començar a oferir millors resultats en la detecció i interpretació dels elements de la interfície. Tot i això, encara es va mantenir un problema rellevant: les proves executades no eren prou fiables. Fins aquell moment, el procés seguit va consistir en tres fases principals:

- La detecció dels components de la interfície.
- L'execució de proves sobre components específics.
- La interacció automàtica amb aquests components mitjançant IA, on es detecta el comportament resultant i es fan ajustaments recursius.

Aquest enfocament va presentar un desafiament important: el sistema sovint identificava components antics com a nous, la qual cosa generava errors en les proves. A més, com que el procés de proves es basava en una seqüència d'interaccions guiades per IA, el model no sempre generava respostes consistents, fet que dificultava la reproducció exacta de les proves. També es va observar que no es guardava un històric de les accions executades, simplement es redescobria constantment l'aplicació com si totes les accions fetes anteriorment no s'haguessin executat, i es va descobrir que aquesta funcionalitat era molt difícil d'aplicar amb fiabilitat en aquell moment.

6.3.2 Versió optimitzada

Davant d'aquests reptes, per a una versió més avançada del projecte, es va decidir canviar la manera com es gestionaven les proves i adoptar un enfocament més estructurat. En comptes de centrar-se a testejar directament els components, es va definir que com a objectiu final es volia generar una matriu de proves (una taula amb la llista dels casos de prova amb diferents *tags* referents a característiques dels casos de prova) basada en la detecció i estudi dels components. Aquesta nova estratègia va permetre definir característiques clau per a cada element de la interfície, i d'aquesta manera es va aconseguir la funcionalitat de crear un conjunt del que es coneix com features que descriuen el seu comportament. Posteriorment, es va preveure que les persones que utilitzessin l'aplicació poguessin tenir automatitzat aquest procés, com a mínim, sobre els components que formaven part del que es coneix com Golden Path[19] en el món del testing, de manera que es pogués agilitzar el llançament cap a producció de les aplicacions que es volien automatitzar.

A més, es va començar a treballar amb la idea d'utilitzar diagrames de flux que permetessin establir camins sobre aquests components que formaven part del Golden Path [Fig. 4] per definir les seqüències d'accions necessàries per validar el correcte funcionament d'un component. Aquesta aproximació va assegurar que el sistema de proves no es basava només en interaccions recursives poc fiables, sinó

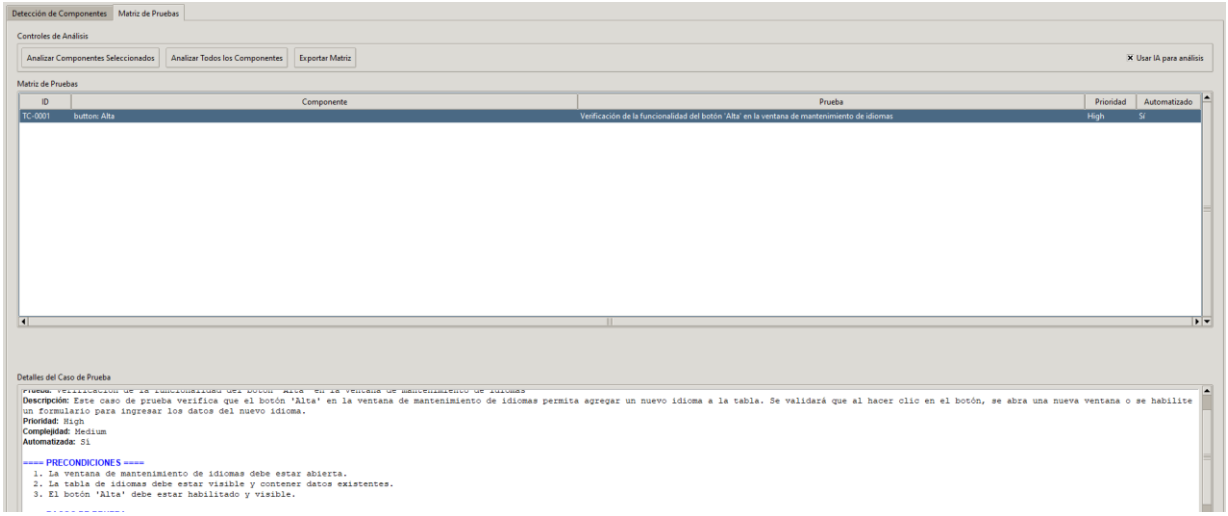

Fig. 4: Imatge de la primera versió de la generació dels casos de prova, on s'ha testejat el "botó d'alta" i un quadre inferior amb el cas de prova explicat

en una estructura organitzada que seguia una lògica predeterminada.

