Escola Universitària
Politècnica de Mataró

Centre adscrit a:

UNIVERSITAT POLITÈCNICA
DE CATALUNYA

GRAU EN ENGINYERIA INFORMÀTICA

COMPUTACIÓ DISTRIBUÏDA SOBRE DISPOSITIUS MÒBILS

Memòria

JOSE CARLOS MARUGAN CALLES
PONENT: DR. ENRIC SESA NOGUERAS

PRIMAVERA 2015

TecnoCampus
Mataró-Maresme

Dedicatòria

A la meva dona Angie, la meva filla Noa i el meu fill Nil, entre tots ho heu fet possible.

Agraïments

A la meva família pel seu recolzament incondicional i la seva paciència.

A tos els meus companys i en especial al Dani i al Roger pel seu suport.

A tots els meus professors i en especial als doctors Enric Sesa Nogueras i Josep Roure Alcobé per orientar-me i fer-me de guia.

Resum

En aquest projecte, s'ha desenvolupat una plataforma que ajuda a la paral·lelització d'aplicacions utilitzant dispositius mòbils com a nodes o processadors. Per al seu desenvolupament, el projecte s'ha dividit en tres subprojectes (client, servidor i proveïdor). La plataforma de computació distribuïda i paral·lelització es compon dels subprojectes client i servidor. El subprojecte proveïdor s'ha desenvolupat per utilitzar la plataforma de computació distribuïda creada, i concretament, s'ha desenvolupat per resoldre una tasca pròpia del camp de la Bioinformática, l'alineació de seqüències d'ADN.

Resumen

En este proyecto, se ha desarrollado una plataforma que ayuda a la paralelización de aplicaciones utilizando dispositivos móviles como nodos o procesadores. Para su desarrollo, el proyecto se ha dividido en tres subproyectos (cliente, servidor y proveedor). La plataforma de computación distribuida y paralelización se compone de los subproyectos cliente y servidor. El subproyecto proveedor se ha desarrollado para utilizar la plataforma de computación distribuida creada, y concretamente, se ha desarrollado para resolver una tarea propia del campo de la Bioinformática, la alineación de secuencias de ADN.

Abstract

In this project, we have developed a framework that helps parallelization of applications using mobile devices as nodes or processors. For its development, the project has been divided into three sub-projects (client, server and provider). The distributed computing and parallelization framework consists of a client and a server subprojects. The provider subproject has been developed for using the distributed computing framework created, and, more specifically, has been developed to solve a typical task in the field of Bioinformatics, DNA sequence alignment.

Índex

Índex de figures

Glossari de termes

Bioinformàtica	Camp que aplica la computació a la gestió i anàlisi de dades biològiques.
Xarxa de sensors	Xarxa de nodes amb sensors que col·laboren en una tasca comú.
Multi-core	Processadors amb múltiples nuclis.
API	(Aplication Programming Interface) Conjunt de funcions, subrutines i procediments que s'ofereix per ser fet servir per una altra aplicació.
MPP	Utlització d'un ampli número de processadors per realitzar un conjunt de processaments corrdinats en paral·lel
Serveis Web	Conjunt de protocols i estàndards que serveixen per intercanviar dades entre aplicacions
Volunteer computing	Tipus de computació distribuïda on els propietaris dels dispositius donen els seus recursos per un o més projectes
Idle	Temps en el que un processador no es utilitzat per cap programació
Fasta	Format de fitxer informàtic basat en text, utilitzat per representar seqüències d'àcids núclids.
Programació dinàmica	mètode per reduir el temps d'execució d'un algoritme.
MUSCLE	(Multiple Sequence Comparation by Log-Expectation) Aplicació de domini públic per alineament múltiple de seqüències.
Neighbor Joining	Mètode per la creació d'arbres fenètics (fenogrames).

Framework	Estructura conceptual o tecnològica que serveix de base pel desenvolupament de software.
Restful	És un estil arquitectònic que especifica regles, que aplicades a un servei de web indueix propietats desitjables, com rendiment i escalabilitat
Open source	Software distribuït i desenvolupat lliurament. Es centre més en els beneficis pràctics que en qüestions ètiques o de llibertat que tant es destaquen en el software lliure.
Hibernate	És una eina que facilita la relació entre atributs d'una base de dades relacional i el model d'objectes d'una aplicació.
Dalvik vm	(Dalvik Virtual Machine) és la màquina virtual que utilitza la plataforma per dispositius mòbils Android.
Jersey	Framework open source per el desenvolupament de serveis web de tipus RESTful en Java.
MySql	Sistema de gestió de bases de dades relacional
Android SDK	Conjunt d'eines pel desenvolupament d'aplicacions per sistemes Android.
Fitxer class	És un fitxer que conté Java bytecode que es pot executar en una màquina virtual de Java.
Dex	Equivalent dels fitxers class en el sistema operatiu Android.
Jar	És un format d'empaquetatge que s'utilitza per contenir diferents fitxers class o dex i recursos associats

1.Introducció

1.1. Objectius.

1.1.1. Propòsit

El projecte tracta de provar la potencialitat i la viabilitat de la computació distribuïda aplicada a dispositius mòbils (telèfons, tabletes, etc.) amb sistema operatiu Android.

1.1.2. Finalitat

Crear una plataforma per a la paral·lelització d'aplicacions a cost reduït que aprofiti la potencia i el creixent nombre de dispositius mòbils en la societat actual.

1.1.3. Objecte

Aquest projecte intenta mostrar les possibilitats de la computació distribuïda fent servir dispositius mòbils com nuclis de processament amb un cas concret del mon de la Bioinformàtica, l'alineació de seqüències d'ADN.

1.1.4. Abast

Aquest projecte implementa una plataforma genèrica per paral·lelitzar aplicacions i al mateix temps, treu partit dels dispositius mòbils per portar-ho terme. La plataforma resultant es pot utilitzar en camps molt diversos. Com a exemple, també s'ha implementat una eina d'us generalitzat en el camp de la Bioinformàtica, que és adient per mostrar el funcionament de la plataforma.

1.2. Planificació

A continuació es mostra el diagrama GANTT amb la planificació prèvia de les tasques a realitzar per desenvolupar el projecte.

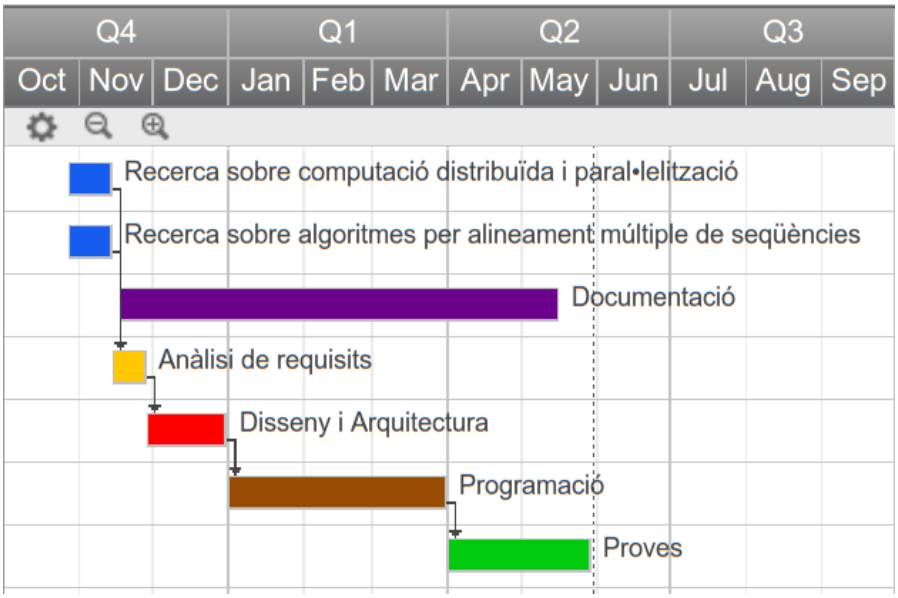


Figura 1-1. Diagrama GANTT de la planificació del projecte.

En un primer terme, es farà una recerca sobre paral·lelització i computació distribuïda (arquitectures, característiques, conceptes, etc.). En paral·lel es farà una recerca sobre l'alineament múltiple de seqüències (seqüències d'ADN, tipus d'alineaments, algoritmes, etc.)

Una vegada les recerques anteriors hagin finalitzat, és començarà la documentació del projecte que es farà al mateix temps que la resta de tasques i que es preveu que finalitzi poc abans de la finalització del projecte.

Al mateix temps que es comenci la tasca de documentar, també es començarà la tasca d'anàlisi de requisits. En aquesta fase es determinaran les necessitats pròpies i les condicions que ha de satisfer el projecte.

A continuació es farà el disseny arquitectònic on es planificarà, entre d'altres, els diferents components i com interactuaran entre ells, i es dissenyaran els diferents diagrames que s'utilitzaran en la implementació del projecte.

Per finalitzar es farà la implementació sobre les plataformes triades i les proves de funcionament dels components implementats i les proves globals de funcionament del projecte.

S'han calculat els temps de les tasques en funció de la seva complexitat repartint la major part del temps entre el disseny, la implementació i les proves, i es preveu que aquestes tasques es puguin solapar en alguns moments.

2.Visió general

A continuació es mostra una visió genèrica del projecte, la seva arquitectura, el seu disseny i s'aclareixen alguns conceptes que estan directament relacionats amb el funcionament i la filosofia tècnica en la que es basa.

2.1. Computació distribuïda

La computació distribuïda, avui en dia, esdevé una tecnologia present en un nombre cada vegada més gran d'aplicacions. Des de sistemes punt a punt, fins a xarxes de sensors o arquitectures multi-core.

Totes aquestes aplicacions tenen en comú la idea de fer servir molts processadors individuals treballant independentment per assolir un únic resultat final. Aquests processadors, també anomenats nodes, no tenen la necessitat de residir físicament sota el mateix maquinari encara que comparteixen informació i recursos per arribar a una solució conjunta.

Figura 2-1. Esquema de computació distribuïda

Una de les característiques més importants de la computació distribuïda es aprofitar la paral·lelització de processos, a vegades per aconseguir un increment en el rendiment i a vegades, simplement, perquè si no, algunes d'aquestes aplicacions serien inviables.

Hi han molts models de disseny aplicats a la computació distribuïda, els quals no es detallaran en aquesta memòria però la majoria tenen en comú les següents característiques:

- Paral·lelisme: distribuir les càrregues de treball en els diferents processadors o nodes fa créixer el rendiment de les aplicacions.

- Comunicació: el fet de que els nodes resideixin en diferents ubicacions físiques fa que la comunicació entre ells esdevingui un factor important en la computació distribuïda.

- Coordinació: és primordial una coordinació entre tots els nodes si el que es busca es una finalitat comuna.

- Sincronització: a vegades la paral·lelització directa no es possible ja que poden existir processos que depenguin de resultats anteriors per processar-se, per això, la sincronització es primordial.

2.2. Paral·lelització

Paral·lelització d'un procés, es una tècnica que consisteix en dividir el procés en tasques més petites que es puguin processar en diferents nuclis o processadors a l'hora, per tal de reduir el temps d'execució.

Figura 2-2 Procés sense Parel·lelitzar

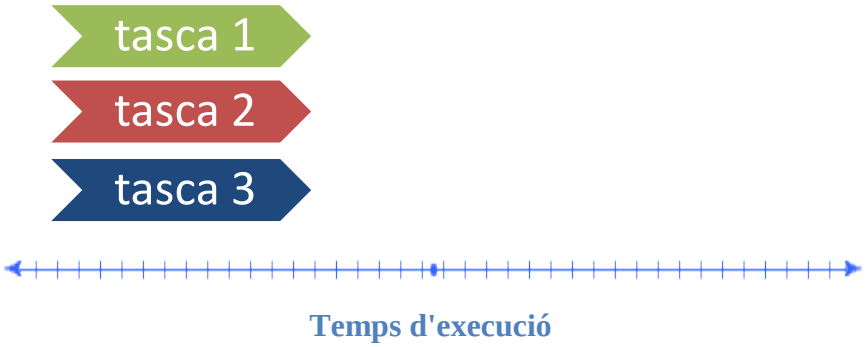


Figura 2-3 Procés paral·lelitzat

En les figures anteriors**,** es pot apreciar la diferencia resultant, en termes de temps d'execució, d'un procés sense paral·lelitzar i el mateix procés paral·lelitzat.

Dins dels diferents tipus de paral·lelitzacions existents, en aquest projecte es generalitza al tipus MPP (Massively Parallel Processing) que està enfocat a processos que es poden trencar en tasques més petites i processar-les independentment sense que hi hagi cap dependència directa entre elles. Es a dir, una tasca no necessita del resultat del processament d'una altra per poder ser processada.

2.3. Computació distribuïda sobre dispositius mòbils

En aquest projecte s'ha dissenyat i implementat un plataforma de computació distribuïda que fa servir dispositius mòbils (telèfons, tauletes, etc.) com els processadors o nodes encarregats d'executar les diferents tasques. A més s'ha dissenyat i implementat una aplicació que mostra la funcionalitat de la plataforma aplicada al mon de la Bioinformàtica. Concretament s'ha desenvolupat una aplicació que fa servir la plataforma per l'alineament de seqüències d'ADN.

A continuació es mostra una visió conjunta a nivell arquitectònic i funcional del sistema i després es detallen els diferents components que hi formen part.

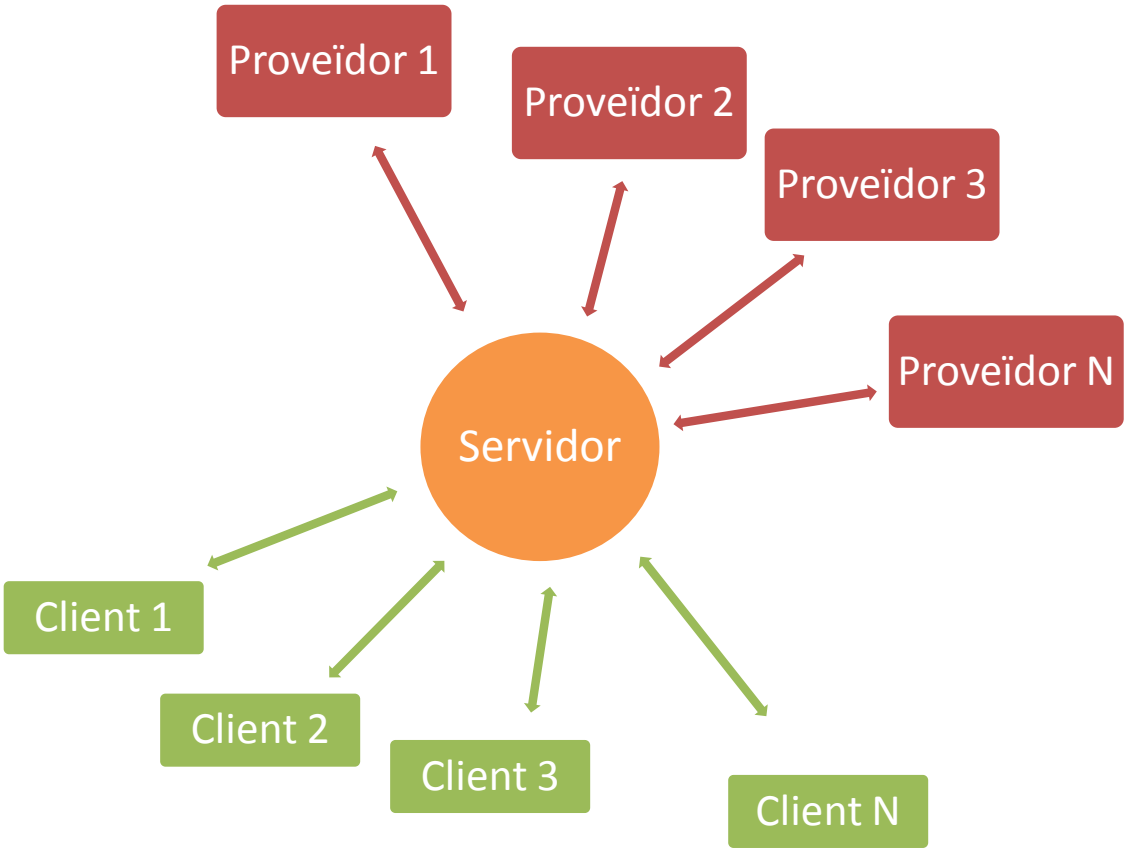


Figura 2-4. Diagrama del funcionament global del sistema

En la figura anterior, es poden diferenciar les tres parts que conformen el projecte i com es comuniquen entre si. Cada una de les parts ha estat implementada en subprojectes individuals.

L'aplicació servidor conjuntament amb l'aplicació client, són les que conformen la plataforma de paral·lelització en si, i les aplicacions proveïdores, són les que fan servir aquesta plataforma.

El flux d'informació és el següent:

1. Una aplicació proveïdora crea les diferents tasques que necessita processar i les envia al servidor.
2. El servidor posa les tasques rebudes a disposició dels clients perquè les processin.
3. Els clients processen les tasques oferides pel servidor i li retornen.
4. El servidor posa les tasques processades a disposició dels proveïdors.
5. Els proveïdors recuperen les tasques ja processades.

