Sistema automàtic de correcció d'exàmens de programació

Jordi Domingo Capellà

Resum — En aquest document es presenta l'anàlisi, el disseny i el desenvolupament d'un programa que té el propòsit de permetre als professors corregir qualsevol tipus d'exàmens de programació. L'objectiu principal és ajudar al professorat a poder avaluar els estudiants d'una manera més ràpida i eficaç, a partir d'una eina que ofereix una visió global de l'examen i indica, de manera precisa, on es troba l'error, quin tipus d'error és i perquè està passant. D'igual manera el programa permet al professor poder enviar missatges d'ajut als estudiants per tal que sàpiguen en que s'estan equivocant. Aquest programa és de fàcil desplegament gràcies a que està pensat per desplegar-lo en el portal Caronte, a la part del Laboratori Virtual de Programació (VPL).

Paraules clau — Programa, Feedback, correcció, codi, implementació, Moodle, Caronte, proves.

Abstract — This paper presents the analysis, design, and development of a program that is intended to allow teachers to correct any type of programming exam. The main goal is to help teachers assess students more quickly and effectively, from a tool that provides an overview of the exam and indicates, precisely, where the error is. , what kind of error it is and why it is happening. Similarly, the program allows the teacher to send messages of help to students so that they know what they are doing wrong. This program is easy to deploy thanks to those that are designed to be deployed on the Caronte portal, in the part of the Virtual Programming Laboratory (VPL).

Index Terms — Program, Feedback, correction, code, implementation, Moodle, Caronte, tests.

1 Introducció - Context del treball

Per entendre millor de què va el projecte i com es farà el desenvolupament primer hem de conèixer quin serà el nostre entorn de treball i els programes que farem servir.
El projecte es realitzarà al caronte, que és l'eina de suport per a enginyeries de la UAB (Universitat Autònoma de Barcelona). Caronte està basat en Moodle.

Moodle és una plataforma d'aprenentatge dissenyada per proporcionar a educadors, administradors i estudiants un sistema integrat únic, robust i segur.

Moodle proporciona un conjunt d'eines centrades en l'estudiant i en els docents.

Proporciona una interfície simple, que facilitar el seu ús per a tots els seus usuaris.

Moodle és proporcionat gratuïtament com a programa de Codi Obert, sota- la Llicència Pública General GNU (GNU General Public License). Qualsevol persona pot adaptar, estendre o modificar Moodle.

La implementació de Moodle en codi obert significa que Moodle és contínuament revisat i millorat, per adequar-se a les necessitats actuals i canviants dels usuaris.

Moodle proporciona el conjunt d'eines més flexibles per suportar tant l'aprenentatge mixt (blended learning) com els cursos 100% en línia.

Com que és Codi Obert, Moodle pot ser personalitzat en qualsevol forma desitjada, per adequar-ho a necessitats individuals. La seva configuració modular i disseny interoperable permet als desenvolupadors crear plugins i integrar aplicacions externes per aconseguir funcionalitats

- *E-mail de contacte: Jordi.DomingoC@autonoma.cat*
- *Menció realitzada: Enginyeria del Software*
- *Treball tutoritzat per: Francesc Xavier Roca Marva (Departament de Ciències de la Computació)*
- *Curs 2021/22*

específiques.

Moodle es pot escalar per suportar les necessitats, tant de classes petites com de grans organitzacions. Això ens ofereix molta escalabilitat i llibertat de moviments a l'hora de realitzar certes implementacions per a un nombre elevat d'estudiants.

Compromès amb la seguretat de les dades i la privadesa de l'usuari, Moodle té controls de seguretat que són constantment actualitzats.

Està basat en web, per la qual cosa s'hi pot accedir des de qualsevol lloc del món. Amb una interfície per defecte compatible amb dispositius mòbils i compatibilitat creuada amb diferents navegadors d'Internet, el contingut a la plataforma Moodle és fàcilment accessible i consistent a l'amplada de diferents navegadors i dispositius.

A continuació trobem una de les extensions de Moodle, el VPL (Virtual Programming Lab), és una eina de codi obert que permet la gestió de pràctiques de programació a Moodle.

Les seves característiques d'edició, execució i avaluació de programes fan que el procés d'aprenentatge dels estudiants i la tasca d'avaluació dels professors siguin més fàcils que mai.

És gratuït i el seu codi està disponible a GitHub. [1]

2 OBJECTIUS

2.1 Objectiu Principal

L'objectiu principal d'aquest projecte és realitzar el desenvolupament d'un programa que permeti al professor corregir tota classe d'exercici d'un examen.

El que es vol fer és desenvolupar un programa general que ens permeti corregir un examen sense haver de fer modificacions per a cada exercici.

Dins d'aquest propòsit el que es vol és que es puguin dura terme dos tipus de proves.

Les proves de Caixa Negra, que només comprovaran si les dades d'entrada corresponen amb les dades de sortida i només s'oferirà informació respecte al resultat.

I també tindrem les proves de Caixa Blanca, les quals ens permetran tenir un cert control de l'execució d'aquell programa que l'alumne/a ha executat i saber en tot moment on està fallant i determinar quina informació al respecte se li pot retornar per tal que pugui dur a terme els canvis que consideri adients.

2.2 Objectius Específics

Per assolir aquest objectiu, s'ha dividit el projecte en diferents objectius:

1. Control dels paràmetres d'entrada al programa. Què se li passa, en quin moment i amb quina finalitat

2. Funció inicial que rep els paràmetres i fa la crida a les altres funcions.

3. Creació de funcions "branca", que el que fan es preparar les dades i determinar cap a on ens dirigirem a continuació. Faran la crida a les funcions finals que faran l'anàlisi determinat.

4. Creació de les funcions "fulla". Són aquelles funcions que ens permeten realitzar les correccions. Són les que reben els paràmetres ja adequats per tal de fer les comparacions i fer les proves de Caixa Blanca. En elles també trobem les funcions que fan les proves de Caixa Negra.

5. Associar tot el desenvolupament en un fitxer que ens permeti fer el desplegament al VPL.

6. Realitzar les proves corresponens a l'entorn Caronte, en el VPL amb exàmens reals de programació.

7. Millorar el rendiment per fer-lo el més òptim possible.

8. Facilitar l'ús del programa per al professorat que ho necessiti.

El propòsit final és completar tots aquests subobjectius per tal d'aconseguir un programa totalment funcional que ofereixi un servei eficient al professor que així ho necessiti.

3 METODOLOGIA I PLANIFICACIÓ

En aquesta secció s'exposa detalladament l'organització i la metodologia seguides per dur a terme el projecte.

Pel que fa a la metodologia de treball que s'ha dut a terme durant l'evolució del problema plantejat és l'Agile.

Aquesta metodologia ha estat triada, ja que m'hi sento bastant familiaritzat i també perquè 'Agile' és molt més que una metodologia per al desenvolupament de projectes que necessiten rapidesa i flexibilitat, és una filosofia que suposa una manera diferent de treballar i organitzar-se.

De manera que cada projecte es trosseja en petites parts que s'hi han de completar i lliurar en poques setmanes. La

finalitat és desenvolupar productes i serveis de qualitat que responguin a les necessitats les quals canvien a una gran velocitat.