El projecte es va anar pujant i actualitzant a un repositori privat de Bitbucket de la CCMA per mantenir un control exhaustiu del desenvolupament i poder tenir un control de les versions que es volien mantenir o recuperar.

6.3.3 Versió final

A continuació, es van continuar detectant limitacions en el la versió optimitzada.

Una de les limitacions més rellevants va ser relacionada amb les crides a l'API de Pixtral. En general, quan es treballa amb APIs d'intel·ligència artificial, aquestes fan servir sistemes de tokens que limiten el volum d'informació que es pot enviar i rebre. En el cas de Pixtral, el límit que es va establir és de 2.000 tokens per petició, fet que implica que el màxim d'elements retornats per cada crida és d'aproximadament 17-18 components. Aquesta restricció obliga a limitar el nombre de components que es poden testejar per aplicació, almenys de manera immediata. Per aquest motiu, es va decidir desenvolupar un *prompt* específic [Fig. 8] que indicava a la IA que prioritzés sempre els components més importants de l'aplicació a provar.

Una altra limitació important que es va detectar era el temps de resposta de Pixtral, que resultava especialment lent durant la detecció dels components. Aquesta lentitud generava bloquejos en la nostra aplicació, ja que quedava congelada mentre esperava una resposta. Per mitigar aquest problema, es va decidir incorporar una pantalla de càrrega que es mostrava durant cada crida a l'API, millorant així l'experiència d'usuari i oferint un flux de treball molt més fluid i estable.

També es va observar que, en aplicacions amb interfícies compostes principalment per icones, els resultats de la detecció no eren gaire precisos ni consistents entre diferents execucions. Per solucionar-ho, es van enriquir els *prompts* amb més informació contextual i, paral·lelament, es va afegir una funcionalitat que permetia a l'usuari introduir manualment context sobre la seva aplicació. Aquesta informació incloïa detalls sobre l'estructura de la interfície, la posició dels components i funcionalitats clau, fet que ajudava la IA a generar respostes més acurades.

A partir d'aquesta mateixa idea, també va sorgir la proposta d'integrar documentació específica del projecte. Es va implementar la possibilitat d'afegir un fitxer README.md a l'aplicació, que es podia enviar conjuntament amb el *prompt* a la IA, incrementant així el coneixement del sistema sobre el context i objectius de la prova.

En definitiva, l'estratègia adoptada es va centrar en enriquir progressivament la informació proporcionada a la IA, amb l'objectiu d'obtenir respostes més robustes, coherents i repetibles entre iteracions, incrementant així la fiabilitat global del sistema.

Finalment, es va identificar una altra limitació significativa: fins al moment, només es permetia testejar un component per cada crida a l'API, fet que alentia el procés de detecció i generació de proves. Davant d'aquest inconvenient, es va optar per implementar la funcionalitat que permetés testejar múltiples components alhora i generar més d'un *feature* per cada crida. Aquesta millora va suposar una optimització substancial del temps i recursos, fent que el sistema fos molt més àgil i escalable.

Amb tot això, la interfície de la aplicació ha quedat definitivament com es pot veure a la imatge de la pàgina principal de l'aplicació on es pot veure el funcionament, amb botons per connectar-se a l'aplicació, fer captures de pantalla, detectar components, components detectats, analitzar components, generar matriu de proves en format HTML, entre d'altres [Fig.9] i la segona pantalla que es pot veure a la imatge de la figura 10 de l'apèndix [Fig.10].

6.4 Anàlisi i finalització

En aquest apartat s'explicaràn les anàlisis dutes a terme sobre el projecte i les discussions, a més de com s'ha dut a terme la finalització del treball.