El sistema està dissenyat per donar servei a varies aplicacions proveïdores a l'hora. També s'ha tingut en compte que la plataforma (servidor i clients) no hagi de conèixer en cap moment la lògica de processament de les tasques enviades pels proveïdors i pugui donar servei a qualsevol aplicació proveïdora que implementi l'API de comunicació amb el servidor. Això s'aconsegueix encapsulant la lògica del algoritme dins de la tasca que s'envia, i així el client la pot executar sense haver-la de conèixer a priori.

2.3.1. Servidor

El servidor és la part central del projecte. És el component que s'encarrega de gestionar les tasques que els clients han de processar i després tornar-les als seus respectius proveïdors.

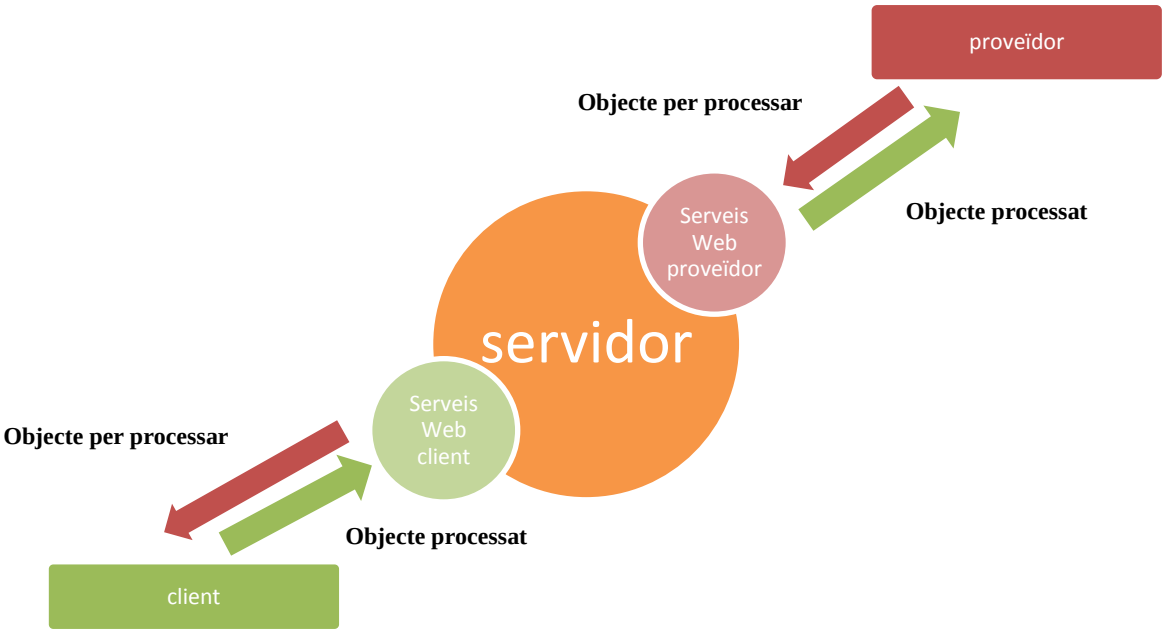


Figura 2-5. Esquema del flux dels objectes a través del servidor

La comunicació, tant amb els clients, com amb els proveïdors, es fa a partir de serveis web per tal de fer la plataforma més oberta futures ampliacions i més dinàmica als possibles canvis o modificacions.

Les aplicacions proveïdores, utilitzen uns serveis web específics per fer arribar les parts (objectes) que volen que siguin processades per la plataforma.

El servidor identifica individualment cada objecte i a quina aplicació pertany. A més també controla en tot moment en quin estat del procés es troba l'objecte.

El servidor posa a disposició dels clients aquests objectes per mitjà dels serveis web de client.

Els clients obtenen aquests objectes i els processen. Per establir l'ordre en que el servidor ofereix aquests objectes, s' utilitza un algoritme propi.

Els objectes processats, són retornats al servidor per part de les aplicacions clients, també a través dels serveis web, i el servidor els posa a disposició de l'aplicació proveïdora que el va originar.

2.3.2. Client

Dins de l'arquitectura de la computació distribuïda, aquest component és el que fa les funcions de processador o node. El client es l'encarregat de processar tasques individuals i retornar-les al servidor.

En aquest projecte, aquest component, s'ha desenvolupat per funcionar sobre dispositius mòbils, i més concretament sobre dispositius amb sistema operatiu Android. S'ha implementat amb la filosofia de computació voluntària (Volunteer computing) on els usuaris dels dispositius, en aquest cas dispositius mòbils, donen voluntàriament els temps d'inactivitat (idle) per ser aprofitats pels diferents projectes.

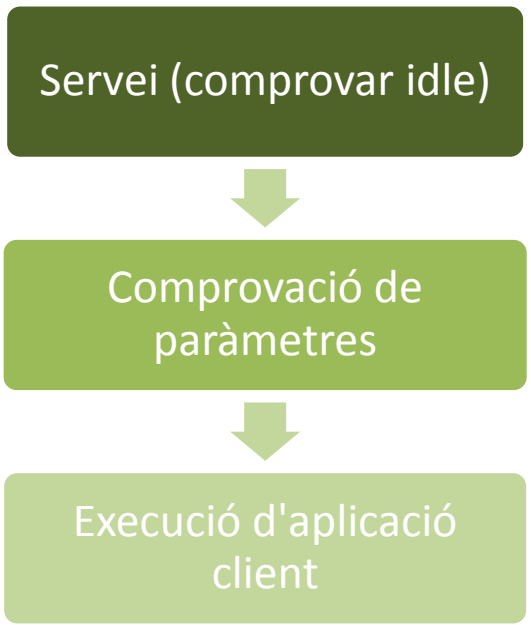


Figura 2-6. Cicle bàsic del client

L'aplicació funciona com un servei en segon pla que està latent fins que es donen una sèrie de circumstàncies. En aquestes circumstàncies o paràmetres, s'ha tingut en compte les peculiaritats pròpies dels dispositius mòbils. Per exemple, el servei nomes executarà l'aplicació quan el dispositiu porti un cert temps en estat d'inactivitat, estigui connectat a una xarxa Wifi, estigui connectat a la xarxa elèctrica, etc.

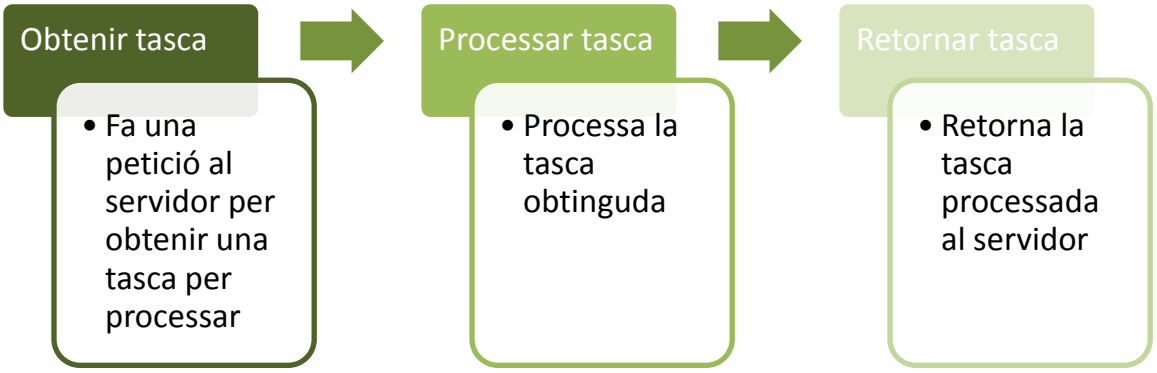


Figura 2-7. Esquema del flux d'execució de l'aplicació client

Quan tots els paràmetres són favorables l'aplicació es posa en marxa automàticament i fa una petició al servidor per obtenir una tasca per processar, a continuació processar-la i per fi retornar-la al servidor.

En qualsevol moment que un dels paràmetres de control canvia de valor, l'aplicació cancel·la el procés i queda de nou en espera fins que es tornen a donar les circumstàncies correctes pel seu funcionament.

2.3.3. Proveïdor

El proveïdor és l'aplicació que utilitza la plataforma de paral·lelització per processar les seves tasques .

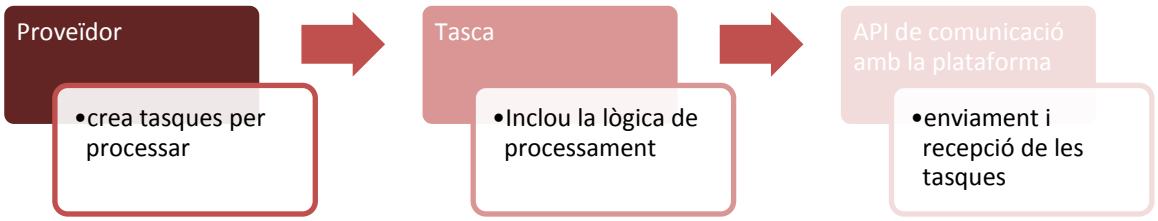


Figura 2-8. Esquema de l'arquitectura d'un proveïdor

Per tal de comunicar-se i fer servir la plataforma, les aplicacions proveïdores utilitzen una API de comunicacions amb el servidor. Aquesta API ofereix les funcionalitats necessàries per enviar les tasques al servidor i recuperar-les una vegada processades. Les tasques que envia el proveïdor contenen internament tota la lògica de processament necessària perquè tant el client com el servidor, no tinguin la necessitat de conèixer-la.

En aquest cas en concret, s'ha desenvolupat una aplicació que ofereix el servei d'alinear seqüències d'ADN, una tasca habitual en el camp de la Bioinformàtica i la investigació genètica

2.4. Alineament de seqüències d'ADN

2.4.1. Què es l'ADN?

L'ADN (àcid desoxiribonucleic), és una molècula que conté el material hereditari i informació de totes les funcions biològiques en quasi tots els organismes. L'ADN conté les instruccions genètiques que es fan servir pel desenvolupament i funcionament de tots els organismes vius coneguts.

La majoria del ADN està localitzat en el nucli de les cèl·lules.

Dins de l'ADN, la informació està emmagatzemada com un codi format per quatre bases químiques:

Figura 2-9. Les quatre bases de l'ADN

L'ADN està estructurat com una doble hèlix o una escala de caragol on els *esglaons* estan formats per dues bases. Les bases que formen els *esglaons* (parell de bases) nomes tenen unes unions específiques possibles. L'adenina (A) només pot enllaçar-se a la timina (T) i la citosina nomes pot enllaçar-se amb la guanina (G)

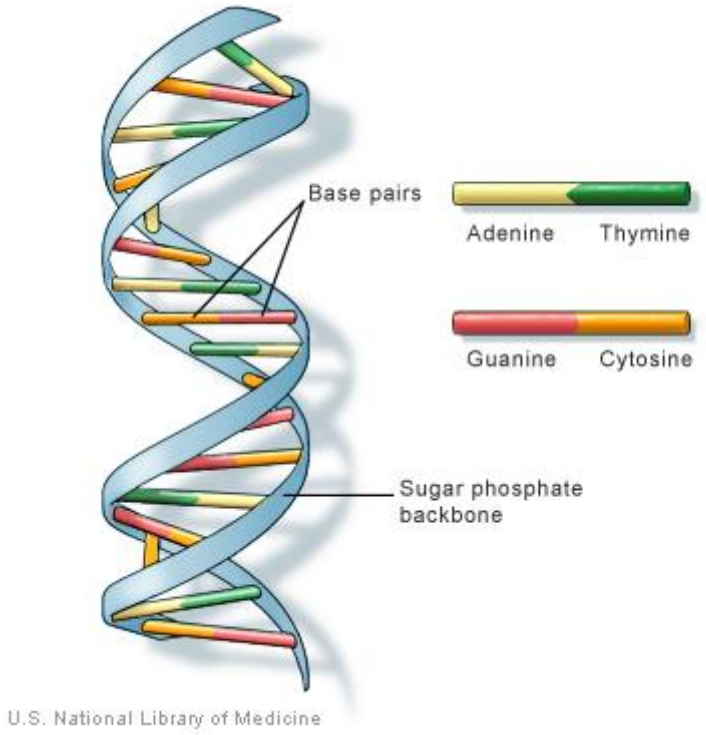


Figura 2-10. Estructura i composició de l'ADN

2.4.2. Què es una seqüència d'ADN?

Una seqüència genètica o seqüència d'ADN és una successió dels codis químics que representen l'estructura primària d'una molècula d'ADN amb la capacitat de transportar informació.

```
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA for prepro cortistatin like peptide, complete cds.|len=368
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCC
CCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGC
CTCCTGACTTTCCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCTCATAGGAGAGG
AAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCCAGCAATCCGCGCGCCGGGACAGAATGCC
CTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAG
TTTAATTACAGACCTGAA
```

Figura 2-11. Exemple de seqüència d'ADN en format FASTA

Una successió a partir de quatre bases pot ser una seqüència. Depenent de la seva funció biològica, una seqüència pot ser codificant o no codificant.

Les seqüències d'ADN són extretes a partir d'un procés anomenat seqüenciació d'ADN.

2.4.3. Per què alinear seqüències d'ADN?

Alienar seqüències és comparar-les per trobar similituds i diferencies entre elles. L'alineament de seqüències d'ADN consisteix en alinear les bases de les seqüències. Per alinear-les, es posicionen fent coincidir les que són iguals i afegint espais o guions (gaps).

Figura 2-12. Exemple de seqüències no alineades

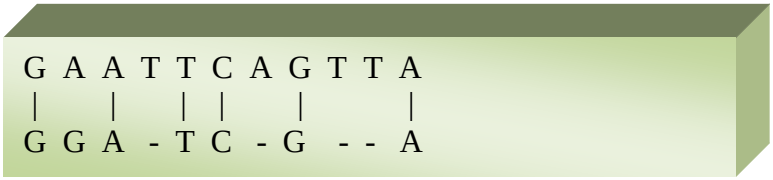


Figura 2-13. Exemple de seqüències alineades

Una seqüència per si sola no es molt informativa, necessita comparar-se amb unes altres per poder extreure informació. L'alineació maximitza la similitud entre les diferents seqüències.

A continuació es mostren alguns dels diferents motius pels que s'utilitza l'alineament de seqüències:

- Comprovar si les seqüències provenen d'un ancestre comú
- Comprovar si les seqüències tenen funcions similars
- Observar les mutacions sofertes al llarg del temps
- Primer pas per generar arbres filogenètics
- Etc.

2.4.4. Alineament de dues seqüències

Hi ha dos tipus diferents d'alineaments, l'alineament global que consisteix en alinear la longitud total de les seqüències i l'alineament local que només alinea les parts que mostren similituds.

En aquest projecte s'ha utilitzat el tipus global. Aquest tipus té com a característica que és un mètode heurístic, que vol dir que, s'obté sempre el mateix resultat però no es pot assegurar que aquest resultat sigui el millor possible.

Hi han molts algoritmes per l'alineament de seqüències. En aquest projecte s'utilitza entre d'altres l'algoritme de Needleman – Wunsch.

Aquest algoritme està basat en el mètode de programació dinàmica per reduir la complexitat de l'alineament.

L'algoritme es basa en valorar els diferents aparellaments amb una puntuació concreta. Hi ha diferents formes de valorar els aparellaments, en aquest cas, s'utilitza el sistema de matriu de similitud (S).

En la figura següent es mostra la matriu de similitud (S) on es representen las puntuacions assignades als diferents aparellaments. D'altra banda, cada "gap" es penalitza amb una puntuació de (-5).

	A	G	C	T
A	10	-1	-3	-4
G	-1	7	-5	-3
C	-3	-5	9	0
T	-4	-3	0	8

Figura 2-14. Matriu de similitud (S)

Basant-se en aquesta puntuació s'obté una puntuació total que valora l'alineament final, quan més alta és la puntuació total, millor es l'aparellament.

Seqüència 1 = AACT
Seqüència 2 = AATT

Alineament 1)

```
A  A  C  T
|  |     |        SCORE TOTAL =  10 + 10 + 0 + 8 = 28
A  A  T  T
```

Alineament 2)

```
A  A  C  T  -
|  |     |        SCORE TOTAL =  10 + 10 + (-5) + 8 + (-5) = 18
A  A  -  T  T
```

L'alineament 1 es millor que l'alineament 2.

Figura 2-15. Exemple de valoració d'alineaments

A continuació es mostren els passos que segueix l'algoritme Needleman – Wunsch per realitzar l'alineament de dues seqüències.

1. Crear una matriu de comparació.

   Es crea una matriu amb les dimensions de les longituds de les seqüències + 1 i es posiciona la primera seqüència a la part de dalt i la segona a la primera columna.