Per tal de flexibilitzar el projecte de la millor manera possible, aquest s'ha dividit en 4 grans blocs. Aquests són:

- **Fase Preparació**: on s'ha definit el projecte que s'ha de realitzar i s'ha dut a terme una anàlisi preliminar de què suposarà realitzar-ho i quins detalls previs hem de tenir en consideració.
- **Fase Disseny**: s'han determinat els objectius, la planificació, els requeriments i el disseny del sistema.
- **Fase Desenvolupament**: s'ha desenvolupat tot el codi necessari per a dur a terme la implementació de l'eina de correcció.
- **Fase Tancament**: on s'ha preparat la documentació final i on es donarà per tancat el projecte.

Per cadascun d'aquests blocs s'ha repetit el mateix mètode de treball, que es basava a realitzar una planificació prèvia que s'havia de fer, la implementació, les proves i el test per determinar si el que s'havia fet era apte o no.

4 Estat de l'art

Per agafar certes referències pel que fa a què és el VPL, s'han buscat eines que facin una funció similar. Com és el cas de les que es llisten a continuació.

El Jutge és un sistema similar al caronte que és el que utilitzen a la Universitat Politècnica de Catalunya, en el qual els permet crear i corregir, de manera automàtica, exàmens o activitats que realitzaran els alumnes.

Online Exam Builder, es tracta d'una eina en línia molt senzilla. Amb aquesta plataforma, el docent pot generar l'examen i establir un límit de temps per a tot l'examen o per pregunta. També es pot decidir la data de començament i de finalització. Un cop finalitzat, es pot veure les estadístiques per usuari o obtenir una vista general del rendiment del grup.

QuestBase ens permet crear diferents tipus d'exàmens en línia i corregir-los automàticament. Dues de les característiques més importants és que podem incloure diferents tipus de preguntes, comentaris, imatges i sons als exàmens i que podem afegir un límit de temps per fer-ho.

Testmoz, aquesta és una altra eina en línia on els docents poden crear diferents tipus d'exàmens en línia per avaluar els estudiants. A més de corregir totes les proves de manera automàtica, el docent pot comprovar tota la informació relativa als encerts i errors, així com quant de temps ha trigat cada alumne a acabar l'examen.

iGiveTest, aquesta eina en línia ens permet crear exàmens en línia i analitzar els seus resultats de forma automàtica. Com a característiques fonamentals, podem assenyalar que ens permet fer qualsevol classe de pregunta - múltiple, veritable o fals i de desenvolupament- i que ens permet ajustar la puntuació de cadascuna. A més, es poden deixar comentaris a les respostes.

Gexcat, es tracta d'un programa potent que ens permet de manera ràpida tant dissenyar i corregir exàmens tipus test com generar exàmens de desenvolupament. Gexcat ofereix fins a nou models diferents d'examen. Només cal importar les preguntes a la base de dades, i el programa genera els exàmens i els corregeix de manera automàtica. Posteriorment, podem conèixer les dades estadístiques sobre les preguntes que més encerten i més fallen els estudiants.

Gnowledge és una altra eina que ens permet crear i corregir qüestionaris de manera automàtica. Els aspectes més destacats d'aquesta plataforma és que inclou estadístiques per mesurar el progrés acadèmic de l'alumnat i que podem conèixer quins alumnes han obtingut les millors qualificacions en un mateix test.

Google Forms, aquesta eina de formularis de Google també ens permet generar exàmens tipus test i corregir-los de manera automàtica. Com a funcions interessants, cal destacar que Google Forms ens permet conèixer en quina pregunta ha fallat un nombre més gran d'estudiants i ens ofereix una sèrie de gràfics amb respostes correctes, així com una mitjana de les puntuacions assolides.

5 Requisits del sistema

En aquest apartat es descriuen els requisits funcionals, aquells que indiquen com s'ha de comportar el sistema, i els no funcionals, aquells que especifiquen quines propietats té el sistema.

5.1 Requisits Funcionals

1. La introducció de dades per part del professor segueix una estructura determinada per tal de realitzar les mínimes modificacions possibles i amb l'objectiu d'oferir els millors resultats amb els mínims canvis.

- **RF-01**. El programa ha de permetre al professor introduir les dades necessàries a prtir d'un fitxer de configuració.
- **RF-02**. El programa ha de permetre al professor variar les dades d'entrada.
- **RF-03**. El programa/funció de l'estudiant no es pot tocar.

2. El programa ha d'oferir un codi fragmentat en diferents estructures per tal de separar totes les parts de les quals es vol fer una correcció.

- **RF-04**. El programa ha d'estar fragmentat en diferents estructures per a cada cas.

- **RF-05**. El programa ha de permetre un accés ràpid i intuïtiu al professor per tal de trobar què s'està fent en cada cas de correcció.

3. El programa ha de sedr accessible pel professor per tal que, en cas que es consideri necessari, es puguin realitzar les modificacions pertinents en cada funció.

- **RF-06**. El programa ha de permetre al professor realitzar les modificacions que consideri oportunes en cada funció de correcció.

- **RF-07**. El programa ha de permetre al professor poder afegir o eliminar funcions en els casos on sigui necessari, per tal de poder oferir a l'estudiant una millor versió de la seva correcció.

4. Traspàs de paràmetres fàcil

- **RF-08**. El programa ha de permetre que el traspàs de paràmetres entre funcions sigui clar i intuïtiu per si fos necessari realitzar qualsevol modificació.

- **RF-09**. El programa ha de guardar el resultat final de les execucions en una variable que es mostrarà al final de l'execució. Tant per part del que fa la funció de l'estudiant com pel que fa a la funció del professor.

5. El professor pot decidir que i com mostrar-ho als estudiants per tal d'oferir el màxim d'informació que es consideri en cada cas.

- **RF-10.** El programa ha de permetre al professor decidir quina informació mostrar de les execucions. Determinar si estem en un test públic o en un privat i quins missatges s'envien en cada cas.

6. Fitxer de configuració inicial.

- **RF-11**. El programa ha de permetre que es realitzi tota l'execució a partir d'un fitxer de configuració inicial on trobarem tots els paràmetres corresponents a cada execució del codi.

5.2 Requisits No Funcionals

7. Millorar el rendiment del programa.

- **RNF-01**. El programa ha de ser capaç d'oferir una resposta en menys de 5 segons d'execució com a màxim.

- **RNF-02.** Les dades de cada correcció han de ser actualitzades depenent dels paràmetres d'entrada.

8. Facilitar l'ús del programa per usuaris sense coneixements previs d'aquest.

- **RNF-03**. El programa ha de ser user friendly.

- **RNF-04**. El temps d'aprenentatge del programa ha de ser de menys de 10 minuts.

- **RNF-05.** El programa ha d'oferir una guia d'ús, comentaris dins del codi, per tal de fer més entenible que fa cada funció i que fa en global el programa.

- **RNF-06**. El desplegament del programa s'ha de poder fer de forma senzilla. En el nostre cas que només sigui enganxar les línis de codi en el fitxer corresponent dins del Caronte, a la part del Laboratori Virtual de Programació (VPL).

6 ARQUITECTURA

A continuació es detalla com és l'estructura interna del programa i s'explica com aquesta està implementada per complir amb els requisits del projecte.

Abans d'iniciar cap execució, el primer que hem de tenir en compte és la creació d'un fitxer de configuració en format JSON. El professor haurà de crear un fitxer de configuració per a fer els tests públics i un altre per tal de realitzar els tests privats.

En aquests fitxers JSON indicarem els inputs que volem tenir en la funció, els paràmetres que podem passar a la nostra funció i els outputs que esperem que ens retornin les funcions de l'estudiant i la del professor.

Cal remarcar que aquestes entrades del fitxer JSON de configuració són opcionals.