6.4.1 Testing i validació amb usuaris finals

En aquesta fase, no s'ha realitzat un procés de testejat

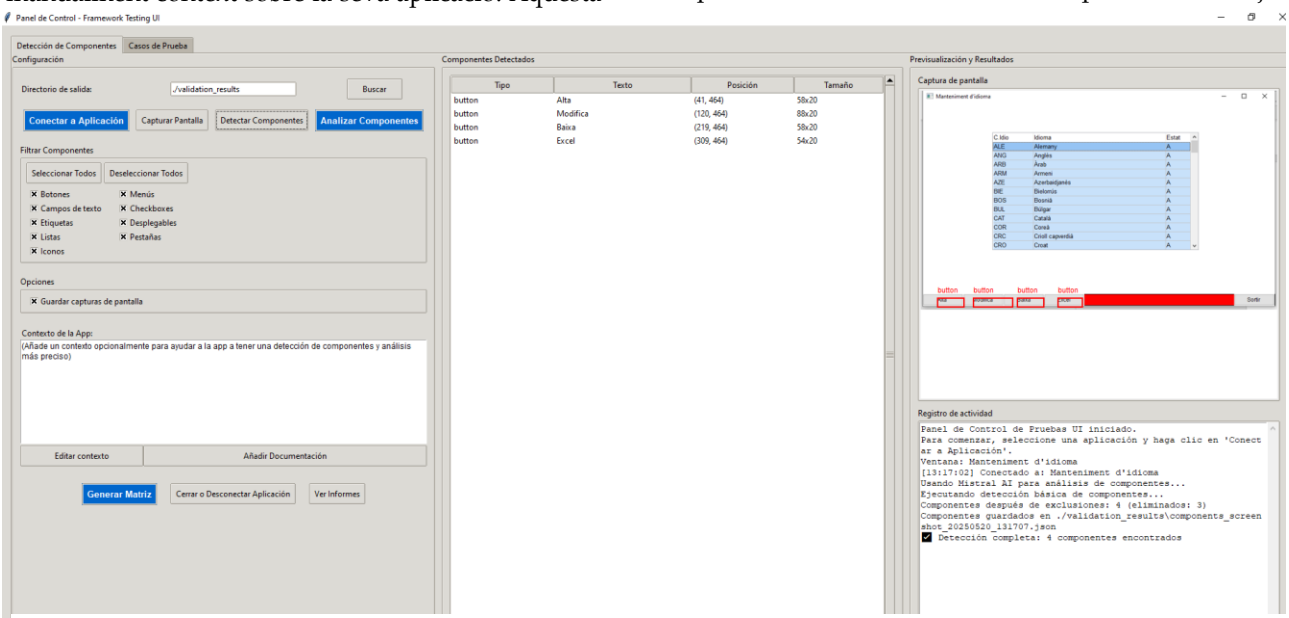

Fig. 9: Pantalla principal de l'aplicació actual

formal, sinó que s'han dut a terme comprovacions constants de la fiabilitat de les implementacions a mesura que es van desenvolupant. Això significa que, en lloc de fer una bateria de proves al final, s'han anat provant totes les funcionalitats implementades sobre les diferents aplicacions, validant-ne el funcionament i ajustant els errors detectats.

Les simulacions es basen en la detecció de components i l'execució de proves per observar com responen les aplicacions a les interaccions automatitzades. A mesura que es feien aquestes proves, es continuaven afinant els resultats obtinguts i millorant tots aquests errors o *bugs* a mesura que s'anaven detectant, per tal d'assegurar que el sistema no només detecta correctament els elements de la interfície, sinó que també pot interactuar-hi de manera efectiva.

Amb aquest enfocament iteratiu, s'han anat corregint les inconsistències a mesura que apareixien i s'ha millorat la fiabilitat del sistema progressivament, assegurant que pugui ser aplicat en entorns reals de manera estable i eficient.

A més d'això, un cop es va assolir un punt en què l'aplicació estava pràcticament desenvolupada de manera definitiva, es van començar a compartir els resultats dels informes amb usuaris reals de les aplicacions que s'estaven provant. L'objectiu era comprovar que els informes generats cobrien completament les funcionalitats de les aplicacions a testejar i que els casos de prova complien amb els estàndards esperats. També es va verificar que els informes fossin clars, entenedors i útils per als equips implicats.

Paral·lelament, es va compartir l'aplicació amb els companys del departament de QA per validar-ne l'accessibilitat i la facilitat d'ús. Es volia confirmar que l'eina complia correctament la seva funció com a aplicació de testing i que realment representava una ajuda efectiva en els processos de qualitat. Aquestes interaccions amb diferents perfils professionals van permetre recollir propostes de millora, idees i comentaris valuosos sobre el nivell assolit pel desenvolupament. Gràcies a aquest procés de validació col·laborativa, es va poder determinar si l'aplicació realment facilitava la feina dels equips de QA i si estava alineada amb les necessitats pràctiques dels seus usuaris finals.