   La segona fila i la segona columna s'omple amb una successió de valors començant pel 0 i amb un increment de -5. Aquest increment ve donat pel valor associat al gap (-5)

		A	**A**	**C**	**T**
	0	-5	-10	-15	-20
A	-5				
A	-10				
T	-15				
T	-20				

Figura 2-16. Exemple de matriu de comparació (Needleman – Wunsch)

2. Omplir la matriu amb valors (scoring)

   Per omplir la matriu de valors es segueix la següent formula:

$$F_{ij} = max(F_{i-1,j-1} + S(A_i, B_j), F_{i,j-1} + d, F_{i-1,j} + d) \quad (2.1)$$

   On ***F*** és la matriu de comparació, ***i*** és la fila, ***j*** la columna, ***S*** la matriu de similitud, ***A*** la primera seqüència, ***B*** la segona seqüència i ***d*** el valor de penalització del "gap".

   Per tant, per cada cel·la s'ha de triar el valor màxim entre els 3 possibles. El primer és el valor de la cel·la superior esquerra més el valor del aparellament de les bases corresponents a la cel·la, el segon és el valor de la cel·la esquerra més el valor de la penalització del "gap" i el tercer és el valor de la cel·la superior més el valor de penalització del "gap".

Per omplir la primera cel·la (A, A)

$$\boldsymbol{F_{i-1,j-1} + S(A_i, B_j) = 0 + 10 = 10}$$

$$\boldsymbol{F_{i,j-1} + d = (-5) + (-5) = -10}$$

$$\boldsymbol{F_{i-1,j} + d = (-5) + (-5) = -10}$$

El valor màxim entre els tres possibles és **10** per tant s'assigna aquest valor.

		A	**A**	**C**	**T**
	0	-5	-10	-15	-20
A	-5	**10**			
A	-10				
T	-15				
T	-20				

Si es continua omplint la matriu s'obté el següent resultat:

		A	**A**	**C**	**T**
	0	-5	-10	-15	-20
A	-5	**10**	**5**	**0**	**-5**
A	-10	**5**	**20**	**15**	**10**
T	-15	**0**	**15**	**20**	**23**
T	-20	**-5**	**10**	**15**	**28**

Figura 2-17. Exemple de com omplir la matriu de valors (Needleman – Wunsch)

3. Recórrer la matriu de forma inversa per obtenir l'alineament.

   Una vegada la matriu de valors està completada, la última cel·la ($\boldsymbol{F_{m,n}}$) conté el valor de la puntuació total i màxima de l'alineació. (En el cas de la figura anterior, el valor total de l'alineament és 28).

   A partir d'aquí, s'ha de recórrer la matriu de manera inversa fins arribar a la primera fila o la primera columna. Per decidir el camí a seguir i per tant, les parelles de bases que s'han d'alinear, a cada cel·la ($\boldsymbol{F_{i,j}}$) s'ha de comprovar l'origen del seu valor ($\boldsymbol{F_{i-1,j}}, \boldsymbol{F_{i,j-1}}, \boldsymbol{F_{i-1,j-1}}$). En cas de que l'origen sigui $\boldsymbol{F_{i-1,j}}$ voldria dir que l'element $\boldsymbol{A_i}$ s'ha alienat amb un "gap", si l'origen del valor és $\boldsymbol{F_{i,j-1}}$ voldria dir que l'element $\boldsymbol{B_i}$ s'ha alineat amb un "gap" i si l'origen del valor és $\boldsymbol{F_{i-1,j-1}}$ voldria dir que els elements $\boldsymbol{A_i}$ i $\boldsymbol{B_i}$ s'han alienat junts.

Es comença a recorre la matriu des de la cel·la inferior dreta i es comprova d'on prové el valor que conté. En aquest cas el valor 28 prové de la superior esquerra ja que 20 + 8 = 28. Per tant, l'últim element de l'alineament seria T,T.

A continuació es comprova el valor de la cel·la del que prové el resultat anterior, en aquest cas la cel·la que conté el valor 20. I es fa el mateix que en el pas anterior. En aquest cas el valor de la cel·la prové de la cel·la superior esquerra, per tant els valors alineats són C,T.

		A	**A**	**C**	**T**
	0	-5	-10	-15	-20
A	-5	**10**	**5**	**0**	**-5**
A	-10	**5**	**20**	**15**	**10**
T	-15	**0**	**15**	**20**	**23**
T	-20	**-5**	**10**	**15**	**28**

Si es continua fins arribar a la primera columna o la primera fila, s'obté el següent resultat que és l'alineament de les dues seqüències:

A A C T
| | |
A A T T

SCORE TOTAL = 10 + 10 + 0 + 8 = **28**

Figura 2-18. Exemple de com fer l'alineament a partir de la matriu de valors (Needleman – Wunsch)

2.4.5. Alineament múltiple de seqüències (MSA)

En el apartat anterior, s'ha descrit com fer l'alineament de dues seqüències i s'ha descrit amb un exemple el funcionament de l'algoritme Needleman – Wunsch.

Malauradament, per alinear un número més gran de seqüències no es pot seguir el mateix camí. El problema és que si per alinear dues seqüències es necessita una matriu de 2 dimensions, per alinear N seqüències es necessitaria una matriu de N dimensions. Això es contempla que podria ser viable per valors de N <10 i longituds de seqüències inferiors a 200. Com es pot veure la complexitat creix depenent del número de seqüències.

Per solucionar l'anterior problema, s'han desenvolupat diferents algoritmes dels quals podem destacar dos tipus, els algoritmes progressius i els algoritmes iteratius.

Els algoritmes progressius calculen els alineaments de parelles , trien el millor aparellament i van afegint les altres seqüències progressivament.

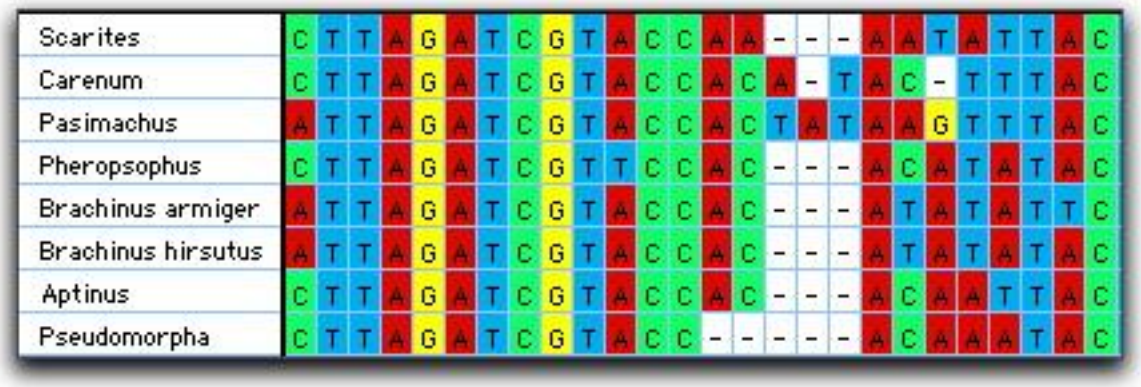


Figura 2-19 Exemple d'alineament de múltiples seqüències

CLUSTAL és el software més utilitzat per l'alineament múltiple de seqüències i fa servir un algoritme progressiu, que és el que s'ha implementat en aquest projecte.

Els algoritmes iteratius primer troben un sub-solució de manera progressiva i després, per mitjà de tècniques de programació dinàmica, van extraient seqüències i alineant-les als perfils fins que convergeixen.

Un exemple de software que fa servir un algoritme iteratiu és MUSCLE (Multiple Sequence Comparation by Log-Expectation).

2.4.6. CLUSTAL

Clustal és el software més utilitzat per l'alineament múltiple de seqüències, utilitza un algoritme progressiu conegut com Feng and Doolittle. Aquest algoritme és el que s'ha implementat en aquest projecte. L'algoritme Feng and Doolittle consta de tres passos:

1. Alineament global de les seqüències per parelles.

   En aquets pas, es fan tots els aparellaments possibles entre les seqüències. Els alineaments entre seqüències es fan mitjançant l'algoritme Needleman – Wunsch i es construeix una matriu amb els resultats de les alineacions.

	Seq 1	**Seq 2**	**Seq 3**	**Seq 4**	**Seq 5**
Seq 1	0	5	9	9	8
Seq 2	5	0	10	10	9
Seq 3	9	10	0	8	7
Seq 4	9	10	8	0	3
Seq 5	8	9	7	3	0

Figura 2-20. Exemple de matriu de valors d'aparellaments (Feng - Doolittle)

En aquest projecte aquesta part és la que s'ha paral·lelitzat i la que utilitza la plataforma de computació distribuïda. Les característiques d'aquesta part del algoritme són les idònies per la paral·lelització ja que cada aparellament es pot resoldre per si sol i no necessita del resultat dels altres aparellaments per ser processat. A més aquesta part del algoritme és la que ocupa quasi el 90% del temps total d'execució en sistemes no paral·lelitzats, per tant, la paral·lelització d'aquesta part del algoritme hauria de tindre un efecte directe en el rendiment del processament total.

2. Creació d'un arbre guia.

Per la construcció de l'arbre guia es fa servir un mètode que es coneix com Neighbor Joining (NJ).

Aquest mètode consisteix en traduir els resultats de la matriu de valors d'aparellaments, en una nova matriu de distancies. Es van aparellant les seqüències amb menor distancia entre si i cada vegada que es fa un aparellament es torna a calcular la matriu de distancies.

	Seq 1	**Seq 2**	**Seq 3**	**Seq 4**	**Seq 5**
Seq 1		-50	-38	-34	-34
Seq 2	-50		-38	-34	-34
Seq 3	-38	-38		-40	-40
Seq 4	-34	-34	-40		-48
Seq 5	-34	-34	-40	-48	

Figura 2-21 Exemple de matriu de distancies (Feng - Doolittle)

A partir d'aquests aparellaments, es construeix l'arbre guia on les fulles són les seqüències i els nodes intermedis són els alineaments que s`han de fer.

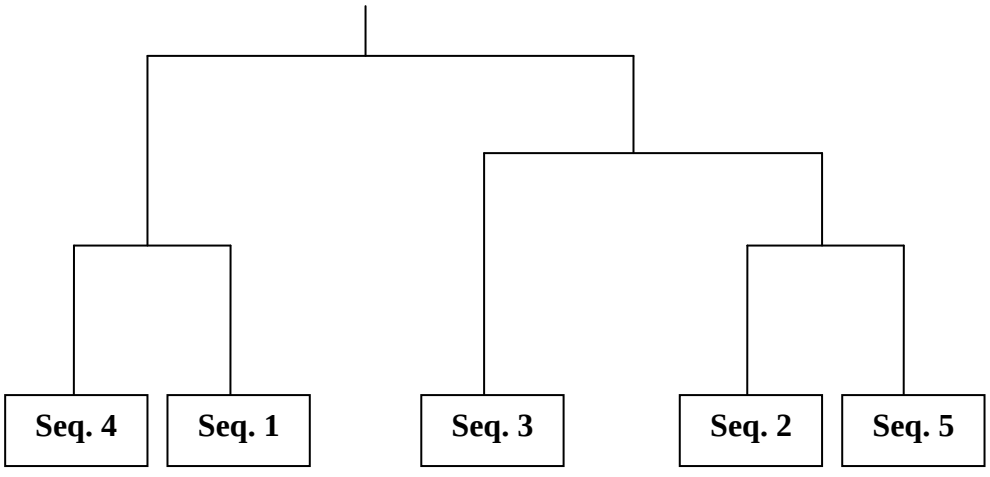


Figura 2-22. Exemple d'arbre guia (Feng - Doolittle)

3. Alineament progressiu utilitzant l'arbre guia per denotar l'ordre de l'alineament.

   A partir de l'arbre descrit en l'apartat anterior, es van fent els alineaments progressivament.

   Segons l'arbre de la figura anterior, l'ordre dels alineaments seria el que es mostra a la taula

Ordre dels alineaments			
Ordre	**Component 1**	**Component 2**	**Resultat**
1	**Seq. 2**	**Seq. 5**	**Alineament 1**
2	**Seq. 4**	**Seq. 1**	**Alineament 2**
3	**Seq. 3**	**Alineament 1**	**Alineament 3**
4	**Alineament 2**	**Alineament 3**	**Alineament Total**

Figura 2-23. Exemple d'ordre d'alineament de seqüències.

3.Implementació

En aquest apartat es detalla la solució tècnica implementada pel desenvolupament de la solució descrita en l'apartat anterior.

La implementació està separada en tres subprojectes diferents. Al final de la memòria s'adjunten els annexes on es mostra la documentació tècnica que s'ha fet servir per realitzar la implementació de cada un d'ells.

3.1. Servidor

El servidor és la part encarregada de distribuir els objectes a processar entre els diferents nodes i una vegada processats tornar-los a les aplicacions que els van originar.

Tecnologies:

- La implementació s'ha fet en llenguatge Java (v.7). [24]
- La implementació dels serveis web s'ha fet en amb Jersey framework (v. 2.16), per la seva simplicitat a l'hora de desenvolupar serveis restful. [14]
- Per la persistència s'ha utilitzat Hibernate framework (v. 2.0) per la seva productivitat i amplia documentació. [15]
- Com base de dades s'ha triat MySql (v. 5.5.41) perquè ofereix les prestacions necessàries pel projecte i la seva llicencia és Open Source. [25]

3.1.1 Paquets

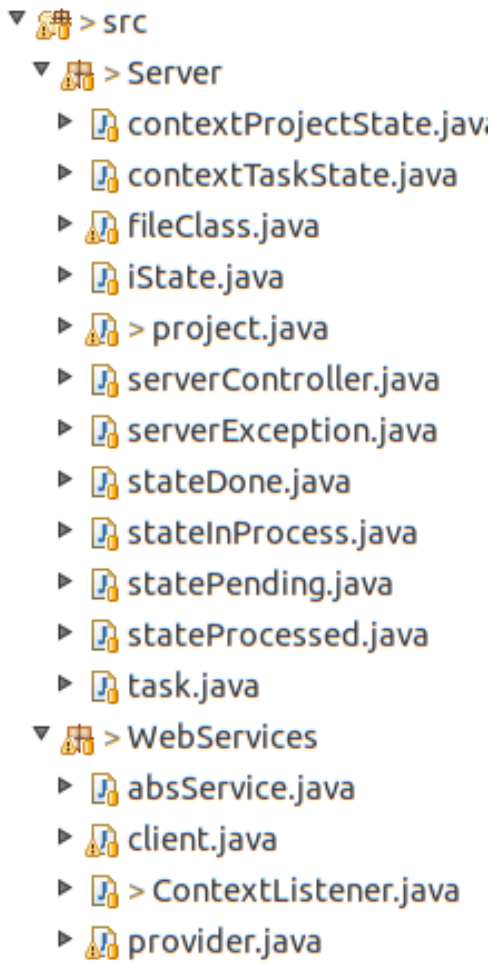


Figura 3-1. Estructura de paquets del servidor

El projecte Java del servidor s'ha dividit en dos paquets, Server i WebServices, per diferenciar la part que ofereix els serveis restful, a les aplicacions proveïdores i a les aplicacions clients (WebServices), de la part de domini que conté la lògica del funcionament intern (Server).

3.1.2 Domini

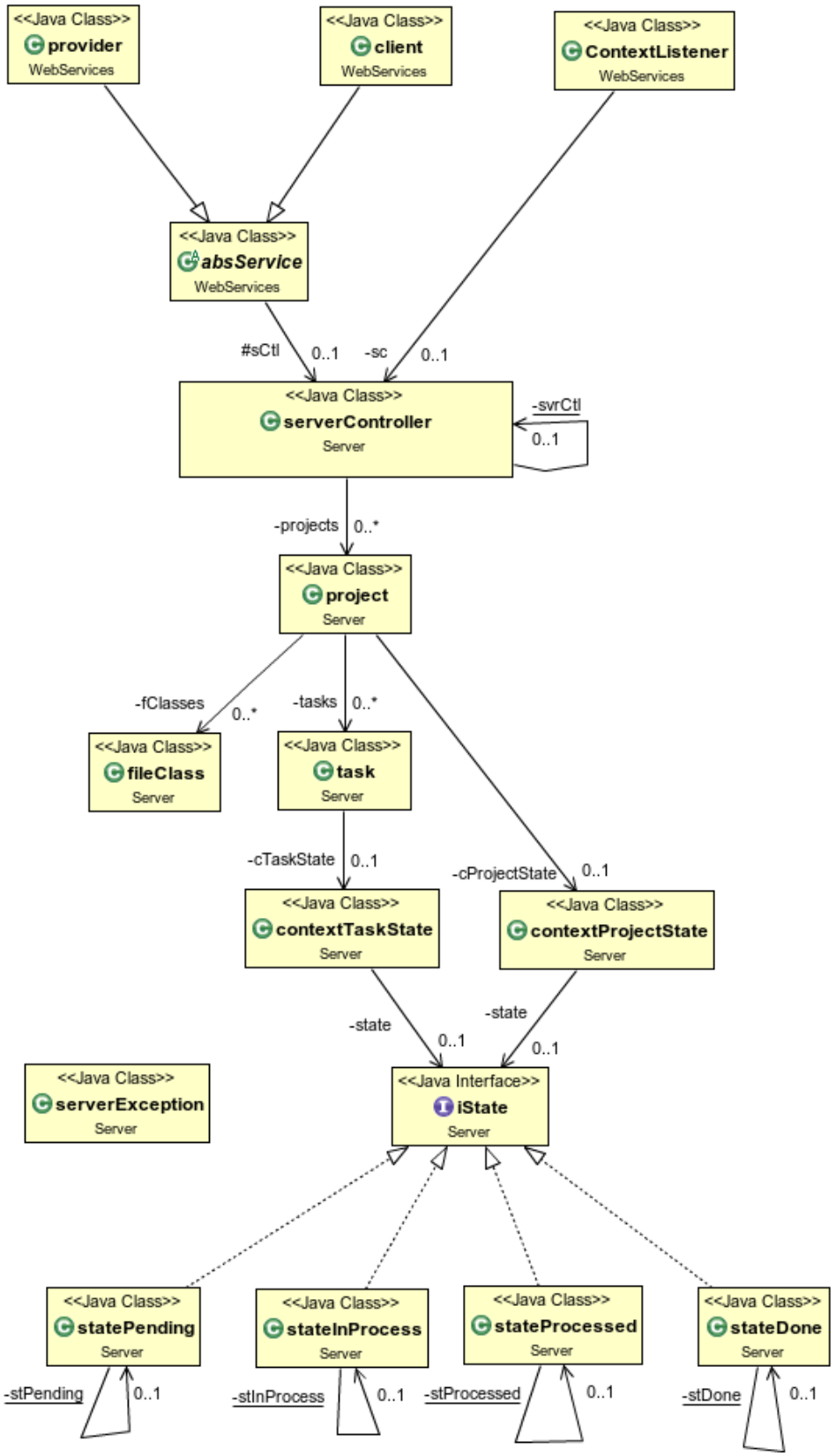


Figura 3-2. Diagrama de domini

3.1.3 Serveis web (Restful)

Les classes que conformen la part de comunicació entre el servidor i les aplicacions externes (clients i proveïdors), són Client i Provider. Totes dues classes estenen d'una classe abstracta (AbsService) que simplement càrrega la classe ServerController.