També el professor ha de proporcionar la solució a l'exercici que se li està demanant a l'estudiant per tal de saber quins són els resultats esperats i per poder fer la correcció.

Seguidament, el programa ha de permetre al professor llegir aquest fitxer de configuració i extreure tota la seva informació i iniciar l'execució un cop hem, guardat la informació pertinent i ja tenim tot el necessari per començar.

Per tenir una idea més clara de com seria l'arquitectura del programa tenim el següent diagrama (Figura 1) on

podem veure les diferents parts que té el programa, i veure de manera més definida l'arquitectura d'aquest.

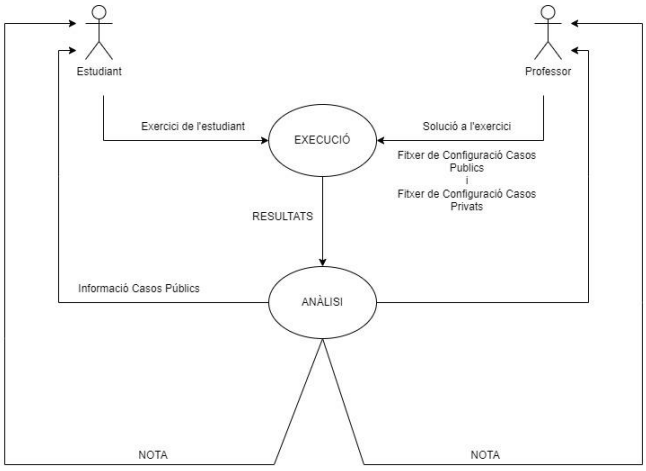

Figura 1. Diagrama que mostra com serà l'arquitectura del programa.

A l'hora de fer el disseny d'aquest programa, el primer que havíem de tenir clar és que el volíem fer molt modular, és a dir, molt dividit en diferents funcions i mòduls que ens permetessin, cada un d'ells, enfocar-nos en el que ens interessava en cada correcció.

Per això, el que s'ha fet és dividir el programa en una funció principal que inicia l'execució i a partir d'aquesta es van cridant les altres de forma seqüencial, com si es tractés d'una estructura en format arbre, on d'una branca en neixen més fulles i així seguidament.

Això es pot percebre més en el Diagrama de Casos d'Ús que trobem a l'Apèndix.

7 DESENVOLUPAMENT

A continuació es detalla com s'ha realitzat el procés de disseny i d'implementació del codi per a complir amb els requisits d'aquest projecte i assolir els objectius marcats.

S'ha utilitzat el llenguatge de programació Python. És un llenguatge d'alt nivell de programació interpretat la filosofia del qual fa èmfasi en la llegibilitat del seu codi, s'utilitza per desenvolupar aplicacions de tota mena, exemples: Instagram, Netflix, Spotify, Panda 3D, entre d'altres. Es tracta d'un llenguatge de programació multiparadigma, ja que suporta parcialment l'orientació a objectes, programació imperativa i, en menor mesura, programació funcional. És un llenguatge interpretat, dinàmic i multiplataforma. S'ha usat aquest llenguatge per garantir un dels principals requeriments, la correcció d'exàmens amb el llenguatge de programació Python.

A l'hora de dur a terme el disseny del programa, el primer que ens hem de preguntar és quins inputs havia de tenir el programa i com gestionar-los. S'ha determinat que al programa se li passaria un fitxer de configuració en format JSON, on s'indicaria tots els paràmetres i valors que se li passaran al programa. Això, ens permet realitzar les modificacions que es necessitin per a cada tipus de prova en aquest fitxer i no haver de modificar cap part del codi.

A continuació, ens hem de centrar en la manera que hem de tenir per gestionar aquests paràmetres que ens arriben.

Pel que fa als paràmetres generals, es guarden en una llista simple per tal de processar-los a continuació. També tenim els inputs, aquests serien els que simulen l'entrada per teclat per part de l'estudiant i es guarden en un fitxer per tal d'utilitzarlos mitjançant la funció input() més endavant. I per acabar tenim els outputs, que són les sortides esperades i guardades per tal de poder fer les comprovacions.

El primer que es fa és agafar aquests paràmetres i adequar-los per tal que la funció de l'estudiant els rebi sense cap problema. A partir d'aquí s'inicia el procés d'avaluació dels resultats.

Primer de tot obtenim els resultats que ens passa l'alumne i els guardem en una variable, si el programa de l'estudiant no té cap error que aturi l'execució, seguim endavant. Seguidament, es guarda en una variable els resultats esperats pel professor respecte a aquells paràmetres que hem rebut prèviament. Al tenir les dades del professor i de l'estudiant guardades en variables i retornades per la funció corresponent hem de fer una anàlisi sobre aquestes.

Per tal de fer aquest estudi dels resultats es van pensar diferents opcions i diferents camins per tal de poder acomplir aquesta tasca de la millor manera possible, però finalment s'ha decidit que el primer que s'ha de fer és comprovar el que retornen les dues funcions, la del professor i la de l'estudiant. Primerament, es comprova si el que retornen les funcions és una excepció. En cas de ser així més endavant haurem de determinar que farem i com seguirà el codi. Si no és una excepció, el que farem és mirar els resultats i comparar el que estem esperant amb el que se'ns ha passat.

Per la comparació de resultats el que fem primer és veure si aquestes dades que se'ns passen són iguals en els dos casos, en cas de ser així, l'execució ja estaria finalitzant, ja que obtindríem el que estem esperant. Si fos el cas contrari, el que es fa primer és determinar de quin tipus de dades estem parlant.

Com és evident, per a cada tipus de dades, la manera de tractar-les es diferent. Per explicar aquesta part anirem pas a pas.

Primer de tot el que hem de fer és conèixer quins són els errors més freqüents que es cometen per cada cas, per així poder determinar què és el que més ens interessa comprovar i de quina manera distribuirem la puntuació de

cara a l'última avaluació per tal de determinar la nota de l'estudiant.

En el primer dels casos, parlem que si la funció ens retorna un ['int', 'float', 'bool', 'str', 'complex'] el que fem per tractar aquest cas és mostrar-li a l'estudiant el tipus de dades que estem esperant i el que ens està retornant ell actualment.

La segona comprovació seria en el cas que el que se'ns està retornant és ['tuple', 'list', 'set'].

En el cas de les tuples el que fem és comprovar si les mides de les dues variables són iguals, i en cas de no ser-ho mostrem les diferències a l'estudiant. Seguidament, mirem si l'ordre en el qual s'han retornat les dades és el correcte o hi ha algun tipus d'error i en cas de ser així, es mostra a l'estudiant.

En el cas que el que se'ns retorna és una llista, el primer que fem és mirar les mides, com hem fet anteriorment en el cas de les tuples, en cas d'haver-hi alguna discordança, ho mostraríem a l'estudiant, i com hem fet també anteriorment comprovem si l'ordre de les dues llistes són iguals.

Pel cas dels 'sets', seguiríem una implementació igual que la de les tuples i la de les llistes. Primer mirem el tamany i posteriorment comprovem si l'ordre de les dues variables que estem rebent és l'esperat o no.

La tercera comprovació seria en el cas dels diccionaris ['dicts']. En aquest cas, el primer que fem és comprovar si les claus del diccionari són iguals. En cas afirmatiu, comprovem si en aquella clau els valors no tenen cap discordança.

Si les claus són diferents, ja tenim l'error i el que mostrem a l'estudiant és quin és el resultat esperat i quines són les claus que falten o sobren en el seu resultat.