6.4.2. Anàlisi i discussió dels resultats

A totes les persones amb qui es va compartir l'aplicació i els resultats obtinguts, se'ls va fer arribar una enquesta per recollir-ne l'opinió. Es van dissenyar dues enquestes diferenciades: una adreçada als usuaris finals de les aplicacions provades i una altra adaptada específicament per a l'equip de QA, tenint en compte les seves funcions i experiència en processos de qualitat. Aquestes enquestes, que es poden consultar a l'apèndix [10.1], van permetre obtenir una visió clara sobre el nivell de qualitat que s'estava assolint amb l'eina, així com mesurar el grau de satisfacció dels possibles usuaris finals i professionals que podrien fer-ne ús real en un entorn de producció.

Amb els usuaris finals, es va poder constatar que, per a aplicacions més senzilles com ara el Manteniment d'Idiomes o el Manteniment de Països, el framework desenvolupat funciona amb un grau de precisió prou acceptable per dur a terme les proves automatitzades. Tanmateix, a mesura que l'aplicació es complica —especialment en aquelles que depenen de dades dinàmiques o de la interacció entre diversos components—, com en el cas de l'Alta de Sol·licitud de Compres de Serveis, els resultats obtinguts se situen força per sota del nivell esperat.

Aquestes limitacions es deuen principalment al fet que, en l'estat actual, el sistema no permet encadenar de manera efectiva diversos components entre si per verificar-ne el funcionament conjunt. Això impacta especialment en aquelles funcionalitats que depenen del resultat o l'estat d'altres elements, reduint així la capacitat del framework per validar fluxos complexos de l'aplicació.

Amb l'equip de QA, es va arribar a la conclusió que el framework proporciona una base sòlida i pràctica per a l'automatització de proves sobre interfícies, especialment en entorns on es requereix una resposta ràpida amb configuracions mínimes. En general, les valoracions obtingudes a partir de l'enquesta reflecteixen un grau alt de satisfacció en aspectes clau que s'havien definit prèviament com a mètriques rellevants, com ara la cobertura funcional, l'eficiència en la reducció del temps i l'esforç manual, i la facilitat d'ús de l'aplicació. També es va destacar molt positivament la utilitat de la generació automàtica de casos de prova, que representa un suport important per a les tasques habituals de QA.

Tanmateix, es van detectar algunes àrees de millora. Tot i que la detecció de components i la fiabilitat general del sistema —fortament condicionades per la precisió de la IA— van ser ben valorades, en escenaris més complexos (com, per exemple, la identificació d'icones específiques en el manteniment d'Idiomes), es va posar de manifest la necessitat d'una major robustesa. Així mateix, es va suggerir reforçar l'encadenament d'interaccions entre components per poder validar millor els fluxos de treball més elaborats. En conjunt, els resultats de l'enquesta mostren un alt potencial de l'eina i un bon nivell d'acceptació per part de l'equip, alhora que apunten a un ampli marge de millora pel que fa a la seva versatilitat i escalabilitat en entorns més exigents.

En relació amb els objectius planejats al començament del projecte, es pot dir que s'han assolit en la totalitat, amb opcions de continuar el treball en el futur. L'eina desenvolupada ha permès testar aplicacions Progress sense modificar-ne el codi font. Tot i les limitacions detectades en aquests escenaris més complexos, els resultats mostren que la metodologis és util i aplicable en entorns reals.

Pel que fa a la viabilitat del projecte, es confirma que és una feina assumible per un enginyer informàtic. Tant el plantejament com la implementació utilitzen coneixements habituals dins l'àmbit professional, i l'enfocament pràctic demostra que aquesta proposta pot aportar valor real en equips de desenvolupament i qualitat de programari.

7 CONCLUSIONS

Aquest treball ha tingut com a punt de partida la necessitat de facilitar el testing d'aplicacions *legacy* desenvolupades amb Progress OpenEdge, un entorn on la integració d'eines modernes de qualitat sovint és limitada. La motivació principal ha estat explorar com es podria automatitzar la validació de la interfície d'usuari en aquests entorns, reduint l'esforç manual que avui dia encara recau en els

equips de QA. El repte era clar: dissenyar una solució que pogués adaptar-se a sistemes antics sense modificar-ne el codi, però aprofitant tecnologies actuals com la visió per computador i la intel·ligència artificial.