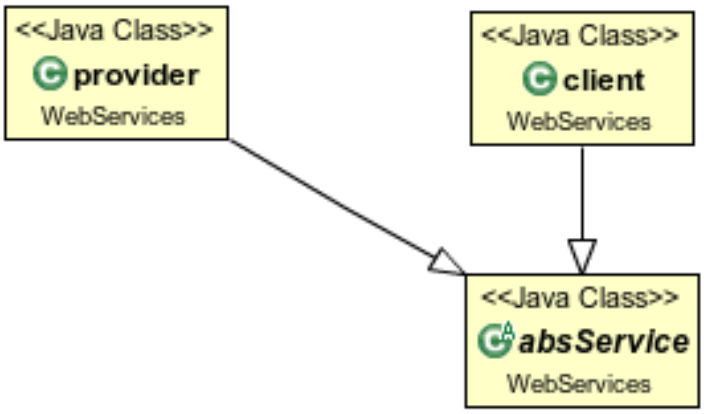


Figura 3-3. Classes que implementen els serveis Web

3.1.3.1 Classe Provider

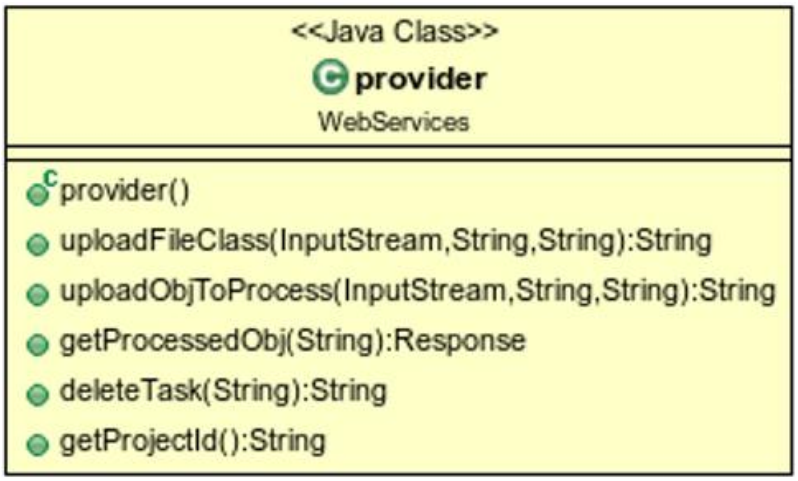


Figura 3-4. Classe Provider

La classe provider conté els serveis web que consumeixen les aplicacions proveïdores d'objectes per processar. Els mètodes de la classe fan merament de interlocutors entre les aplicacions proveïdores i la classe serverController, la qual es detalla més endavant. S'ha fet servir el framework Jersey per implementar els diferents serveis.

```
@PUT
@Consumes(MediaType.MULTIPART_FORM_DATA)
@Produces(MediaType.TEXT_PLAIN)
@Path("uploadObj")
public String uploadObjToProcess(@FormDataParam("file") InputStream fileInputStream,
        @FormDataParam("idProject") String idProject,
        @FormDataParam("idFileClass") String idFileClass){
```

Figura 3-5. Nomenclatura pròpia del framework Jersey

A la figura anterior es pot apreciar la nomenclatura pròpia de Jersey. En aquest cas @PUT està indicant que el servei respondrà a peticions de tipus PUT, @Consumes indica el tipus de dades que el servei espera rebre, @Produces indica el tipus de dades que el servei produirà i @Path indica la ruta on aquest servei en concret escolta les peticions.

Els passos que segueixen les aplicacions proveïdores per utilitzar els serveis són els següents: (per tindre una visió detallada de cada pas es poden consultar els diagrames de seqüència de l'annex 2)

1. Enviar l'arxiu .class al que pertany l'objecte a processar
2. Enviar l'objecte a processar
3. Rebre l'objecte processat
4. Esborra la tasca creada.

Els serveis web que ofereix la classe provider són:

- UploadFileClass: es fa servir perquè les aplicacions proveïdores d'objectes facin arribar al servidor els arxius .class al que pertanyen els objectes a processar
- UploadObjToProcess: és l'encarregat de rebre els objectes que les aplicacions proveïdores envien perquè siguin processades pels clients.
- GetProcessedObj: és el mètode que es fa servir perquè les aplicacions proveïdores obtinguin els objectes que ja han estat processats.
- DeleteTask: es fa servir per canviar l'estat d'un objecte de la classe "task", a finalitzat. (La classe "task" es descriu més endavant).

3.1.3.2 Classe Client

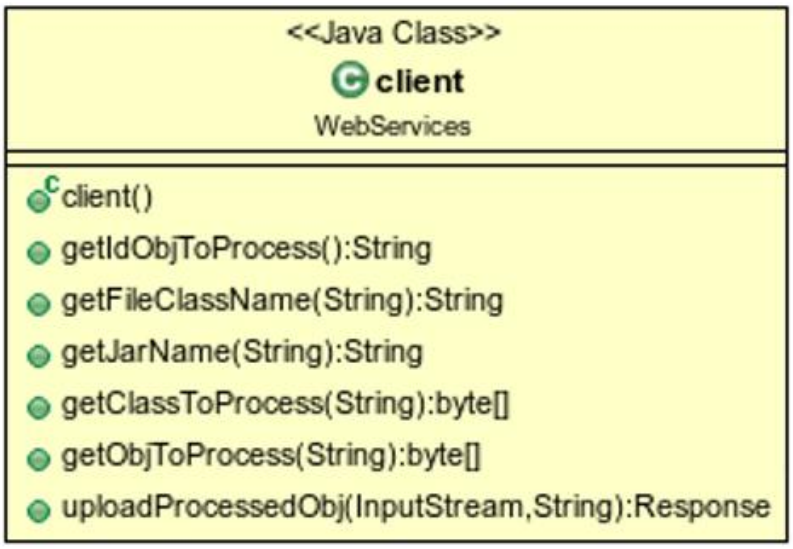


Figura 3-6. Classe Client

La classe client implementa els serveis web oferts als Clients (dispositius mòbils) per interactuar amb el servidor. Com la classe Provider, també utilitza el framework Jersey per oferir els serveis.

Els passos que segueixen les aplicacions clients són els següents: (per tindre una visió detallada de cada pas es poden consultar els diagrames de seqüència de l'annex 2)

1. Demanar un identificador de tasca per processar.
2. Demanar el nom de la classe corresponent al objecte a processar
3. Demanar el nom de l'arxiu .jar que conté la classe del objecte a processar.
4. Demanar l'arxiu .jar que conté la classe
5. Demanar l'objecte a processar
6. Processar l'objecte
7. Retornar l'objecte processat

Els serveis implementats que faciliten els passos anteriors són:

- GetIdObjToProcess: retorna el identificador d'una tasca pendent de processar
- GetFileClassName: retorna el nom de la classe del objecte a processar.
- GetJarName; retorna el nom de l'arxiu Jar on la classe està continguda.
- GetClassToProcess: retorna l'arxiu de classe al que pertany l'objecte a processar.
- GetObjectToProcess: retorna l'objecte a processar.
- UploadProcessedObj: rep l'objecte ja processat.

3.1.3.3 ContextListener

Figura 3-7. Classe ContextListener

Aquesta classe s'utilitza per carregar els paràmetres necessaris pel correcte funcionament de l'aplicació i per recuperar de la persistència tots els objectes pendents de processar.

El mètode "contextInitialized" es executat al iniciar l'aplicació.

La recuperació es fa nomes d'objectes i projectes que tinguin un estat pendent o en procés i no es recuperen els que tenen un estat de finalitzat.

3.1.4 Patró façana (Facade)

Per tal de separar la part de serveis web de la resta del domini i que aquests no hagin de conèixer la complexitat de la implementació d'aquest subsistema, s'ha utilitzat el patró de disseny Facade.

El patró Facade consisteix en una classe que fa d'interlocutora entre les diferents parts d'un sistema, amagant la complexitat i fent-lo més independent, portable i reutilitzable.

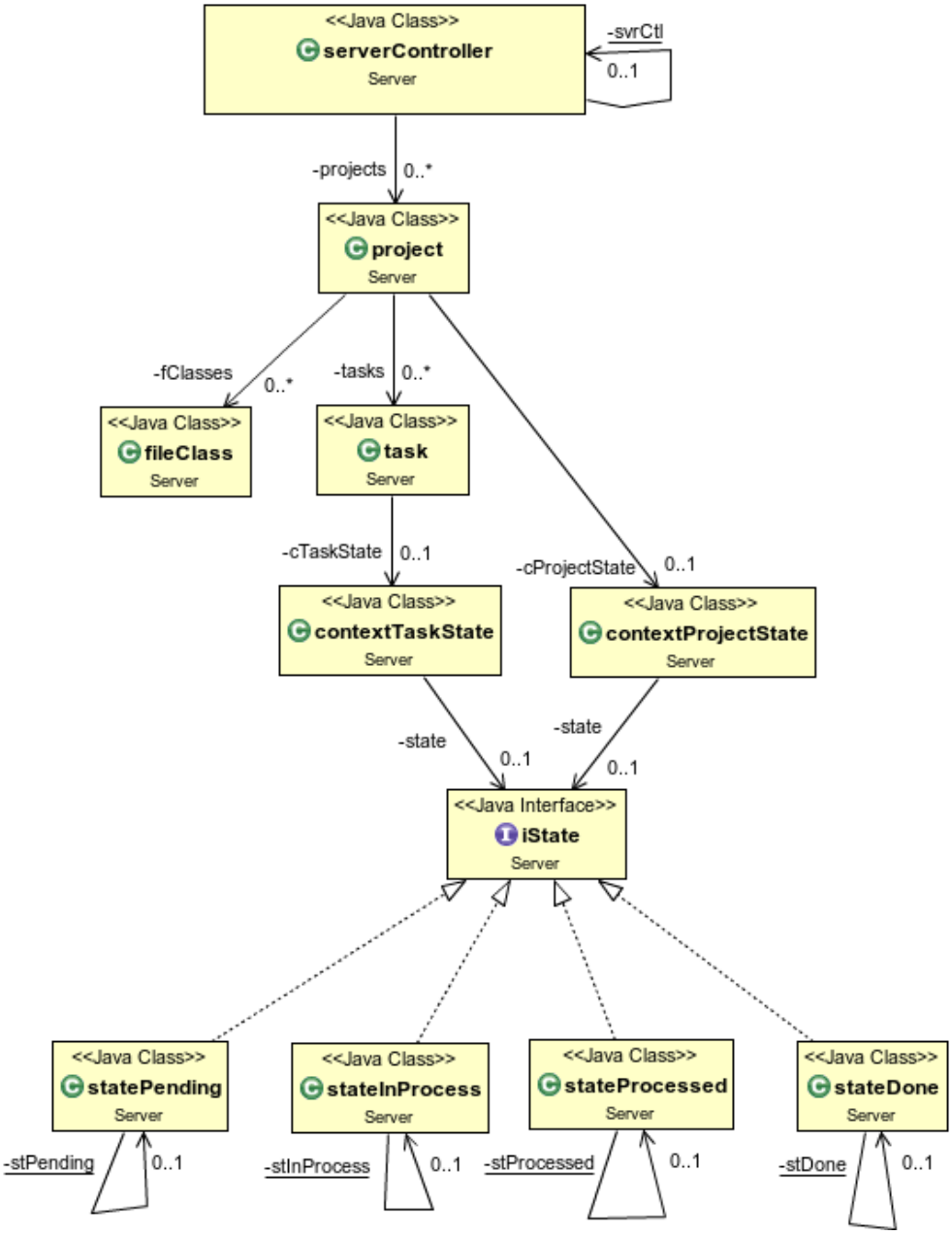


Figura 3-8. Patró façana

En aquest cas la classe serverController es la que fa de façana del subsistema dedicat a gestionar els objectes a processar. Totes les peticions de les classes que conformen els serveis web es fan sobre aquesta classe, sense que les classes dels serveis web coneguin cap de les classes del subsistema. Això entre altres, ofereix l'avantatge de poder modificar el subsistema independentment sense que afecti a la part de serveis web.

Per tal de que nomes hi hagi una instancia de la classe serverController, aquesta classe implementa el patró Singleton.

3.1.5 Subsistema de gestió d'objectes

El conjunt de classes que s'encarrega de la gestió dels diferents objectes processats i pendents de processat s'ha denominat subsistema de gestió d'objectes. Les classe que en formen part i la seva implementació es descriuen en els següents apartats.

3.1.5.1 Classe ServerController

És la classe façana, totes les peticions al subsistema es fan des d'aquesta classe.

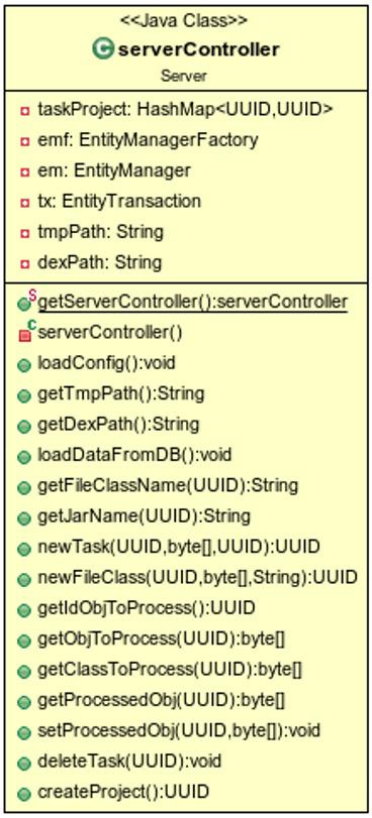


Figura 3-9. Classe ServerController

Aquesta classe conté dos col·leccions per emmagatzemar els projectes actius i la relació entre l'identificador d'una tasca i l'identificador del projecte al que pertany. Aquesta col·lecció s'ha creat per reduir temps de cerca del projecte al que pertany una tasca concreta sense haver de recorre tots els projectes actius.

```
public class serverController {

    private HashMap<UUID, project> projects;
    private HashMap<UUID, UUID> taskProject;
    private EntityManagerFactory emf;
    private EntityManager em;
    private EntityTransaction tx =null;
    private String tmpPath="";
    private String dexPath="";
```

Figura 3-10. Atributs de la classe ServerController

Tots els identificadors de l'aplicació es creen amb la classe UUID de Java. Totes les col·leccions "Map", s'implementen com LinkedHashMap ja que interessa l'ordre d'inserció dels valors.

Els mètodes referents a la gestió directa dels objectes task, i fileClass fan crides directes a mètodes de la classe project que és qui realment s'acobla amb aquestes classes.

El mètode loadClass càrrega objectes del tipus project, task i fileclass des de la base de dades al subsistema. Això es fa nomes una vegada a l'inici de l'aplicació per deixar el sistema en el mateix estat que estava en la última execució.

3.1.5.2 Classe Project

S'ha de tenir en compte que la plataforma de paral·lelització pot ser utilitzada per diversos projectes o aplicacions proveïdores a la vegada. Per que cada aplicació quedi encapsulada amb els seus propis objectes, s'ha creat la classe project.