En el cas que el tipus de dada que hem rebut sigui una classe el primer que farem serà comprovar els valors dels atributs i comparar-los amb els valors dels atributs que serien els esperats per tal de fer una primera correcció. En cas que aquesta primera comparació fos negativa s'enviaria un missatge a l'estudiant d'en quin punt els seus valors estan fallant.

En cas que la classe de l'estudiant retorni uns valors dels atributs iguals als esperats pel professor passaríem al següent nivell de correcció, on comprovarem si tots els mètodes de la classe han sigut implementats i els comparerem amb els esperats pel professor. Igual que anteriorment, en cas de trobar qualsevol classe d'error en aquesta comprovació se li enviarà un missatge a l'estudiant per tal que ho pugui retocar.

Un cop fetes aquestes dues primeres comprovacions, en cas de ser correctes i de no trobar cap error, començaríem a analitzar els mètodes de les classes. Per això, hi ha un fitxer en format JSON, que ens indica els mètodes que volem comprovar i els paràmetres amb els quals ho farem.

Primer llegim el mètode que es vol corregir del fitxer JSON i comprovem que es troba en la classe creada per l'estudiant i també en la classe creada pel professor.

Seguidament, se separen els valors i els tipus dels paràmetres en dues llistes diferents. Un cop aquí comprovem que el que se'ns retorna és del mateix tipus que la classe principal.

A partir d'aquí, el que fem és una nova comprovació de resultats per veure si es corresponen i tenen el mateix tipus.

Si és així, el que farà el programa és veure que es tracta de dues classes i mirarà quins son els seus atributs.

En cas de no ser així, no es tractarà de dues classes i el que farà el programa és una anàlisi de resultats normal, comprovant que es tracta i corregint-ho de la manera que s'hagi determinat per a aquell tipus.

Un cop vist com tractarem els resultats en cas de ser correctes o incorrectes en els diferents tipus de dades que tenim en compte, hem de mirar que fem si el que passa és que es genera una excepció.

En cas de ser així, el primer que fem és veure si aquesta excepció és esperada o no. Ja que es pot donar el cas que el que vulguem és que el codi fet a la funció de l'estudiant doni una excepció esperada.

Per això el que fem és veure si l'excepció que ha retornat la funció de l'estudiant és una excepció esperada comparant els resultats obtinguts amb els esperats.

En cas de no tractar-se d'una excepció esperada la cosa es complica en certa manera, ja que existeixen una gran quantitat d'excepcions que poden ocórrer. Per això el que es fa és, depenent de cada cas, si les excepcions coincideixen o no i si els missatges que s'envien en cas d'excepció son els mateixos.

Tota aquesta informació també es retorna a la funció principal per tal de tenir el màxim d'ajuda possible de cara a l'estudiant.

Un cop tota aquesta informació ha sigut recopilada, l'estudiant rep tots els missatges amb la màxima informació possible per tal d'ajudar-lo a millorar el seu codi. Tota aquesta informació pot ser amagada pel professor en cas de creure-ho necessari.

Un cop tota l'execució de la funció principal ha finalitzat, el que es fa és mostrar per pantalla, els resultats que s'han obtingut i els resultats que serien els esperats.

A part del descrit anteriorment, a cada iteració de la funció principal, depenent del nombre de paràmetres que s'hagin passat en el fitxer de configuració, es va mostrant tota la informació de les dades que s'estan tractant.

Per tal de mostrar en certa mesura, de manera més visual, com seria l'execució del codi i quins passos es van seguint, mostrem el següent diagrama de flux que ens permet entendre una mica més la lògica del sistema:

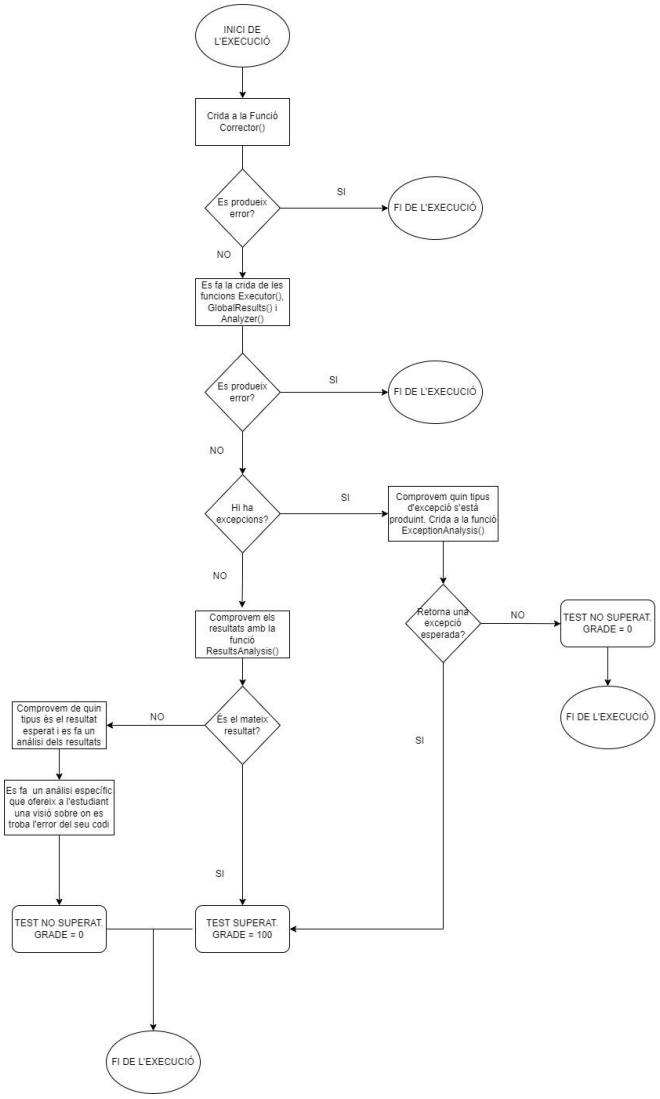

Figura 2. Diagrama de flux que mostra el funcionament del programa.

8 Fase de Proves.

En aquesta secció es descriuen les proves que s'han realitzat per comprovar que les funcionalitats de l'aplicació funcionessin com s'esperava.

Primer de tot es feien els tests anomenats Exploratory Testing, amb els quals es tracta d'anar realitzant test a mesura que s'explora el programa i experimentar els errors o comportaments inusuals durant la seva execució.

Aquest tipus de test s'ha anat fent a mesura que s'avançava en el desenvolupament del programa, ja que s'anaven comprovant que les funcionalitats programades funcionessin correctament.

Finalment, quan ja es tenia el programa enllestit es va realitzar una exploració profunda per a detectar possibles errors o bugs que es poguessin escapar i no s'haguessin detectat fins ara.

Seguidament, un cop el desenvolupament estava finalitzat, es van començar a realitzar les proves al Caronte, amb un examen de veritat, i allà és on anàvem a comprovar si el codi era funcional o no.

La idea de les proves en el Caronte era agafar un examen on poguéssim trobar diferents tipus d'exercici, per tal de provar el programa en el màxim de casos possibles i veure si els resultats eren els esperats.

A continuació s'exposen les proves fetes a l'examen, mostrant els exercicis amb els quals s'han dut a terme les comprovacions.

Cal remarcar que en cada exercici trobarem l'arquitectura del programa. Primer de tot el programa de correcció, a continuació trobarem els fitxers de configuració públic i privat i l'exercici realitzat de manera correcta fet pel professor.