Per assolir aquest objectiu, s'ha desenvolupat una eina que automatitza la generació i execució de proves funcionals mitjançant la detecció visual de components. A través de diverses iteracions, s'han posat en pràctica tècniques com la captura automatitzada d'imatges, l'anàlisi visual amb Pixtral, la interpretació d'elements mitjançant OCR i la generació de fitxers .feature per integrar l'eina amb Cucumber. S'ha treballat amb aplicacions reals de la CCMA i s'ha dissenyat una metodologia que permet obtenir casos de prova de manera automàtica a partir de la interfície existent.

Els resultats obtinguts mostren que la proposta és viable en escenaris senzills o mitjanament complexos. En aplicacions com el Manteniment d'Idiomes o el Manteniment de Països , el sistema ha aconseguit generar i executar proves de forma fiable, tot i que el Manteniment de Països ha donat petits problemes a l'hora de la detecció de components. Tanmateix, s'han evidenciat limitacions importants quan s'intenta aplicar a aplicacions més complexes, com l'Alta de Sol·licitud de Compres de Serveis, on les interaccions entre components o la dependència de dades dinàmiques dificulten l'efectivitat de les proves automatitzades.

Una enquesta de satisfacció adreçada a l'equip de QA ha confirmat l'interès i la utilitat pràctica de l'eina, tot destacant-ne la facilitat d'ús, la integració senzilla i la capacitat d'estalviar temps. No obstant això, també s'ha fet evident la necessitat de millorar la robustesa de la detecció de components visuals, així com d'habilitar la definició de fluxos de prova més complexos i contextuals. Algunes limitacions, com la dependència de serveis externs com Pixtral o la imprecisió davant icones i gràfics no textuals, marquen àrees clares de millora.

De cara al futur, les línies de treball haurien d'anar orientades a millorar l'eficiència i l'escalabilitat del sistema, reduir la dependència de serveis externs, enriquir el context semàntic de les proves i dissenyar una arquitectura modular capaç de gestionar seqüències d'accions més complexes. També seria interessant explorar l'ús de models més especialitzats en interfícies i millorar la capacitat del sistema per aprendre de l'historial de proves executades.

Per acabar es pot dir que aquest projecte demostra que és tècnicament factible iniciar el camí cap a una automatització funcional del testing en aplicacions *legacy*. Tot i les limitacions actuals, les bases establertes representen un pas important per apropar-se a una solució que pugui millorar de forma real la qualitat del programari en entorns tradicionals com el de la CCMA.

8 Agraïments

Primerament, m'agradaria donar les gràcies per tot el support de la meva família, per estar al meu costat, i perquè sense ells no podria haver arribat fins on sóc ara.
En segon lloc, m'agradaria agraïr al meu tutor Rafael Bermúdez Guijo per tot el support donat i acompanyament, tant com en la tutorització a l'empresa del projecte, com a l'estada de pràctiques.

Finalment, voldria agraïr al meu tutor per part de la universitat, Ramón Martí Escalè, pel seu acompanyament i aconsellament de cara a tot el procès de redactament d'informes i planificació del projecte.

9 Bibliografia

[1] Progress Software Corporation, "Connectivitat de dades a aplicacions i dades Progress OpenEdge", Progress, Disponible a: https://www.progress.com/data-connectivity/openedge-applications-and-data. [Accedit: febr. de 2025].

[2] Anthropic, "Computer Use (beta) - Anthropic," Disponible a: https://docs.anthropic.com/en/docs/build-with-claude/computer-use#computer-tool. [Accedit: febr. 2025].

[3] C. Greyling, "Anthropic's Claude 3.5 Computer Use Framework (AI Agent)," Medium, 2024. Disponible a: https://cobusgreyling.medium.com/anthropics-claude-3-5-computer-use-framework-ai-agent-6b3c48dac410. [Accedit: febr. 2025].

[4] IEEE, "Get API Key - Anthropic," Disponible a: https://docs.anthropic.com/en/api/admin-api/apikeys/get-api-key. [Accedit: febr. 2025].

[5] IEEE Xplore, "Agentic AI: Autonomous Intelligence for Complex Goals—A Comprehensive Survey", Disponible a: https://ieeexplore.ieee.org/abstract/document/10849561. [Accedit: febr. 2025].

[6] Anthropic, "3.5 Models and Computer Use," Disponible a: https://www.anthropic.com/news/3-5-models-and-computer-use. [Accedit: febr. 2025].

[7] OpenAI, "Introducing Operator," Disponible a: https://openai.com/index/introducing-operator/. [Accedit: febr. 2025].

[8] Akira AI, "GUI Agents Automate Repetitive Tasks," Disponible a: https://www.akira.ai/blog/gui-agents-automate-repetitive-tasks. [Accedit: febr. 2025].