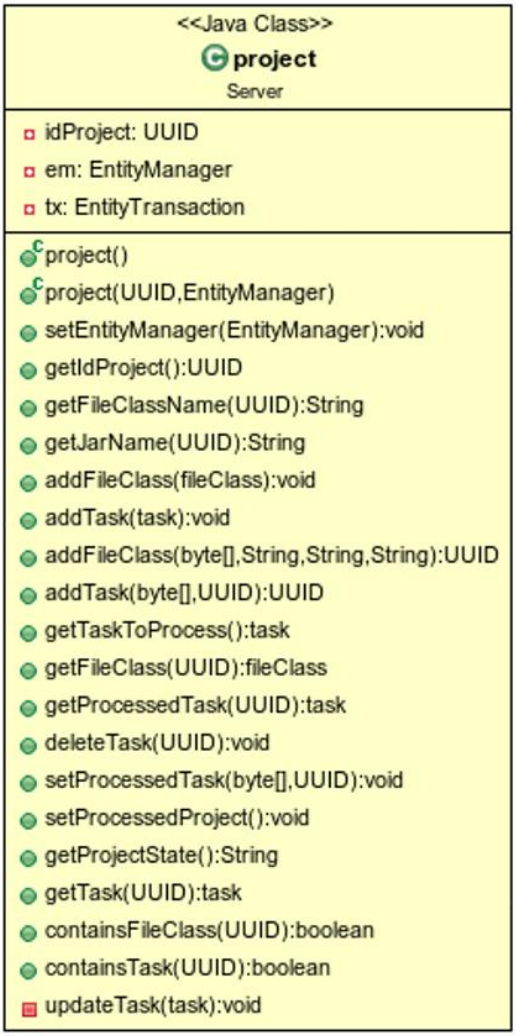


Figura 3-11. Classe project

La classe project representa l'aplicació proveïdora. Una aplicació proveïdora no pot utilitzar la plataforma si no té un identificador vàlid de projecte, aquest identificador pertany a una instancia concreta de la classe project. Si aquesta instancia no existeix o està en un estat no actiu l'aplicació proveïdora no podrà utilitzar la plataforma.

```
@Entity
public class project {
    @Id
    private UUID idProject;

    @Transient
    private HashMap<UUID,task> tasks;
    @Transient
    private HashMap<UUID, fileClass> fClasses;

    @Transient
    private contextProjectState cProjectState;
    @Transient
    private EntityManager em;
    @Transient
    private EntityTransaction tx =null;
```

Figura 3-12. Atributs de la Classe project

Com es mostra a la figura anterior**,** a la classe project es fa servir Hibernate per la persistència dels objectes. La classe project conté dues col·leccions, una que conté els objectes task del projecte i un altre que conte els objectes fileClass. Aquest objectes es detallen més endavant però s'ha de saber que els objectes task contenen els objectes que s'envien cap els clients per ser processats i els objectes fileClass contenen les classes a la que pertanyen aquests objectes.

La classe project conté els mètodes per gestionar directament els objectes task i fileClass. Alguns d'aquests mètodes s'han sobrecarregat per donar servei a la càrrega d'objectes des de persistència (nomes a l'inici de l'aplicació).

El mètode getTaskToProcess, retorna un identificador d'objecte task que conté un objecte per processar des de l'aplicació client. Per triar quin identificador d'objecte enviar, el mètode itera per tots els objectes Task continguts a la col·lecció tasks fins que troba un en estat pendent. En el cas que s'arribi al final de la iteració i no hagi trobat cap objecte task en estat pendent, es torna a iterar, però en aquest cas, es retorna el primer objecte task en estat "en procés". Això pot fer que varies aplicacions clients estiguin processant el mateix objecte a la vegada, però d'altra banda soluciona la possibilitat de que un objecte que s'ha enviat per processar a un client (i per tant té estat de "en procés), però per alguna raó mai ha estat retornat pel client, quedés sempre en estat "en procés". La col·lecció tasks que conté els objectes Task es del tipus LinkedHashMap. S'ha utilitzat aquest tipus de

col·lecció perquè respecta l'ordre d'inserció a l'hora d'iterar. Això fa que l'algoritme triï sempre el primer objecte task introduït en estat "pendent" o al primer en estat "en procés".

```
public task getTaskToProcess() throws serverException{
    Iterator<task> iterator;
    task tskTmp, tsk=null;

    //get the first task pending to process
    iterator = tasks.values().iterator();
    while (iterator.hasNext() && tsk==null) {
        tskTmp = iterator.next();
        if (tskTmp.getState().equals("Pending")){
            tskTmp.setInProcessState();
            tsk=tskTmp;
            //persistence
            updateTask(tsk);
        }
    }
    //if there are no pending tasks then get the first "In process" task.
    if (tsk==null) {
        iterator = tasks.values().iterator();
        while (iterator.hasNext()) {
            tskTmp = iterator.next();
            if (tskTmp.getState().equals("inProcess")){
                tsk=tskTmp;
            }
        }
    }
    return tsk;
}
```

Figura 3-13. Mètode getTaskToProcés

3.1.5.3 Classe FileClass

Els objectes que s'han de processar per part de les aplicacions clients o nodes, pertanyen a classes que aquestes aplicacions clients han de conèixer independentment de que totes implementin una interfície comú. Aquestes classes estan emmagatzemades en objectes FileClass.

S'ha optat per crear una classe pròpia per incorporar aquest arxius .class perquè un mateix arxiu .class pot ser utilitzat per varis objectes a processar, per tant, d'aquesta forma es soluciona un possible problema de redundància.

Els objectes fileClass pertanyen a objectes project, cada objecte fileClass té un identificador únic que els objectes task coneixen per poder fer l'associació entre ells.

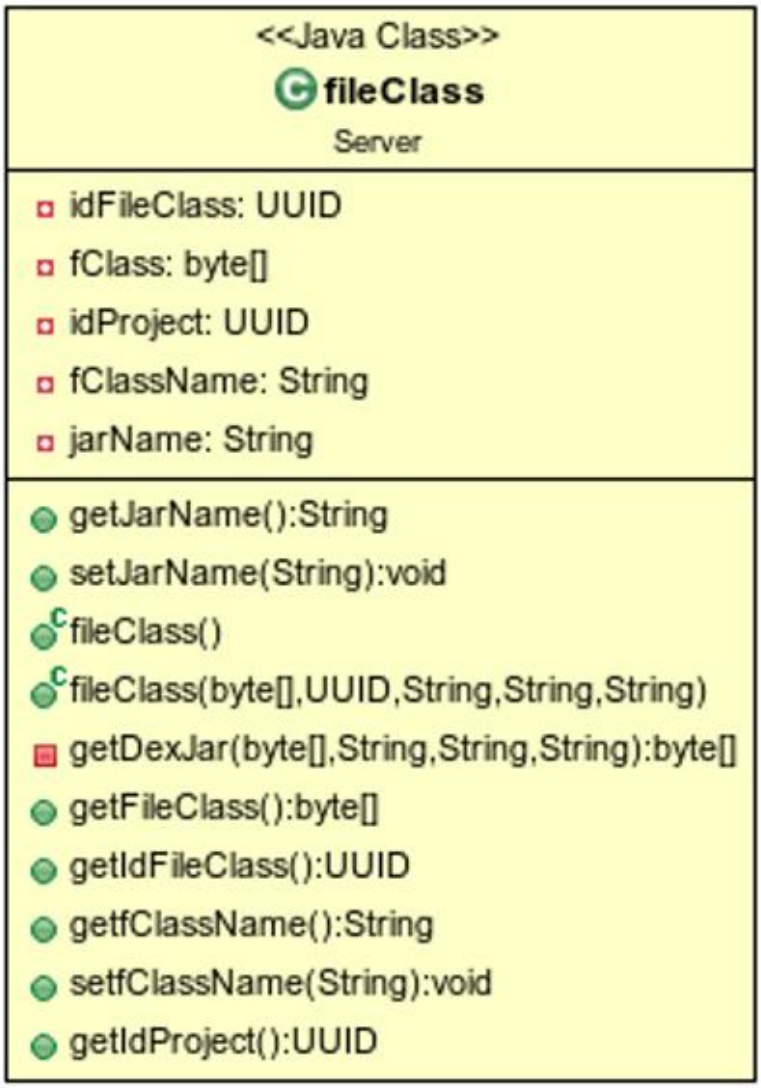


Figura 3-14. Classe fileClass

Dels atributs de la classe fileClass es pot destacar l'atribut fClass que es el que conté l'arxiu .class en format array de bytes.

Un problema que es troben les aplicacions clients, es la impossibilitat de que un sistema Android carregui dinàmicament una classe creada per un projecte Java. Això es degut a que Android utilitza una màquina virtual de Java diferent a la màquina virtual de Java dels sistemes no mòbils. Aquesta màquina virtual d'Android s'anomena Dalvik i codifica els

objectes de forma diferent a com ho fa la màquina virtual de Java. Per tant una classe codificada amb una màquina virtual de Java no és intel·ligible pels sistemes Android.

Per solucionar aquest problema la classe fileClass fa una conversió de l'arxiu .class que s'ha enviat des de la aplicació proveïdora, a un arxiu .dex amb codificació Dalvik nativa. Per fer aquesta conversió, s'utilitza la eina "dex" que està inclosa dins del SDK framework d'Android. Aquesta eina rep com a paràmetre l'arxiu .class que es vol convertir, i crea un arxiu .jar que conté l'arxiu .dex amb codi Dalvik natiu.

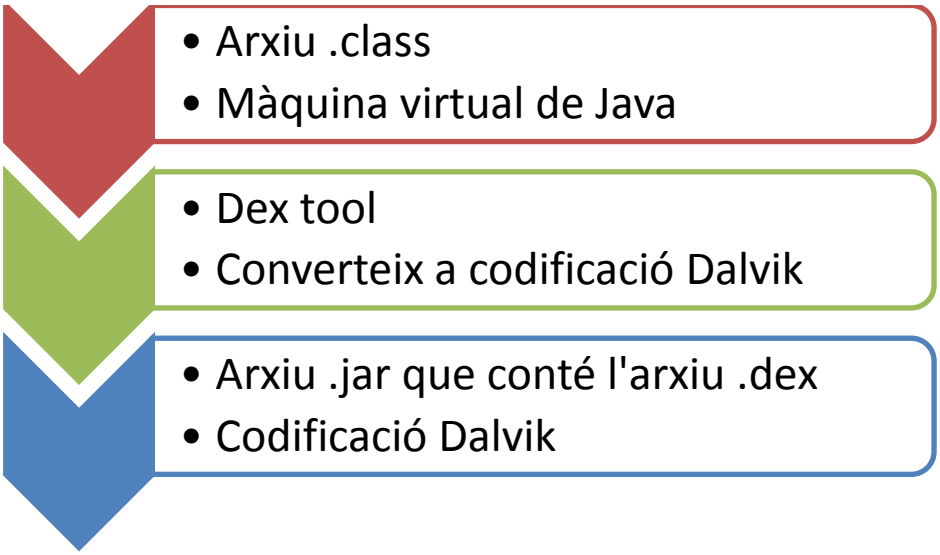


Figura 3-15. Conversió de codificació de màquina virtual de Java a màquina virtual Dalvik

Aquesta conversió es fa directament al constructor de la classe fileClass que és el que rep l'array de bytes amb l'arxiu .class.

```
//create jar with dx command line
String command = dexPath + "dx --dex --no-strict --output="
        + tmpPath +  classNameNoExt + ".jar " + tmpPath + className;

Process p;
p = Runtime.getRuntime().exec(command);
p.waitFor();
```

Figura 3-16. Codi per convertir de JVM a DVM.

El resultat de la conversió és un arxiu .jar que s'emmagatzema dins la classe fileClass en l'atribut fClass.

3.1.5.4 Classe Task

La classe task té com finalitat contenir els objectes que han de ser processats pels nodes o aplicacions clients, i els objectes ja processats.

Cada objecte task pertany a un projecte i a més coneix l'identificador del seu objecte fileClass respectiu.

Cada instancia té un identificador únic que s'utilitza al llarg de tot el domini, i les aplicacions clientes i proveïdores el fan servir per identificar-ho, fer peticions etc.

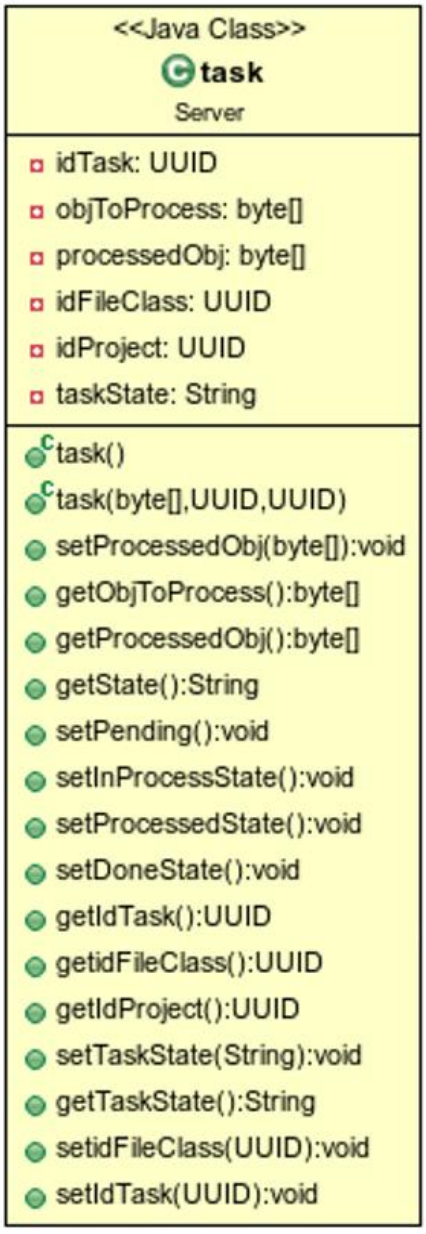


Figura 3-17. Classe task

Els atributs objToProcess i processedObj contenen l'objecte a processar per part dels clients i l'objecte processat respectivament.

El constructor s'ha sobrecarregat perquè es pugui utilitzar per crear nous objectes que provinguin de les aplicacions proveïdores o que provinguin de la persistència, en el cas del inici de l'aplicació.

Les tasques, igual que els projectes tenen associat un estat depenen del moment en el que es trobin dins l'execució del sistema. Aquests estats es controlen a partir de la implementació del patró estats que s'especifica en el següent apartat.

3.1.6 Patró estats

Per tal de controlar els diferents estats en els que un objecte task pot estar en un moment donat, s'ha implementat el patró estats.

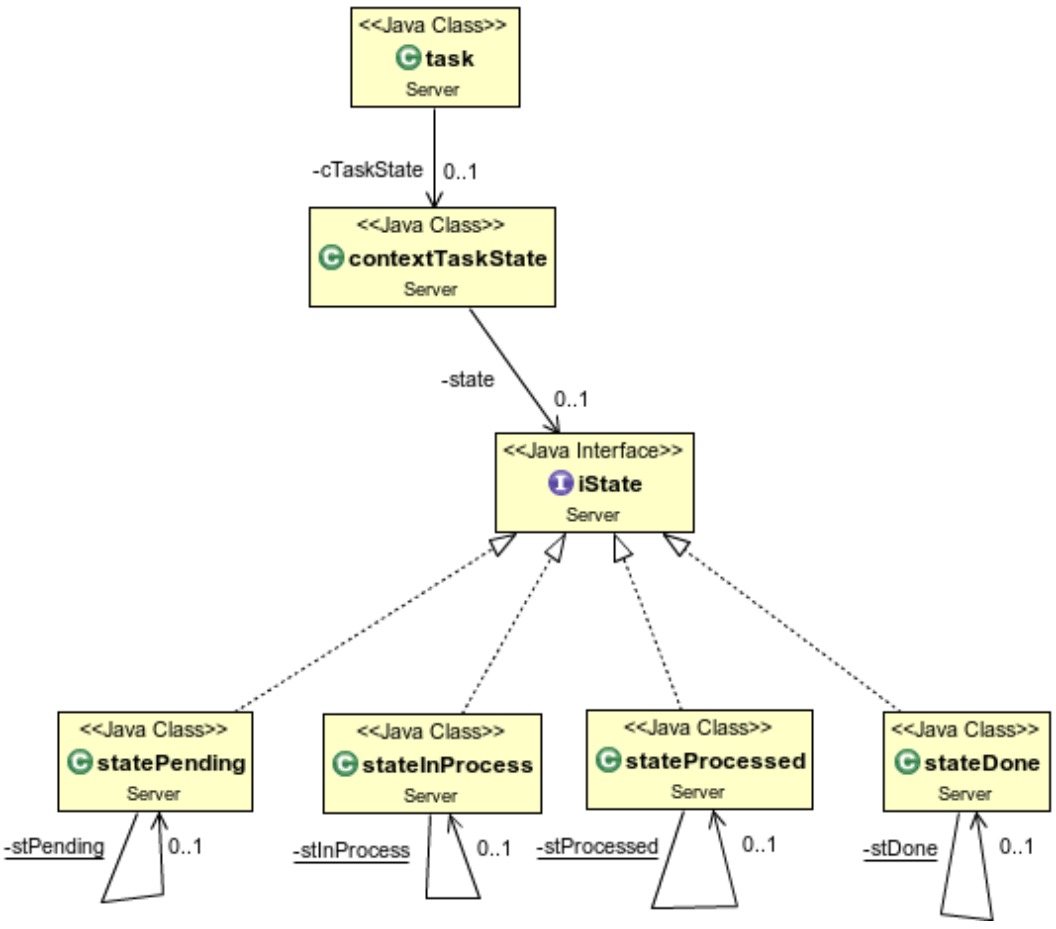


Figura 3-18. Patró estats

En el patró, els diferents estats són classes (statePending, stateInProcess, stateProcessed, stateDone) que implementen el patró Singleton perquè no hi ha la necessitat que existeixi més d'una instancia de cada una. Aquestes classes, implementen una interfície comú (iState). La gestió dels canvis d'estat es fa des d'una classe específica (contextTaskState) que conté un atribut del tipus de la interfície iState.

```
public class contextTaskState {
    private iState state;

    public contextTaskState(){}

    public void setPending() throws serverException{
        this.state= statePending.getStatePending();
    }

    public void setInProcess() throws serverException{
        if (state.getState().equals("Pending"))
            this.state= stateInProcess.getStateInProcess();
        else throw new serverException("Previous state is not 'Pending'");
    }

    public void setProcessed() throws serverException{
        if (state.getState().equals("inProcess"))
            this.state= stateProcessed.getStateProcessed();
        else throw new serverException("Previous state is not 'in Process'");
    }

    public void setDone() throws serverException{
        if (state.getState().equals("Processed"))
            this.state= stateDone.getStateDone();
        else throw new serverException("Previous state is not 'Processed'");
    }

    public String getState(){
        return state.getState();
    }
```

Figura 3-19. Classe contextTaskState

Els diferents mètodes de les classes són els encarregats de fer la transició d'un estat a l'altre i les comprovacions pertinents.