Posteriorment, les proves es realitzaran fent l'exercici de l'estudiant buscant mostrar tots els casos que ens podem trobar en un exercici. Fent modificacions per tal de tenir un exercici correcte, exercicis amb diferents errors i en casa de ser possible buscarem crear excepcions per tal de poder veure també com es controlen aquests casos.

8.1 Fase de proves en un Examen

8.1.1 Exercici 1

Aquest exercici el que es demana a l'estudiant és llegir d'un fitxer uns valors i guardar el contingut del fitxer en un diccionari.

Per tal de fer les comprovacions primer de tot s'ha indicat en els fitxers de configuració quins casos es volen comprovar.

Per començar fem la prova amb un codi de l'estudiant igual a l'esperat pel professor per tal de comprovar que tot funciona correctament en cas que els codis sigui correctes.

Com es pot veure la informació en aquest cas es centra en mostrar les dos sortides, la esperada pel professor i la que ha assolit l'estudiant amb el seu programa, indicant que el resultat és correcte. També es mostra que, en aquest cas, s'han superat tots els tipus de tests que s'han realitzat.

Per comprovar ara els casos incorrectes. El que farem serà modificar la funció de l'estudiant.

Primer de tot modifiquem les claus del diccionari per a que siguin diferents a les esperades, i així podem veure el missatge que s'envia a l'estudiant en aquest cas.

Ara per fer una altra comprovació dels diccionaris, farem que els valors siguin diferents.

Com anteriorment, tenim un missatge d'error que ens diun en que estem fallant, i quin seria el resultat esperat i el que està retornant l'estudiant.

En cas que el problema sigui una excepció, el programa també envia un missatge.

Per veure els resultats d'aquestes proves: Apèndix. A1.1 Exercici 1.

8.1.2 Exercici 2

En el segon exercici seguim treballant amb diccionaris. En aquest cas també hem de llegir els valors d'un fitxer extern i guardar-los en un diccionari.

Com hem fet en l'exercici anterior, el que farem és comprovar com es corregirà l'exercici en cas que la funció de l'estudiant retorni els valors esperats i que passaria en cas que hi haguessin errors.

Al tractar-se que el resultat és un diccionari, les comprovacions fetes són molt similars a les fetes a l'anterior exercici.

Per veure els resultats d'aquestes proves: Apèndix. A1.2 Exercici 2.

8.1.3 Exercici 3

En el tercer exercici de l'examen, el que es buscava era comprovar si un input es corresponia amb una lletra majúscula o no.

Aquests inputs són els que s'han passat en el fitxer de configuració a l'inici de l'exercici.

Per tal de fer les comprovacions, es passen diferents tipus de caràcters a la funció per tal de veure si el funcionament és l'esperat o no.

Per veure els resultats d'aquestes proves: Apèndix. A1.3 Exercici 3.

8.1.4 Exercici 4

A l'exercici número 4, el que es demana és realitzar una puntuació depenent del nombre de lletres que té una paraula en concret.

Básicament el que fèiem era anar modificant els inputs i les puntuacions per veure si les correccions eren correctes o no.

Per veure els resultats d'aquestes proves: Apèndix. A1.4 Exercici 4.

8.1.5 Exercici 5

En el següent exercici, el que es demana és rebre un diccionari i una llista. En ells trobem uns valors i si la clau del diccionari pertany a un dels valors de la llista es realitza una suma del primer valor d'aquelles claus que estiguin dins de la llista.

Per tal de fer les proves, aquí buscàvem aspectes similars als primers exercicis. Modificar les claus i els valors i buscar punts on se'ns permetés trobar alguna excepció.

Per veure els resultats d'aquestes proves: Apèndix. A1.5 Exercici 5.

8.1.6 Exercici 6

A l'exercici 6, trobem un cas similar a l'anterior, rebem un diccionari. A partir d'aquest, el que ha de fer l'estudiant és crear un nou diccionari, en aquest nou diccionari, els valors de l'anterior passaran a ser les claus del nou, i les claus passaran a ser els valors.

En aquest exercici, seguíem fent els mateixos casos per fer les correccions, però en una de les funcions vam aconseguir trobar un excepció i així veure quin missatge se li mostraria a l'estudiant.

Per veure els resultats d'aquestes proves: Apèndix. A1.6 Exercici 6.

8.1.7 Exercici 7

En aquest exercici número 7, el que es busca és utilitzar els inputs per tal de simular l'execució d'un programa on l'usuari introduiria uns valors per tal d'obtenir una puntuació.

A partir d'aquests inputs, l'estudiant ha de fer el càlcul de la puntuació per tal d'oferir una resposta.

Per fer les proves, anàvem buscant inputs que ens permetessin portar el codi als seus límits, però tot va anar com s'esperava i els resultats tant a l'hora de la correcció com al trobar excepcions ens mostraven el que volíem veure.

Per veure els resultats d'aquestes proves: Apèndix. A1.7 Exercici 7.

8.1.8 Exercici 8

En aquest útlim exercici, el que se li demana a l'estudiant és implementar una classe amb uns certs mètodes. Cada funció haurà de rebre uns paràmetres i retornaran uns resultats.

El primer que s'ha de mirar a l'hora de fer la correcció d'un exercici d'aquest estil és comprovar si es tracta d'una classe o no i, si tot és correcte, es procedirà a realitzar l'anàlisi de les classes, passant pels seus mètodes i resultats.

D'entrada, realitzem l'exercici de la manera correcta per tal d'obtenir el feedback que rebria l'estudiant en aquest cas.

Tot seguit, procediríem a fer certes variacions en el programa de l'estudiant per tal d'obtenir els errors més típics en aquest tipus de casos.

D'inici, modifiquem la definició, perquè no es tracti d'una classe i ens retorni un altre valor que faci que no es desenvolupi el correcte anàlisi d'aquesta part.

Posteriorment, variarem els resultats dels mètodes per tal que el programa ens mostri on s'està produint l'error i proporcioni aquesta informació de retorn per a l'alumne.

Per veure els resultats d'aquestes proves: Apèndix. A1.8 Exercici 8.

9 Anàlisi i Discussió dels Resultats

Un cop realitzades les proves en un examen real, podem confirmar que els resultats són força positius. Tenim un programa que ens permet, només modificant els fitxers de configuració inicials per als test públics i els privats, corregir qualsevol exercici. Ja es tracti de treballar amb diccionaris, amb caràcters, amb inputs o amb Classes, tots els resultats han estat satisfactoris.

La major part dels problemes que han anat sortint a l'hora de fer les proves al Caronte, han estat amb els fitxers, ja que era necessari fer modificacions en el que s'havia de tenir en compte durant l'execució. Ja que, de no fer-ho així, els fitxers de configuració no apareixien.

També s'ha hagut de tenir en compte el fet que qualsevol error de codificació que apareixia, es podia comprovar a la part del debugador en el software extern que he utilitzat i que, per tant, facilitava molt la feina en aquest aspecte.

Cal destacar també la part de l'Exploratory Testing, on es van fer les comprovacions pas a pas de tots els tipus de dades amb les que podem treballar. Com seria el cas de les llistes, les tuples, els sets, els flotants i els enters, que no han aparegut de manera concreta a les proves fetes a l'examen.

En aquesta part, utilitzant el debugador, hem anat corregint qualsevol tipus d'error que anéssim trobant en les execucions, igual que aplicant totes les millores que anaven sortint. Hi ha hagut molt moments de proves on les coses no acabaven de sortir com es buscava, i s'han cercat moltes alternatives per tal d'aplicar aquella idea inicial, d'una manera totalment diferent.