[9] Google DeepMind, "Google Gemini AI Update (December 2024)," Disponible a: https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/#project-astra. [Accedit: febr. 2025].

[10] Microsoft, "OmniParser," Disponible a: https://microsoft.github.io/OmniParser/. [Accedit: febr. 2025].

[11] SmartBear, "TestComplete - Automated UI Testing Tool," Disponible a: https://smartbear.com/product/testcomplete/. [Accedit: febr. 2025].

[12] "Tkinter — Python interface to Tcl/Tk." Python Software Foundation. Disponible en: https://docs.python.org/es/3.13/library/tkinter.html. [Accedit: febrer 2025].

[13] Selenium web. Disponible a: https://www.selenium.dev/ [Accedit: març. 2025].

[14] SikuliX web. Disponible a: http://sikulix.com/ [Accedit: març. 2025].

[15] AutoIt Script. Disponible a: https://www.autoitscript.com/site/ [Accedit: març. 2025].

[16] PyAutoGUI Documentation. Disponible a: https://pyautogui.readthedocs.io/en/latest/ [Accedit: març. 2025].

[17] Mistral AI Documentation. Disponible a: https://docs.mistral.ai [Accedit: abril. 2025].

[18] Mistral. (2025, març 5). Pixtral: Mistral's Large Vision Model. Disponible a: https://mistral.ai/news/pixtral-large [Accedit: abril. 2025].

[19] Red Hat DevOps Golden Paths. Disponible a: https://www.redhat.com/en/topics/devops/golden-paths [Accedit: abril. 2025].