Les transicions entre estats i qui les provoca, es pot veure en detall a la figura següent**.**

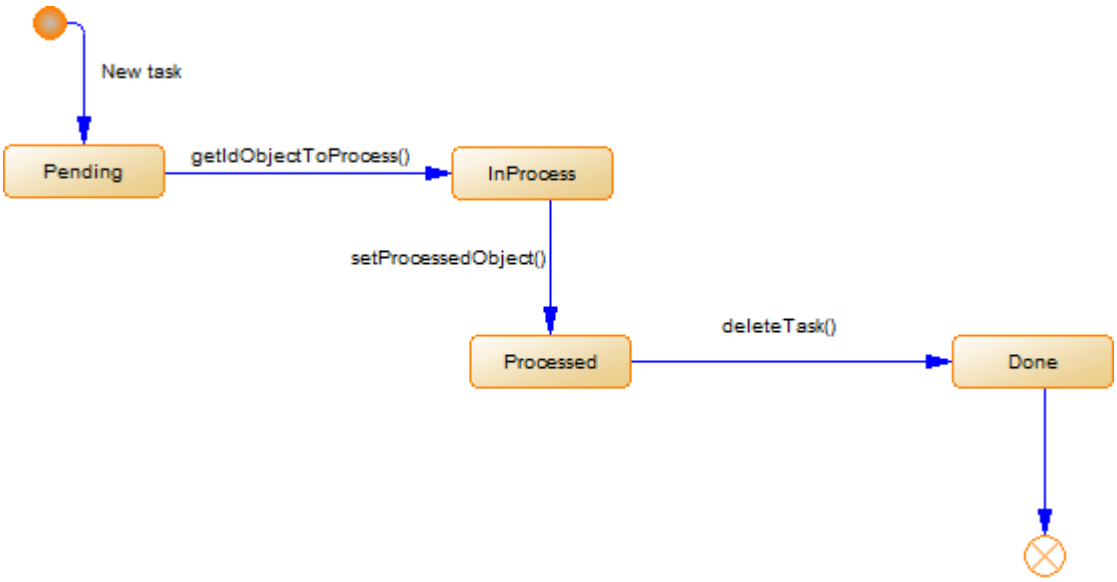


Figura 3-20. Diagrama d'estats dels objectes task.

3.1.7 Persistència

Per implementar la persistència s'ha utilitzat el framework Hibernate. En aquest projecte la persistència de les dades es fa servir únicament per tenir un històric de les operacions realitzades i per la recuperació del sistema a la mateixa situació que es trobava en el moment de finalitzar l'execució. A l'hora d'enviar els objectes cap les aplicacions clients o proveïdores, no es recuperen des de persistència, si no que directament es treballa amb els objectes que hi ha memòria. Això penalitza el consum de memòria però augmenta el rendiment en evitar accessos a disc.

Les classes que es persisteixen són project, fileClass i task. A continuació, com exemple, es mostra com s'ha indicat a Hibernate quins atributs s'han de persistir de la classe task.

```
@Entity
public class task {
    @Id
    private UUID idTask;


    private byte[] objToProcess;
    private byte[] processedObj;
    private UUID idFileClass;
    private UUID idProject;
    private String taskState;
    @Transient
    private contextTaskState cTaskState;
```

Figura 3-21. Mapatge per persistir la classe task.

El gestor de bases de dades triat per el projecte ha estat MySql perquè reunia les característiques tècniques necessàries per el projecte amb l'avantatge de ser amb llicència Open Source.

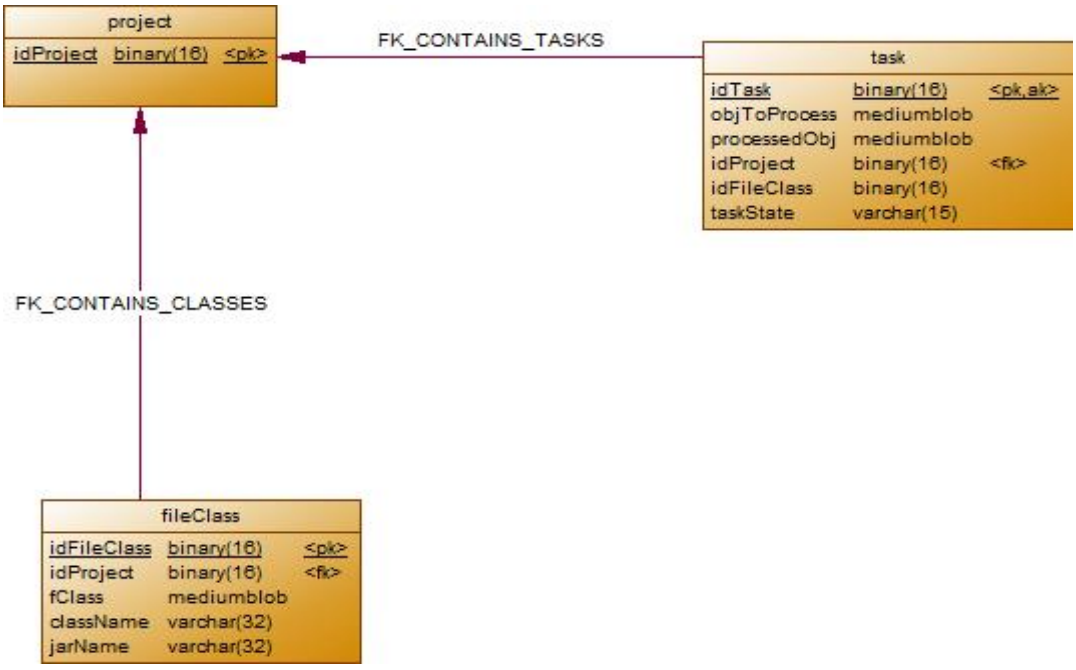


Figura 3-22. Disseny de les taules de la base de dades de l'aplicació.

3.2. Client

L'aplicació client forma part de la plataforma de computació distribuïda. És la part que fa la funció de node de processament. L'aplicació és un servei que s'executa en segon pla quan certs requisits es compleixen i deixa d'executar-se quan aquets requisits deixen de complir-se.

Aquesta aplicació ha estat implementada per a sistemes Android.

3.2.1 Android

Les versions utilitzades pel desenvolupament d'aquest projecte, es mostren a continuació.

Versió d'Android mínima	3.0 (Honeycomb)
Nivell d'API mínima	11
Versió d'Android	5.1.1 (Lollipop)
Versió d'API	22

Figura 3-23. Versions d'Android utilitzades

3.2.2 Paquets

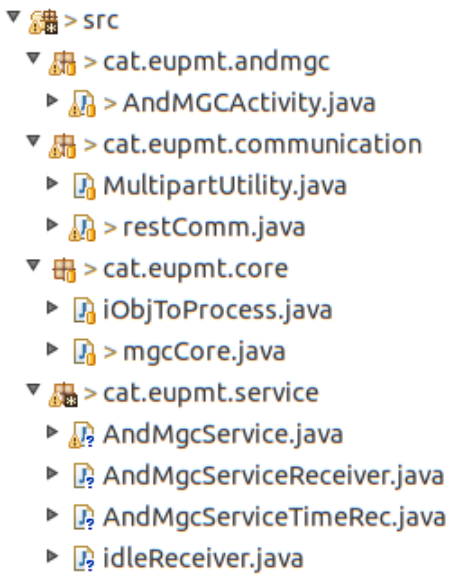


Figura 3-24. Estructura de paquets de l'aplicació

L'aplicació client s'ha estructurat en quatre paquets:

- *cat.eupmt.andmgc*: conté la classe *AndMGCActivity* que permet a l'usuari modificar els paràmetres de l'aplicació.
- *cat.eupmt.communication*: conté les classes relacionades amb les comunicacions amb el servidor.
- *cat.eupmt.core*: conté la classe que processa els objectes i la interfície que han de implementar. Per tal de poder carregar correctament els objectes que implementen la interfície, aquesta també a d'estar declarada a les aplicacions proveïdores dins del mateix paquet.
- *cat.eupmt.service*: conté les classes que implementen el subsistema de control de paràmetres que gestiona en quin moment s'ha de executar el processament d'objectes.

3.2.3. Domini

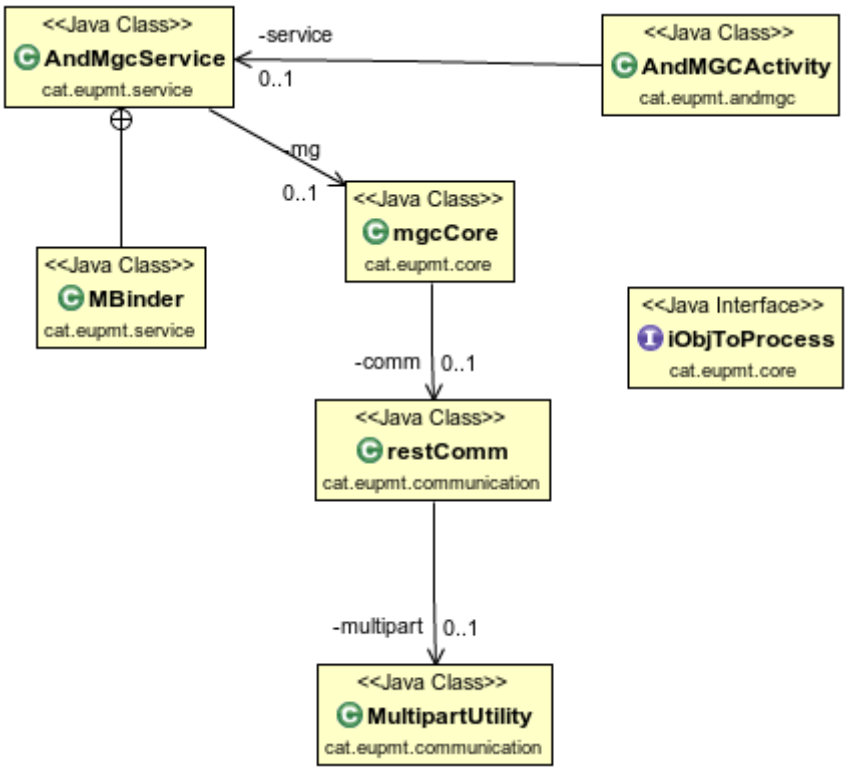


Figura 3-25. Diagrama de domini

L'aplicació consta d'un servei (*AndMgcService*) que controla el valor dels paràmetres que s'han de tenir en compte, perquè, en cas favorable, s'executi una instancia de la classe *mgcCore*, que és la responsable d'obtenir els objectes, processar-los i tornar-los de nou al servidor.

Totes les comunicacions amb el servidor es fan utilitzant la classe *restComm*.

La classe *MultipartUtility* es fa servir per donar el format necessari als enviaments de fitxers cap el servidor (format multipart).

AndMgcActivity es la classe que gestiona la modificació del valor dels paràmetres per part del usuari.

3.2.4 Comunicacions amb el servidor (restComm)

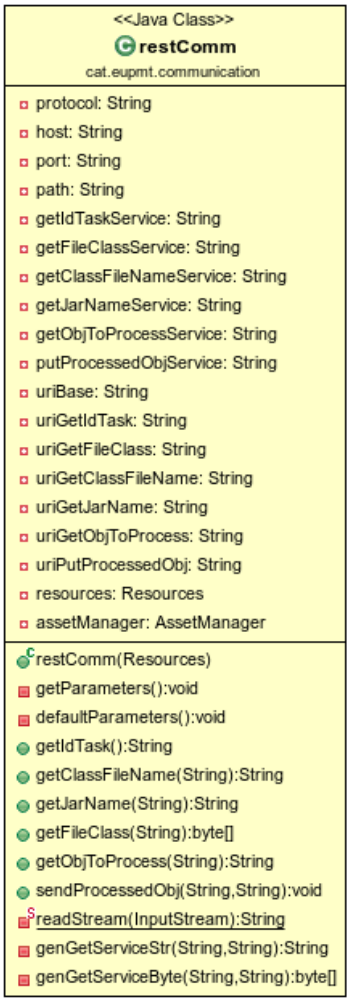


Figura 3-26. Classe restComm

Les comunicacions amb el servidor es fan utilitzant la classe *restComm*. Aquesta classe obté els valors dels seus atributs des d'un arxiu de text. D'aquesta manera, en cas necessari, es poden modificar els valors sense haver de modificar la codificació de la classe.

En aquest cas no s'ha utilitzat ni el framework *Jersey-client* ni el *Jersey-Multipart* ja que les seves versions actuals no estan suportades al 100% per la plataforma Android. Per solucionar-ho s'ha fet servir la classe abstracta *java.net.HttpURLConnection* i per donar suport a l'enviament de fitxers, fent servir el tipus de contingut Multipart, s'ha utilitzat la classe *MultipartUtility*.

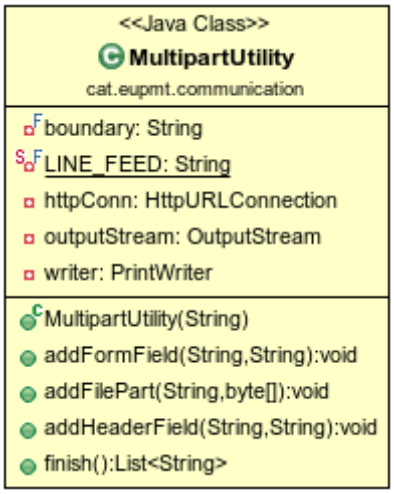


Figura 3-27. Classe multipartUtlity

3.2.5 Configuració de paràmetres (AndMGCActivity)

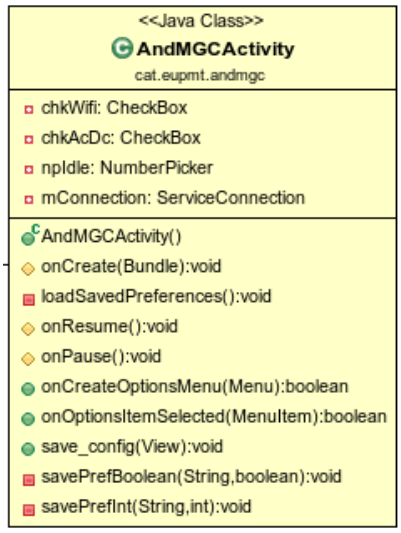


Figura 3-28. Classe *AndMGCActivity*

La classe *AndMgcActivty*, és l'encarregada de mostrar i modificar els paràmetres que s'han de tenir en compte a l'hora d'executar el subsistema de processament d'objectes. La classe *AndMgcActivty* fa servir la classe *SharedPreferences* per desar i recuperar els valors dels paràmetres. Una vegada desats, la classe informa dels canvis al servei *AndMgcService* per que els tingui en compte.

Figura 3-29. Pantalla de modificació de paràmetres

3.2.6 Servei de control de idle i paràmetres (AndMgcService)

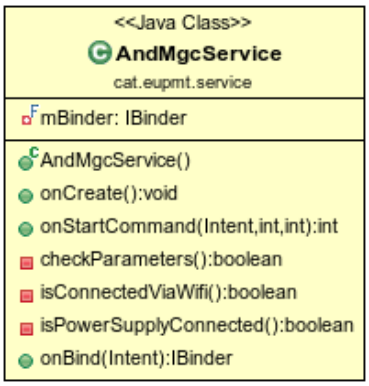


Figura 3-30. Classe AndMgcService

El control de les característiques que s'han de complir per que s'executi el procés de processament d'objectes, es fa utilitzant el servei implementat a la classe *AndMgcService*.