A l'hora d'analitzar els resultats obtinguts, veiem que quan la funció de l'estudiant és correcta i obté els valors esperats, es mostra un missatge amb aquesta informació, que ajuda a l'estudiant a saber que el que ha fet està bé.

En cas que no sigui així, hi ha diferents tipus d'error, cada funció és diferent i per tant s'ha de tractar de manera diferent. Cada correcció té els seus missatges predeterminats i específics per a aquell tipus de valors, donant la informació més útil possible per a l'estudiant.

Pel que fa al desenvolupament del codi, ha sigut un treball més dur del que havia plantejat inicialment. La planificació va ser errònia, i els temps s'han anat allargant a mesura que més s'aprofundia en les funcions de correcció.

A més de la planificació, a l'hora de determinar l'estructura final del programa també hi van haver diferents modificacions. El mètode inicial amb el que es rebien els paràmetres no ens acabava de convèncer i al final es va decidir fer-ho tot a partir d'un fitxer de configuració en format JSON, que va obligar a canviar bastant codi per tal d'adaptar-lo als canvis.

També, un problema bàsic a l'inici i pel que no s'ha pogut realitzar les fases del projecte en el temps que estava pensat des d'un inici, ha estat la necessitat de familiaritzar-me de nou amb el llenguatge Python, que de ben segur és bastant senzill, però per intentar exprémer tot el seu potencial s'ha de fer molta recerca i no parar de codificar per anar millorant.

10 Conclusions

Finalment, cal esmentar que gràcies a la metodologia i a la planificació establerta ha estat possible assolir els objectius que es van plantejar a l'inici del projecte.

El principal objectiu era la creació d'un programa polivalent, que ens permetés fer la correcció de qualsevol exercici d'un examen, realitzant tests de Caixa Negra i tests de Caixa Blanca i oferís un feedback a l'estudiant sobre el seu codi, brindant la informació adient per tal d'ajudar-lo a realitzar les modificacions necessàries per obtenir els resultats esperats.

La divisió de l'objectiu principal en subobjectius, m'ha permès seguir una estructura coherent en l'execució del projecte. Aquests han propiciat que la creació de

l'arquitectura, el disseny del programa, el desenvolupament i les proves s'hagin implementat de manera lògica, aplicant els coneixements adquirits al llarg del grau.

A raó d'això, es posa de manifest el compliment dels objectius proposats, fet que comporta que els resultats obtinguts siguin força positius.

Tot i això, el projecte no està exempt de millores. A nivell general, la planificació podria haver estat més encertada pel que fa a les fases i la temporització de cada una d'elles. Com a aspecte més específic, a l'hora de fer l'anàlisi de la funció que enviava l'estudiant, es podria haver aprofundit més, tenint en compte no només els resultats obtinguts sinó també la manera en la que s'ha implementat el codi, analitzant si era la més correcta i/o òptima per fer-ho.

La posada en pràctica d'aquest projecte m'ha permès aconseguir una visió clara i concloent del que és dur a terme un desenvolupament d'aquestes característiques. Gràcies a la recerca d'informació, he après nous conceptes que desconeixia i he pogut constatar la seva aplicabilitat en un cas real.

En última instància i fent referència al que el projecte m'ha aportat com a futur enginyer, cal destacar la importància d'estar en constant aprenentatge per tal d'anar adquirint noves competències. Tanmateix, m'ha permès obrir la mirada a l'hora de ser conscient els errors i buscar la manera de resoldre'ls, acceptant les crítiques per anar creixent com a enginyer.

A títol personal, considero que he après a millorar certs hàbits de treball, a tenir en compte que no sempre els objectius s'aconsegueixen a la primera i a gestionar millor la frustració.

En definitiva, m'emporto aspectes molt positius d'aquest treball, que ben segur em serviran per a la meva imminent entrada al món laboral.

Agraïments

Per començar, m'agradaria agrair al Xavier, per haver-me fet costat en tot moment, per confiar sempre en mi i per saber donar-me aquell toc d'atenció per seguir millorant i treure la millor versió de mi.

Voldria donar les gràcies a la meva família, als meus pares i als meus germans, per estar des de sempre i per sempre.

Als meus amics que he fet durant aquests anys, sense ells, tot aquest procés no hauria estat el mateix.

I per acabar a la Marta, per ser sempre la millor part de mi, per tot el que em recolza i per sempre estar al meu costat.

Bibliografia

[1] *VPL - Virtual Programming Lab - Home*. (2022). VPL. Recuperado 25 de febrero de 2022, de https://vpl.dis.ulpgc.es/

[2] *Moodle VPL Tutorials - dftwiki3*. (2022). Moodle VPL Tutorials. Recuperado 25 de febrero de 2022, de http://dominiquethiebaut.com/dftwiki3/index.php/Moodle_VPL_Tutorials

[3] *Ejemplo de un VPL Python - Documentación Campus Virtual de la UEx*. (2022). VPL Python. Recuperado 25 de febrero de 2022, de https://campusvirtual.unex.es/docs/index.php?title=Ejemplo_de_un_VPL_Python

[4] B, M., D, L., K, P., S, S., & M, S. (2019). An Online Learning Platform for Teaching, Learning and Assessment of Programming. *International Research Journal of Multidisciplinary Technovation*, 90–95. https://doi.org/10.34256/irjmt19213

[5] Cardoso, M., Marques, R., Castro, A. V., & Rocha, L. (2020). Using Virtual Programming Lab to improve learning programming: The case of Algorithms and Programming. *Expert Systems*, *38*(4).

[6] *PyCharm: the Python IDE for Professional Developers by*. (2021, 2 junio). JetBrains. Recuperado 25 de febrero de 2022, de https://www.jetbrains.com/pycharm/

[7] *Anaconda | Anaconda Distribution*. (2022). Anaconda. Recuperado 25 de febrero de 2022, de https://www.anaconda.com/products/distribution

[8] *Python JSON*. (2022). W3 Schools. Recuperado 25 de febrero de 2022, de https://www.w3schools.com/python/python_json.asp

[9] Robert McNeel & Associates. (2022). *How to use JSON*. Www.Rhino3d.Com. Recuperado 25 de febrero de 2022, de https://developer.rhino3d.com/guides/rhinopython/python-xml-json/#:%7E:text=Use%20the%20import%20function%20to%20import%20the%20JSON%20module.&text=The%20JSON%20module%20is%20mainly,be%20written%20into%20a%20file.&text=While%20the%20JSON%20module%20will,write%20directly%20from%20JSON%20files

[10] *Python os Module*. (2022). TutorialsTeacher. Recuperado 25 de febrero de 2022, de https://www.tutorialsteacher.com/python/os-module

[11] *os — Miscellaneous operating system interfaces — Python 3.10.5 documentation*. (2022). os Documentation. Recuperado 25 de febrero de 2022, de https://docs.python.org/3/library/os.html

[12] Real Python. (2022, 18 marzo). *Write Pythonic and Clean Code With namedtuple*. Write Pythonic. Recuperado 25 de febrero de 2022, de https://realpython.com/python-namedtuple/

[13] GeeksforGeeks. (2020, 23 julio). *Python Traceback*. Recuperado 25 de febrero de 2022, de https://www.geeksforgeeks.org/python-traceback/

11 APÈNDIX

A1. Exercicis de les proves a l'examen

A1.1 Exercici 1

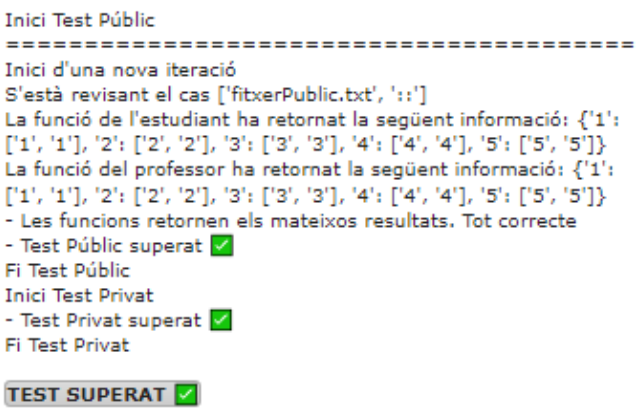

Figura 3: Execució correcta del codi de l'estudiant.