APÈNDIX

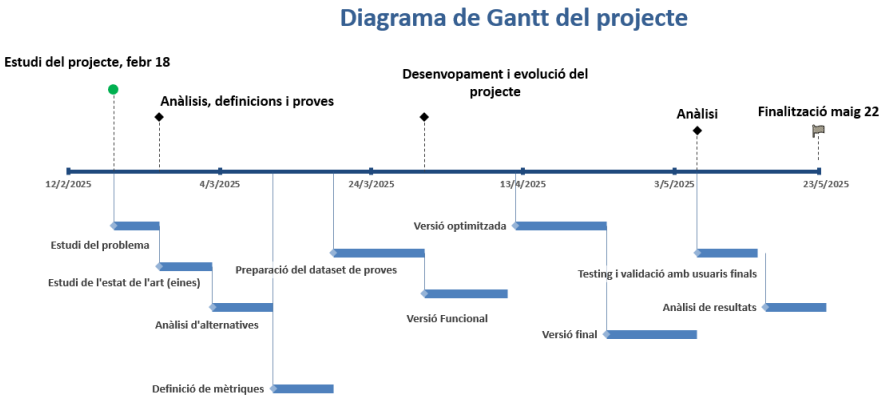

Fig.1: Diagrama de Gantt del projecte

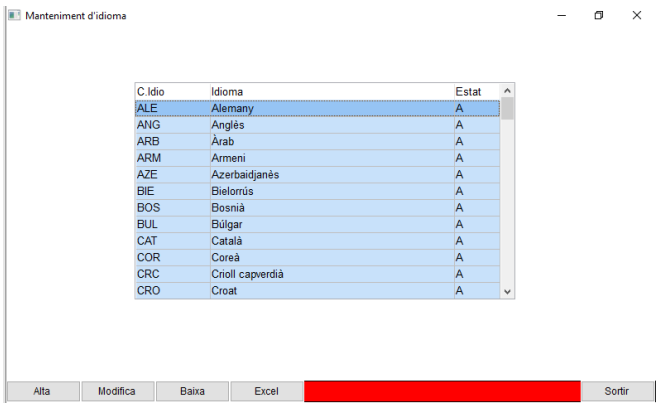

Fig. 5: Interfície de l'aplicació de Manteniment d'idiomes

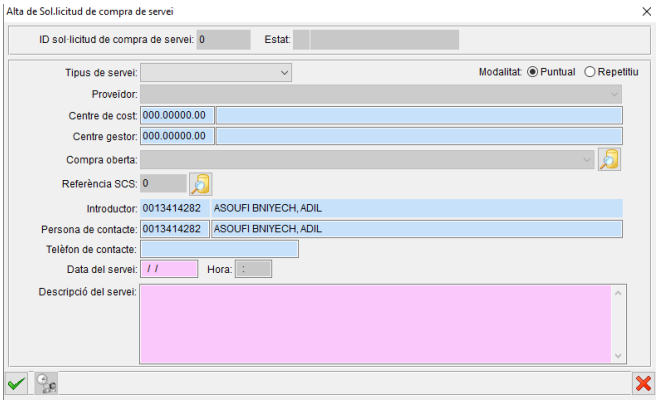

Fig. 7: Interfície de l'aplicació de Alta de Sol·licitud de compra de serveis

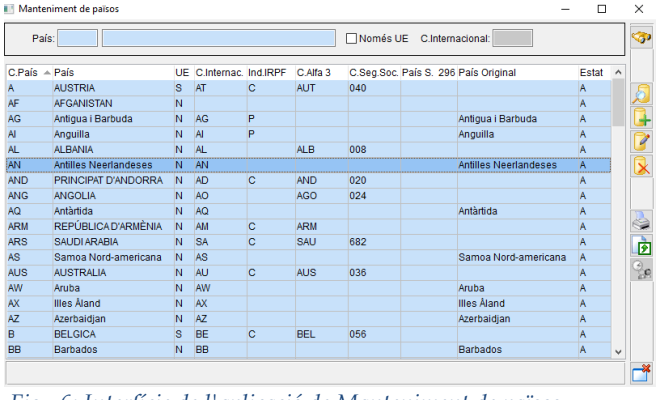

Fig. 6: Interfície de l'aplicació de Manteniment de països

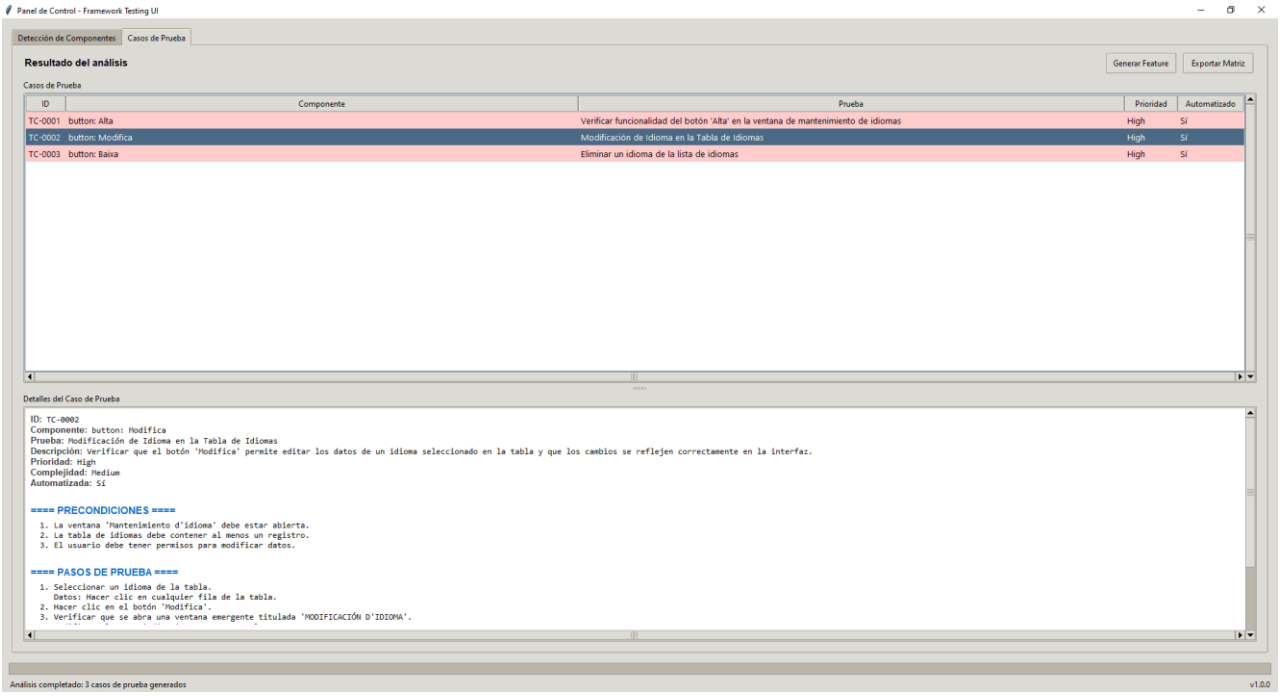

Fig. 10: Pantalla de casos de prova, on podem veure 3 casos de prova de l'aplicació del Manteniment D'idiomes desenvolupades i testejades.

```
REGLAS Y CRITERIOS PARA LA DETECCIÓN:
- Debes poner texto a los componentes que contengan solo un icono, relacionando el icono con una acción si es posible.
- Nunca devuelvas controles de ventana (minimizar, cerrar, maximizar) como componentes.
- Dos componentes diferentes no pueden coincidir en la misma posición ni en una posición muy cercana (menos de 10 píxeles de diferencia).
- Si hay componentes simplemente con iconos, sé muy preciso y describe su función.
- Ignora siempre los iconos de 'X' salvo que claramente sean de confirmación.
- Si hay un icono de tick ✓, márcalo como botón de confirmación si es único.
- Si hay filas de tabla, prioriza otros componentes más importantes.
- Si algún componente no tiene texto literal, ponle un nombre descriptivo básico.
- Devuelve solo los 18 componentes más importantes si hay demasiados.
- Sé exacto con las posiciones y la función de cada componente.
- No incluyas componentes decorativos ni informativos.
```

Fig. 8: Part del codi que conté part del prompt i regles que li enviem a la IA per que ens resolgui la crida per analitzar els components cor-

10.1 ENQUESTES

Enquesta feta a A.V.G. , professional de les aplicacions Progress Corporatives oferides per la CCMA, valorada de la següent manera:
Indica el teu grau d'acord amb cada afirmació utilitzant l'escala següent:

1 = Molt en desacord
2 = En desacord
3 = Neutral
4 = D'acord
5 = Molt d'acord

A.V.G. – Aplicacions corporatives
Aplicació de Manteniment d'Idiomes

Pregunta	Val.
1. El pla de proves cobreix la gran majoria o el 100% de les funcionalitats clau de l'aplicació.	5