L'aplicació està dissenyada per fer us del "Volunteer Computing", es a dir, aprofitar els recursos (com el processador del dispositiu), durant el temps que aquest està en un estat de repòs (idle). Per detectar quan el dispositiu ha entrat en estat de repòs, es fa servir una classe que estén de *Broadcast Receiver*.

Un objecte *Broadcast Receiver*, capta esdeveniments del sistema, i en aquest cas, es fa servir per detectar quan la pantalla del dispositiu entra en estat d'inactivitat.

```
public class idleReceiver extends BroadcastReceiver {

    private boolean scrOff;

    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals(Intent.ACTION_SCREEN_OFF)) {
            scrOff = true;
        } else if (intent.getAction().equals(Intent.ACTION_SCREEN_ON)) {
            scrOff = false;
        }
        Intent i = new Intent(context, AndMgcService.class);
        i.putExtra("idle_state", scrOff);
        context.startService(i);
    }

}
```

Figura 3-31. Broadcast Receiver

Una vegada que el servei detecta que el dispositiu està en estat d'inactivitat, passa a comprovar si es compleixen les condicions informades en els paràmetres de configuració i en cas afirmatiu, executa en un fil apart un objecte de la classe *mgcCore*. En el moment que el servei detecta que el dispositiu deixa d'estar en estat de repòs finalitza l'execució del fil.

```
@Override
public int onStartCommand (Intent intent, int flags, int startId) {
    boolean idle = intent.getBooleanExtra("idle_state", false);
    if (idle) {
        //check parameters
        boolean paramOK = checkParameters();
        if (paramOK){
            if (!mg.isAlive()){
                //start running
                Resources resources = this.getResources();
                Context context =  getApplicationContext();
                mg = new mgcCore(resources, context);
                mg.start();
            }
        }
    } else{
        if (mg.isAlive()){
            //stop running process
            mg.stop();
        }

    }

    return Service.START_NOT_STICKY;
}
```

Figura 3-32. Mètode onStartCommand de la classe AndMgcService

3.2.7 Processament d'objectes (mgcCore)

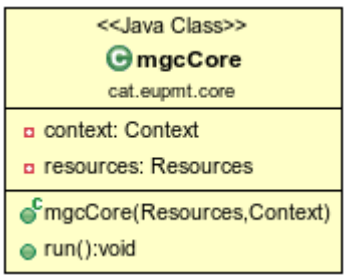


Figura 3-33. Classe mgcCore

La classe *mgcCore* és la encarregada de demanar els objectes al servidor, processar-los i retornar-los de nou al servidor. A continuació es mostra un esquema de les funcions que realitza la classe *mgcCore*.

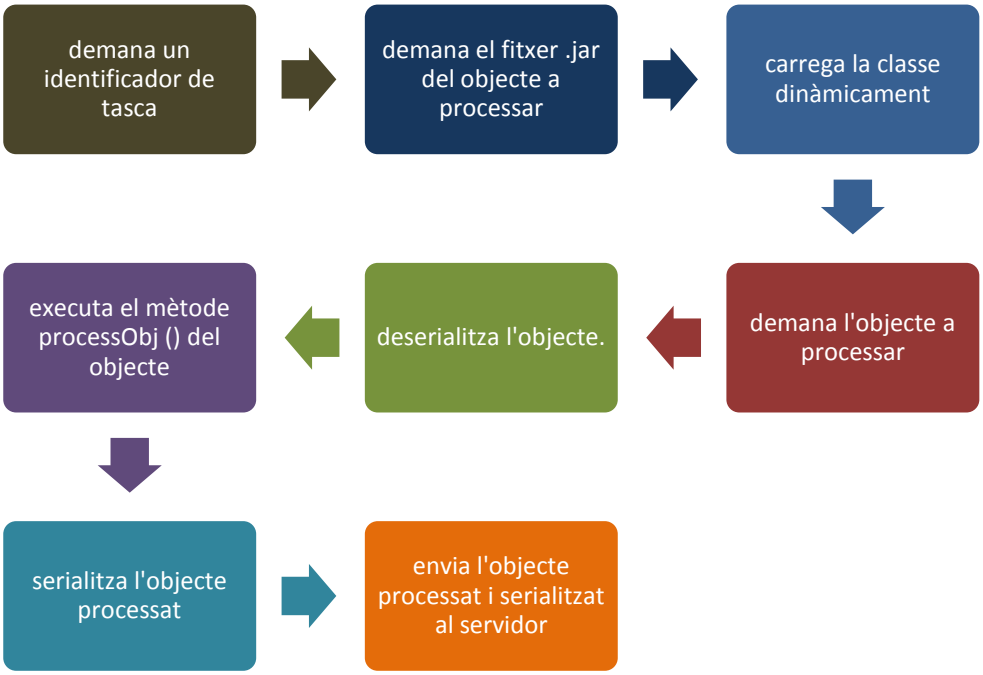


Figura 3-34. Esquema de funcions a realitzar per la classe mgcCore

En l'esquema anterior, es pot apreciar que una vegada s'ha obtingut l'objecte .jar que conté la classe del objecte que es processarà, es carrega aquesta classe dinàmicament. Per poder fer-ho, es necessari que la classe que arribi, estigui en format ".dex" i compilada per una màquina virtual Dalvik, com ja s' explicat en l'apartat del servidor.

Una vegada s'ha obtingut la classe, es desa a disc i es carrega tal i com es mostra en la següent figura.

```
//load class from file
DexClassLoader classLoader = new DexClassLoader(
    clsFile.getAbsolutePath(), MGCTmp.getAbsolutePath(), null, getClass().getClassLoader());

Class<?> MGCClass=null;
try {
    MGCClass = classLoader.loadClass("cat.eupmt.core." + clsName);
} catch (ClassNotFoundException e1) {
    e1.printStackTrace();
}
```

Figura 3-35. Codi per carregar la classe dinàmicament

A continuació s'obté l'objecte que es processarà. L'objecte a processar, implementa la interfície *iObjToProcess* que declara el mètode *processObj()*. Això permet a l'aplicació executar l'objecte sense haver de conèixer la lògica que implementa, i per tant, permet a la aplicació processar objectes de diferents orígens d'una manera totalment transparent.

L'objecte que arriba des del servidor, ho fa serialitzat (des de l'aplicació proveïdora). La serialització en origen, es fa utilitzant la biblioteca Gson de Google. Per tant, el primer pas és deserialitzar l'objecte utilitzant la mateixa biblioteca. A continuació es mostra el codi on es deserialitza l'objecte, i s'executa el seu mètode processObj().

```
//load object
Gson gson = new Gson();
iObjToProcess obj = (iObjToProcess) gson.fromJson(JsonData, MGCClass);

//execute object
obj.processObj();
```

Figura 3-36. Codi per carregar i processar l'objecte

Una vegada processat, l'objecte es torna a serialitzar i es retorna al servidor.

3.3. Proveïdor

Per comprovar el funcionament i l'eficiència de la plataforma de computació distribuïda sobre dispositius mòbils, s'ha implementat una aplicació que té el rol d'aplicació proveïdora, i que resol una tasca freqüent del camp de la Bioinformàtica. L'aplicació que s'ha implementat, realitza l'alineació múltiple de seqüències d'ADN. Per aconseguir-ho s'ha paral·lelitzat el primer pas de l'algoritme de Feng and Doolittle que ha estat detallat anteriorment en aquesta memòria.

3.3.1 Paquets

Aquest subprojecte, s'ha dividit en els següents paquets:

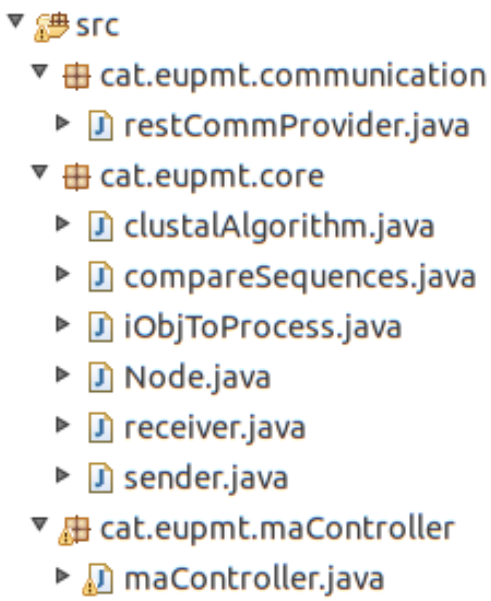


Figura 3-37. Estructura de paquets de l'aplicació

- *cat.eupmt.communication*: conté una única classe que s'utilitza per establir les comunicacions amb la plataforma de computació distribuïda.
- *cat.eupmt.core*: conté les classes que s'encarreguen d'implementar l'algoritme d'alineament de seqüències. Aquest paquet és especialment important ja que conté la interfície que implementen els objectes que s'envien a processar. Aquesta mateixa interfície, ha d'existir en la aplicació client del dispositiu mòbil, dins del mateix paquet, perquè pugui ser importada dinàmicament per l'aplicació.
- *cat.eupmt.maController*: conté la classe que implementa el "servlet" que fa d'interfície de comunicació amb l'usuari.

3.3.2 Domini

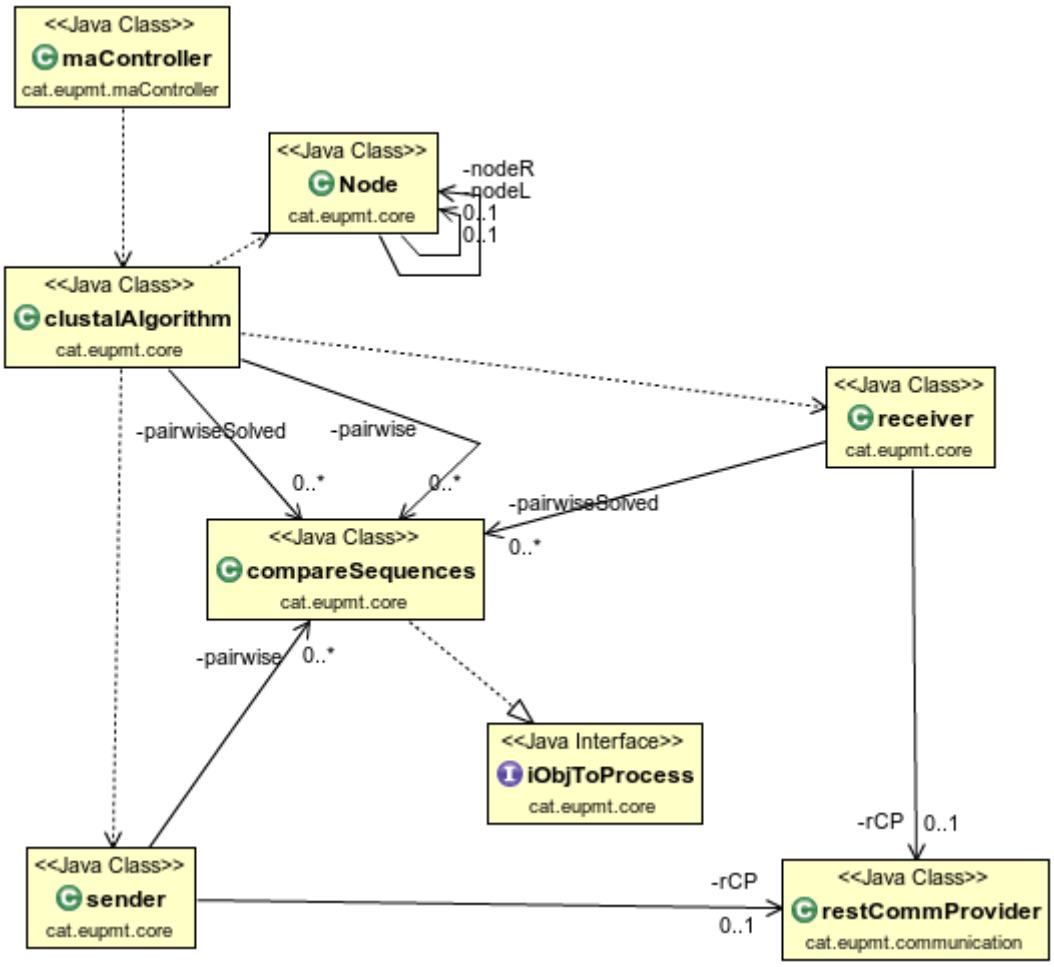


Figura 3-38. Diagrama de domini

Aquest diagrama de domini mostra la interdependència de les classes utilitzades en aquest subprojecte.

L'usuari de l'aplicació inicia el procés enviant un arxiu que conté les seqüències que es volen alinear mitjançant un formulari web creat amb aquest propòsit. La classe *maController* és l'encarregada de rebre aquest arxiu, comprovar el seu format i enviar-li a la classe *clustalAlgorthm*.

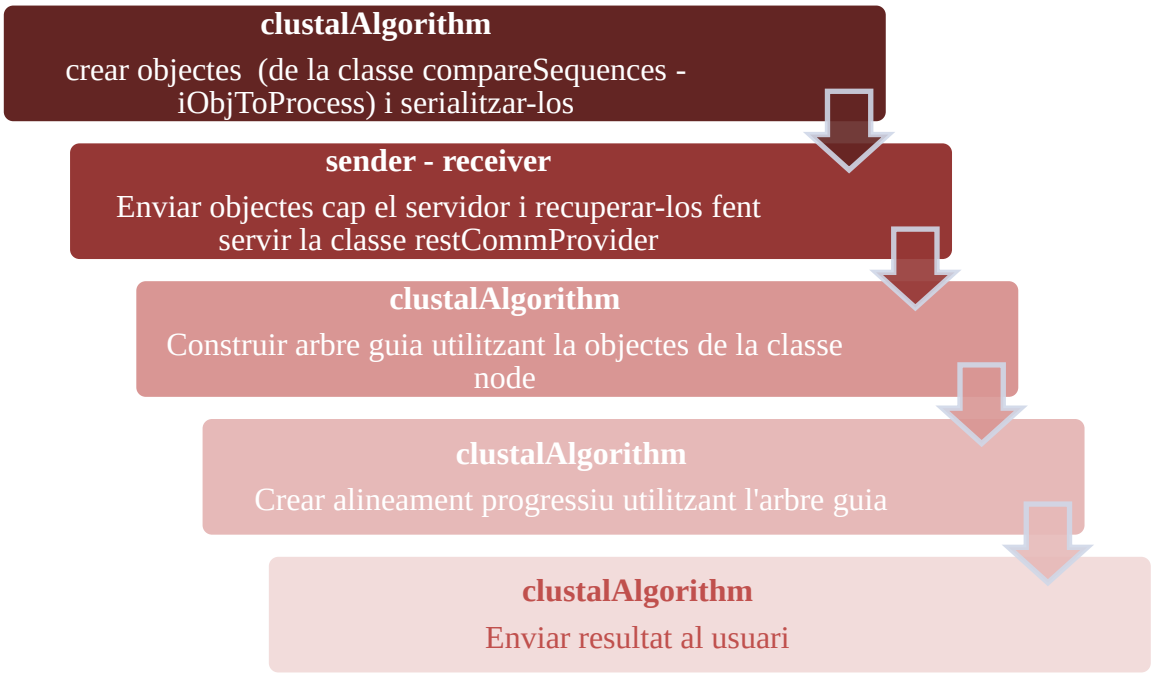


Figura 3-39. Esquema de processos

La classe *clustalAlgorithm* crea els objectes de la classe *compareSequences* (que implementen la interfície *iObjToProcess*) que són serialitzats i enviats a la plataforma de computació distribuïda utilitzant la classe *sender*. La classe *receiver* s'encarrega de recuperar els objectes que ja han estat processats, i convertir-los en objectes Java. Les classes *sender* i *receiver* utilitzen la classe *restCommProvider* per comunicar amb la plataforma de computació distribuïda. Una vegada tots els objectes han estat enviats i rebuts per les classes *sender* i *receiver*, la classe *clustalAlgorithm* construeix l'arbre guia (explicat en l'apartat 2.4.5 d'aquesta memòria) utilitzant objectes de la classe *node*. Després, s'utilitza aquest arbre per crear l'alineament definitiu i per últim s'envia el resultat a l'usuari.