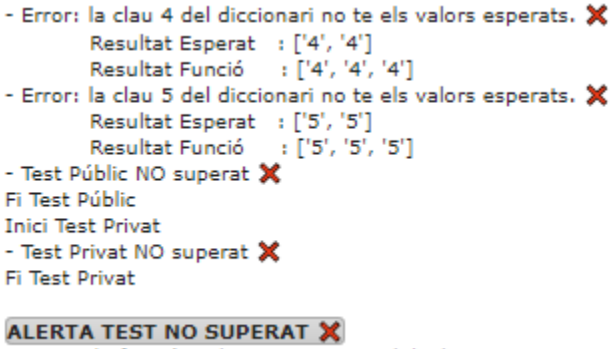

Figura 4: Execució amb errors del codi de l'estudiant.

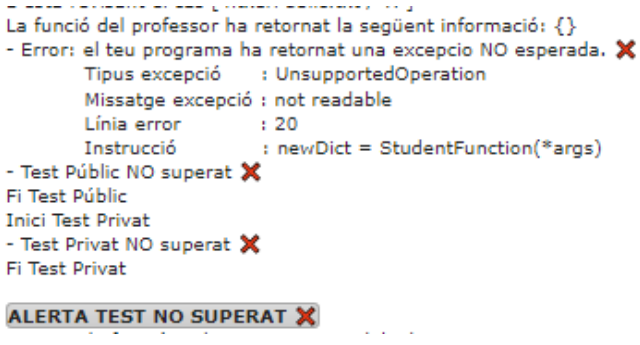

Figura 5: Execució amb una exepció.

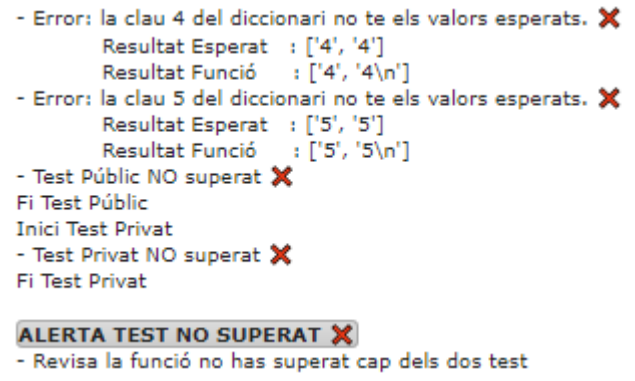

Figura 6. Execució amb errors del codi de l'estudiant.

A1.2 Exercici 2

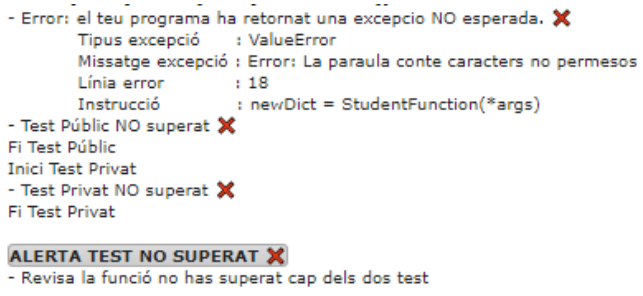

Figura 7. Execució amb errors del codi de l'estudiant.

A1.3 Exercici 3

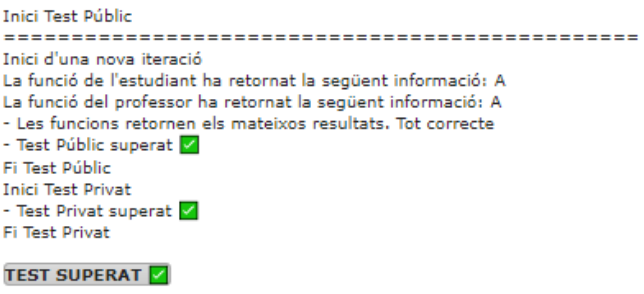

Figura 8: Execució correcta del codi de l'estudiant.

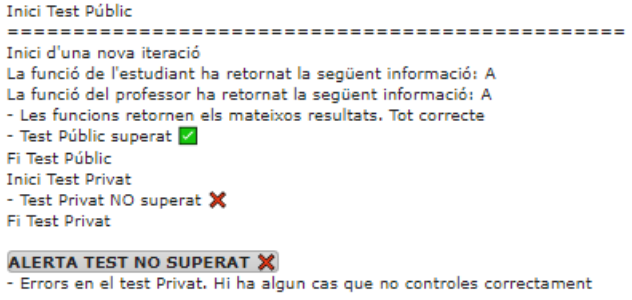

Figura 9: Execució amb errors del codi de l'estudiant.

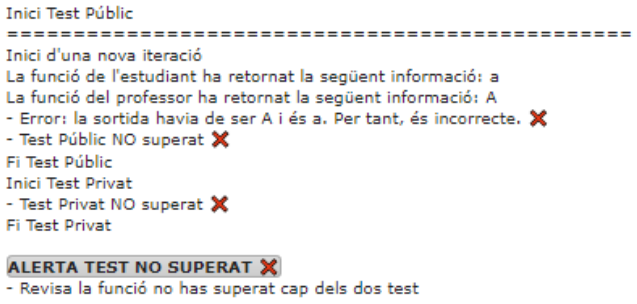

Figura 10: Execució amb errors del codi de l'estudiant.

A1.4 Exercici 4

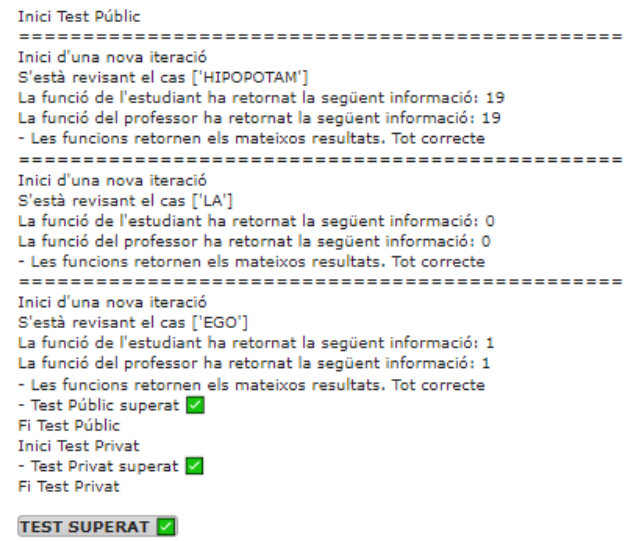

Figura 11: Execució correcta del codi de l'estudiant.