2. El disseny visual de la matriu facilita la lectura i localització dels casos de prova.	5
3. Els diagrames de flux ajuden a entendre clarament els passos i resultats esperats.	5
4. Les prioritats assignades reflecteixen correctament la criticitat de cada cas.	5
5. Aquesta matriu seria útil com a base per a futurs manteniments d'aplicacions.	5

Aplicació de Manteniment de Països

Pregunta	Val.
1. El pla de proves inclou una cobertura adequada dels components d'interfície.	4
2. Les descripcions i l'estructura dels casos estan ben organitzades.	5
3. Els diagrames de flux representen correctament la lògica.	5
4. La matriu permet identificar fàcilment les funcionalitats testejades.	5
5. El disseny de la matriu és intuïtiu.	5

Aplicació de Sol·licitud de Compra de Servei

Pregunta	Val.
1. Contempla tots els camps obligatoris i validacions esperades.	1
2. Els diagrames de flux complementen les descripcions.	1
3. Cobertura adequada dels tipus d'entrada de l'usuari.	1
4. Els passos segueixen la funcionalitat esperada.	1

Enquesta feta a C.G.C., professional de l'equip de QA de la CCMA, Facilitat d'Ús i Aprenentatge,I **D.D.L., professional de l'equip de QA de l'empresa Hiberus, externa subcontractada per la CCMA, valorada seguint el mateix format:**

Pregunta	Val. C.G.C	Val. D.D.L
1. Instal·lació i configuració inicial ben documentada.	5	4
2. Interfície intuïtiva i fàcil de navegar.	4	5
3. Funcionalitats principals accessibles.	5	5
4. Flux per crear i executar proves clar.	5	5
5. Corba d'aprenentatge raonable.	5	5

Utilitat i Cobertura

Pregunta	Val. C.G.C	Val. D.D.L
1. Cobreix les necessitats habituals de testing.	5	3
2. Detecció automàtica fiable i precisa.	4	4
3. Generació de proves automàtica útil.	5	5
4. Informes útils per seguiment.	5	4
5. Adaptable a diferents aplicacions i escenaris.	5	4

Documentació

Pregunta	Val. C.G.C	Val. D.D.L
1. Documentació clara i completa.	5	4
2. Exemples i tutorials útils.	5	4

Integració i Automatització

Pregunta	Val. C.G.C	Val. D.D.L
1. Integració amb eines CI/CD.	5	5
2. Generació de fitxers .feature senzilla.	5	5
3. Reducció del temps i esforç manual.	5	5

Confiança i Fiabilitat

Pregunta	Val. C.G.C	Val D.D.L
1. Resultats i informes fiables.	3	5
2. Bona gestió de la usabilitat i avisos.	5	4