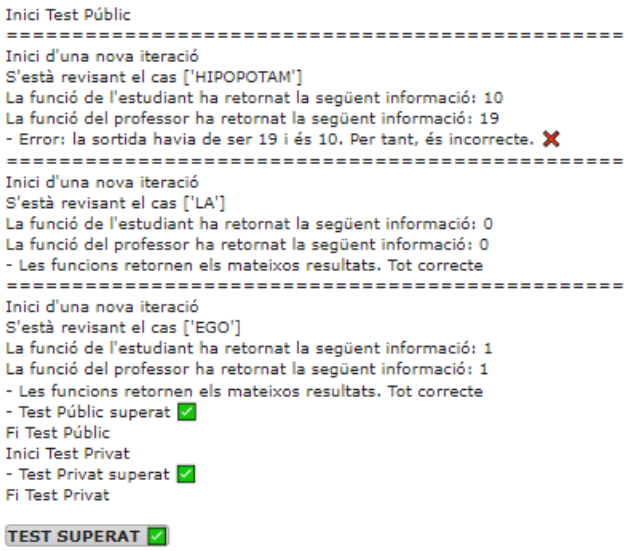

Figura 12: Execució amb errors del codi de l'estudiant.

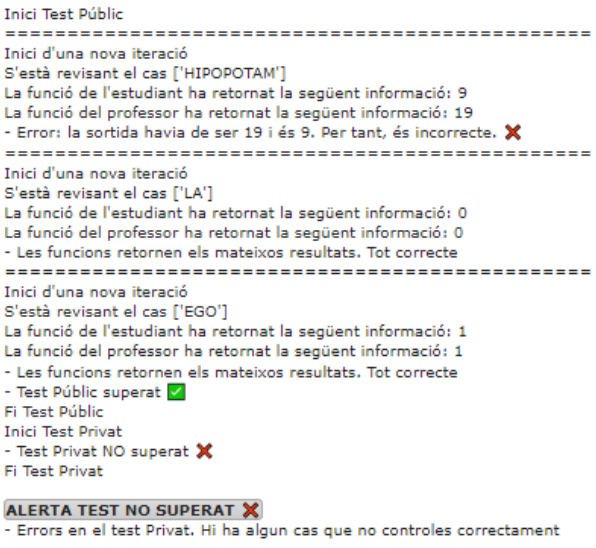

Figura 13: Execució amb errors del codi de l'estudiant.

A1.5 Exercici 5

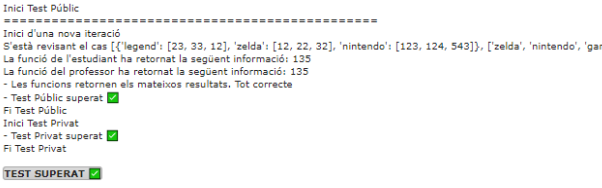

Figura 14: Execució correcta del codi de l'estudiant.

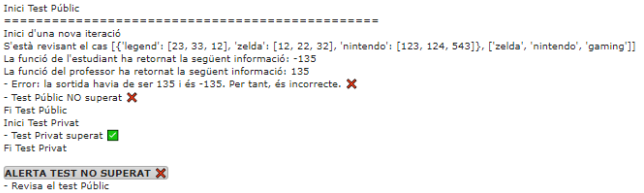

Figura 15: Execució amb errors del codi de l'estudiant.

A1.6 Exercici 6

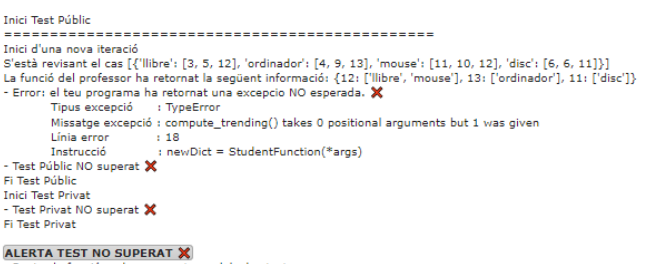

Figura 16: Execució amb una exepció.

A1.7 Exercici 7

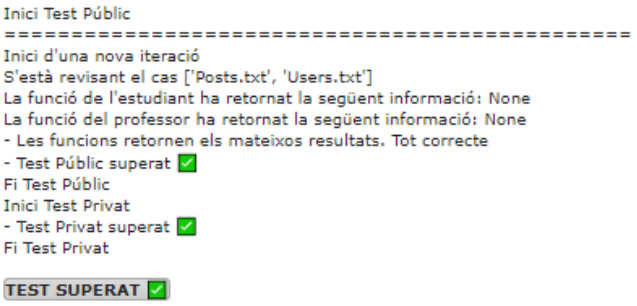

Figura 17: Execució correcta del codi de l'estudiant.

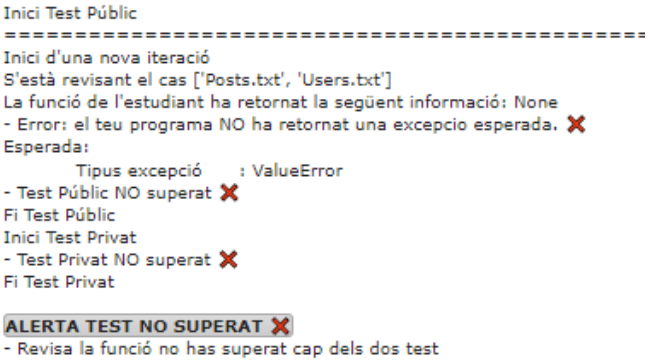

Figura 18: Execució amb una exepció.

A1.8 Exercici 8

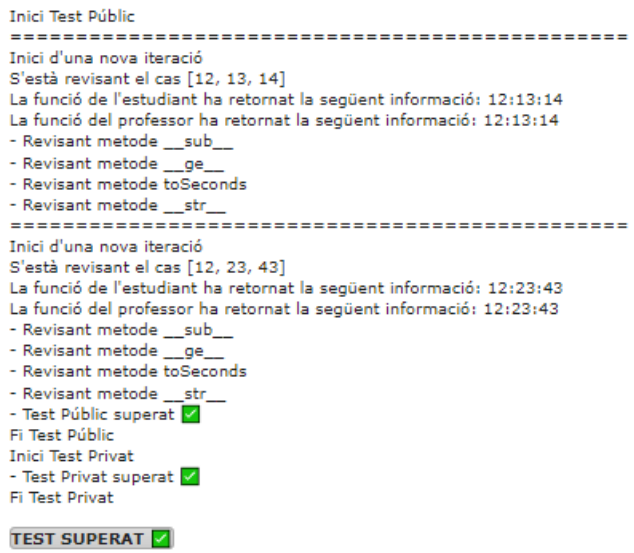

Figura 19: Execució correcta del codi de l'estudiant.

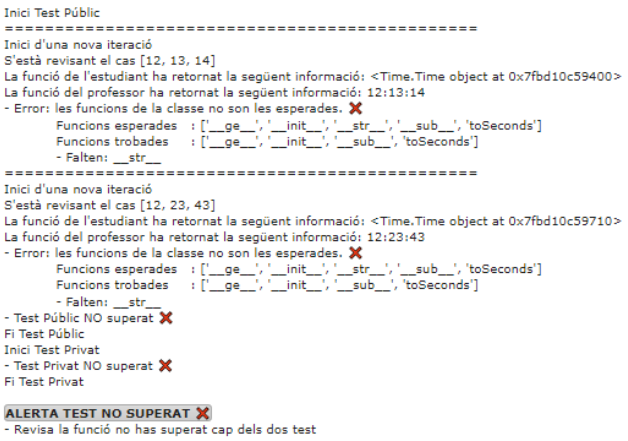

Figura 20: Execució amb errors del codi de l'estudiant.

A2. Diagrama de Casos d'Ús