3.3.3 Inici de la execució (maController)

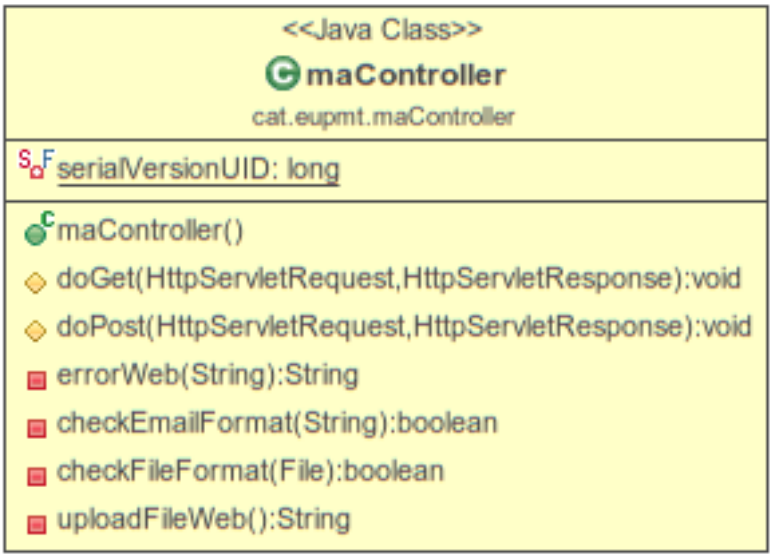


Figura 3-40. Diagrama de domini

La classe *maController* s'encarrega de la interacció amb l'usuari i de executar el procés d'alineament.

L'usuari a traves d'una interfície web creada en el mètode *uploadFileWeb*, envia l'arxiu que conté les seqüències a alinear i l'adreça d'e-mail on vol rebre els resultats. Aquestes dades són rebudes pel mètode *doPost* i a continuació comprova el format de l'arxiu rebut (format Fasta) i el format de l'adreça d'e-mail. Aquestes comprovacions es duen a terme en els mètodes *checkFileFormat* i *checkEmailFormat* respectivament. En cas que les comprovacions siguin satisfactòries, s'envien les dades a la classe *clustalAlgorithm* per que les processi. L'objecte *clustalAlgorithm* s'executa en un fil a part per que l'aplicació pugui continua treballant paral·lelament.

```
//send file and email to be processed
clustalAlgorithm cA = new clustalAlgorithm(fastaFile, email);
cA.run();
```

Figura 3-41. Creació i execució de l'objecte *clustalAlgorithm*

En el cas que les comprovacions dels formats de l'arxiu o de l'e-mail no siguin satisfactòries es crea una plana web utilitzant el mètode *errorWeb* on es mostra la incidència a l'usuari.

3.3.4 Comunicacions amb el servidor (restCommProvider)

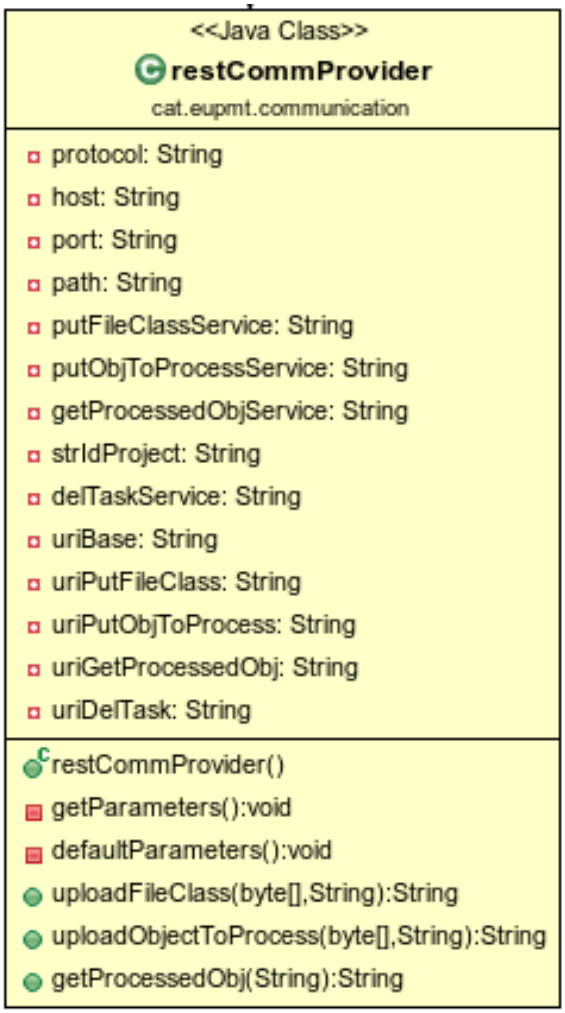


Figura 3-42. Classe *restCommProvider*

La classe *restCommProvider* és l'encarregada de les comunicacions amb la plataforma de computació distribuïda.

La classe utilitza el mètode *getParameters* per carregar el valor dels atributs des d'un arxiu de text on estan especificats els valors. D'aquesta manera es facilita la modificació dels paràmetres sense la necessitat de modificar la codificació de la classe. En el cas que aquests valors no es poguessin carregar, els atributs obtenen uns valors per defecte per mitjà del mètode *defaultParameters*. Els mètodes *uploadFileClass* i *uploadObjecteToProcess*, envien l'arxiu ".class" i els objectes per processar a la plataforma, respectivament. La classe *getProcessedObject* recupera, des de la plataforma, un objecte ja processat.

```
//upload object to process
public String uploadObjectToProcess(byte[] bytes, String idFClass){
    Client client = Client.create();
    WebResource webResource = client.resource(uriPutObjToProcess);
    ByteArrayInputStream ObjStream = new ByteArrayInputStream(bytes);
    FormDataMultiPart ObjmultiPart = new FormDataMultiPart();
    ObjmultiPart.field("idProject", strIdProject);
    ObjmultiPart.field("idFileClass", idFClass);
    FormDataBodyPart bodyPart = new FormDataBodyPart("file",
            ObjStream, MediaType.APPLICATION_OCTET_STREAM_TYPE);
    ObjmultiPart.bodyPart(bodyPart);
    ClientResponse ObjResp = webResource.type(
            MediaType.MULTIPART_FORM_DATA_TYPE).put(
                    ClientResponse.class, ObjmultiPart);
    String idTask="";
    if (ObjResp.getStatus()==200){
        idTask = ObjResp.getEntity(String.class);
    }
    return idTask;
}
```

Figura 3-43. Mètode uploadObjectToProcess

Per tal de simplificar les comunicacions amb el servidor de serveis Web de la plataforma, s'ha fet servir el framework Jersey-Client (v. 1.19) i el framework Jersey-multipart (v. 1.19).

3.3.5 Algoritme d'alineament múltiple de seqüències (clustalAlgorithm)

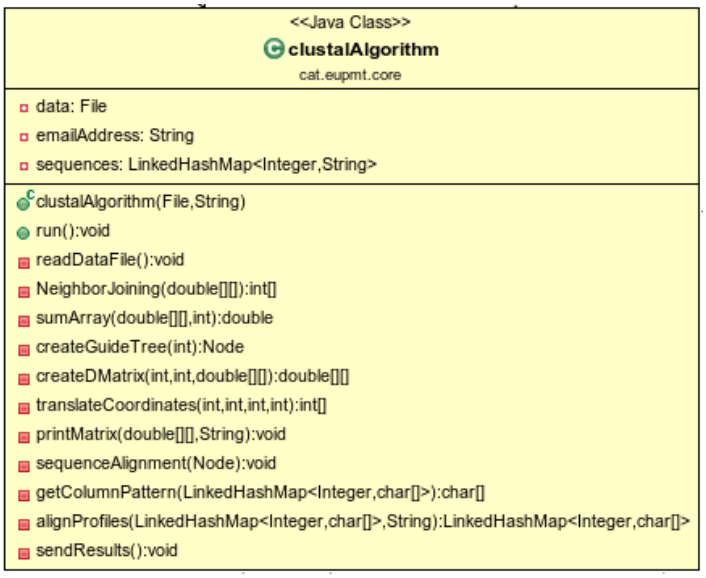


Figura 3-44. Classe *clustalAlgorithm*

Aquesta classe implementa l'algoritme per l'alineament múltiple de seqüències. Com s'explicat en apartats anteriors, aquest algoritme consta de tres parts. La primera, que fa alineaments de parells de seqüències, la segona, que construeix un arbre guia a partir dels

alineaments de la primera part i la tercera, que utilitza l'arbre guia per fer un alineament progressiu de les seqüències.

La classe *clustalAlgorithm* estén de la classe *Thread* per que es pugui executar en un fil apart.

Només la primera part de l'algoritme s'ha paral·lelitzat i és la part on es construeixen els objectes que s'enviaran a la plataforma.

```
LinkedHashMap<String, Integer> uploadedObj=new LinkedHashMap<String, Integer>();
//extract and number each sequence to a map
readDataFile();
//create objects to compare pairwise
int cont;
int pairCont=0;
for (int i=1;i<=sequences.size();i++){
    cont=i+1;
    for (int b=cont;b<=sequences.size();b++){
        //create objects to process
        compareSequences cs = new compareSequences();
        cs.setSeq1(sequences.get(i));
        cs.setSeq2(sequences.get(b));
        cs.setNumSeq1(i);
        cs.setNumSeq2(b);
        pairCont++;
        pairwise.put(pairCont, cs);
    }
}
```

Figura 3-45. Creació d'objectes *compareSequences*.

A la figura anterior es mostra el codi per crear els objectes de la classe *compareSequences* que s'enviaran a la plataforma de computació distribuïda. Primer, s'extrauen les seqüències de l'arxiu Fasta, s'enumeren i s'incorporen a la col·lecció de seqüències (*sequences*). A continuació es creen els objectes de la classe *compareSequences* aparellant les seqüències i s'emmagatzemen en una altra col·lecció (*pairwise*)

Una vegada creats els objectes i emmagatzemats, s'executa en un fil diferent un objecte sender per enviar els objectes. Després, i també en un fil diferent, s'executa un objecte *receiver* que s'encarrega de recuperar els objectes una vegada ja han estat processats. Per tal de saber quins objectes pot intentar recuperar l'objecte *receiver*, l'objecte sender i l'objecte *receiver* comparteixen una col·lecció on l'objecte sender informa dels objectes que s'han enviat amb èxit.

Quan els dos fils han acabat la seva execució, es construeix l'arbre guia. Això es fa mitjançant el mètode crateGuideTree. Una vegada s'ha construït l'arbre, aquest es recorre per fer un alineament progressiu. Aquesta acció es porta a terme al mètode *sequenceAlignment*.

Per últim el resultat de l'alineament s'envia a l'usuari utilitzant el mètode *sendResults*.

3.3.6 Els objectes a processar (compareSequences)

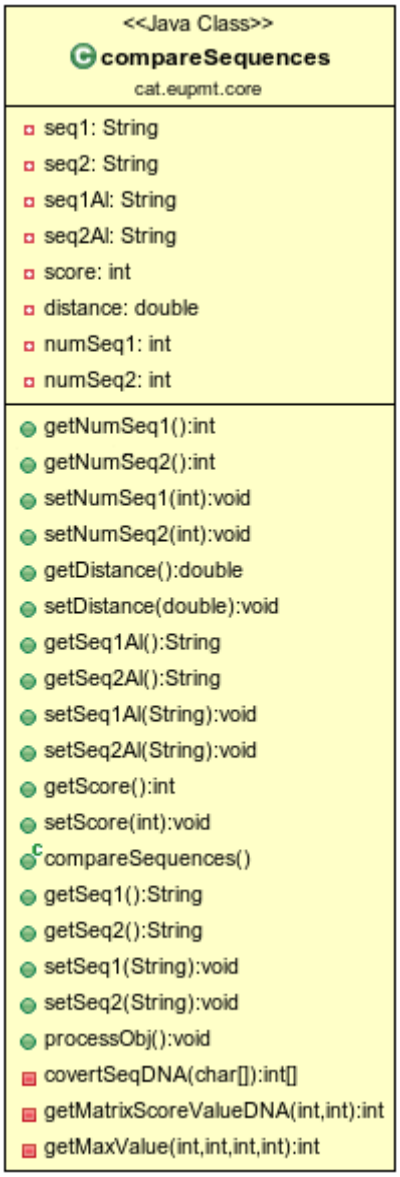


Figura 3-46. Classe *compareSequences*

Els objectes de la classe *compareSequences*, són els objectes que s'envien a la plataforma de computació distribuïda. La classe *compareSequences*, implementa la interfície *iObjToprocess* que conté el mètode *processObj*, Aquest mètode és el que s'executa en l'aplicació client per processar l'objecte, per tant, tota la lògica necessària per implementar l'algoritme de alineament de parells de seqüències (Needleman –Wunsch), està encapsulada a partir d'aquest mètode. D'aquesta manera s'aconsegueix que la plataforma de computació distribuïda executi objectes de diferents aplicacions proveïdores, sense conèixer la lògica de processament que conté l'objecte.

```
@Override
public void processObj() {
    //0=A;G=1;C=3;T=4

    char[] chSq1 = this.seq1.toCharArray();
    char[] chSq2 = this.seq2.toCharArray();
    int [] intSq1 = covertSeqDNA(chSq1);
    int [] intSq2 = covertSeqDNA(chSq2);
    int DistanceMatrix[][]= new int [chSq2.length + 1][chSq1.length + 1];
    double lenCont=0.;
    double matchCont=0.;

    //get lenCont
    if (chSq1.length<chSq2.length){
        lenCont=chSq1.length;
    }else{
        lenCont=chSq2.length;
    }

    //fill the first row and the first column
    int val=0;
    for (int col=0;col<(chSq1.length + 1);col++){
```

Figura 3-47. Porció de codi del mètode *processObj*

3.3.7 Enviament d'objectes (Sender)

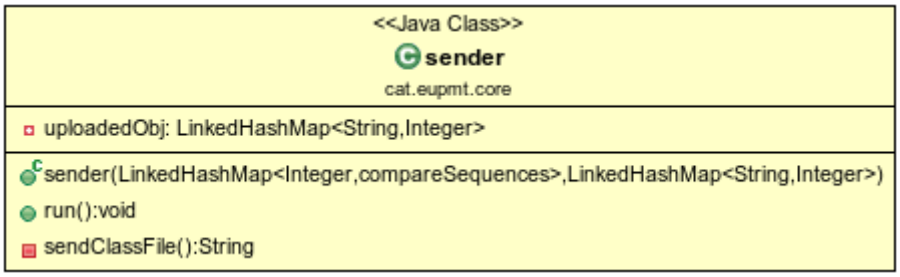


Figura 3-48. Classe *sender*

La classe sender implementa la interfície *Runnable* per poder-se executar com un fil a part. La seva funció és enviar els objectes de la classe *compareSequences*, creats per la classe *clustalAlgorithm*, cap el servidor de la plataforma per que siguin processats.

Aquesta classe, rep com paràmetres dues col·leccions del tipus *LinkedHashMap*, la primera conté els objectes a enviar i la segona, té la funció de guardar els identificadors dels objectes enviats correctament perquè, la calsse *receiver*, els pugui reclamar al servidor.

Els objectes de la classe *compareSequences*, per poder ser enviats cap el servidor, i que després les aplicacions clients els processin, es serialitzen. Aquesta serialització es fa utilitzant la biblioteca de codi obert Gson de Google. Aquesta biblioteca permet la conversió d'objectes Java a format Json i viceversa d'una manera senzilla. Per fer-ho utilitza els mètodes *toJson()* i *fromJson()*. En aquest cas la classe *sender* utilitza el mètode *toJson()* per serialitzar l'objecte i l'aplicació client de la plataforma fa servir el mètode *formJson()* per tornar a convertir-ho en un objecte Java abans de processar-ho.

```
//send objTo process
Set<Integer> keys = pairwise.keySet();
while (numElements<pairwise.size() && numTries<paramNumTries){
    numTries++;
    for(int k:keys){
        compareSequences cS = pairwise.get(k);
        //use Gson to serialize the object
        Gson gson = new Gson();
        String obj = gson.toJson(cS);
        byte[] bytes = obj.getBytes();
        String idTask=rCP.uploadObjectToProcess(bytes, idFileClass);
        //if ok mark it for download thread
        if (!idTask.equals("")){
            uploadedObj.put(idTask, k);
            numElements++;
        }
    }
}
```

Figura 3-49. Utilització de Gson

3.3.8 Recepció d'objectes processats (Receiver)

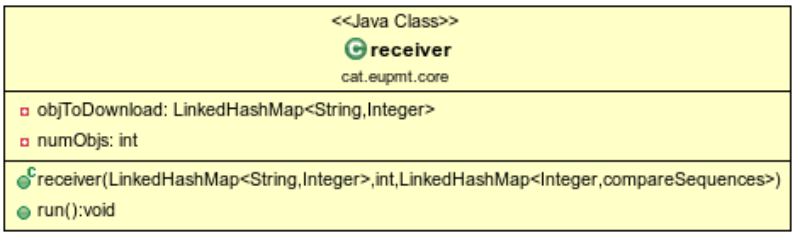


Figura 3-50. Classe *receiver*

Al igual que la classe *sender*, la classe *receiver* també implementa la interfície *Runnable* per que es pugui executar com un fil a part. La raó de fer-ho d'aquesta manera, és no haver d'esperar a que s'hagin enviat tots els objectes, per començar a rebre els que ja han estat processats.

La classe *receiver* rep dos paràmetres, que són dues col·leccions del tipus *LinkedHashMap*. Una es comparteix amb l'objecte sender i és on es troben els identificadors dels objectes que es poden demanar al servidor de la plataforma i l'altre és on es guarden els objectes processats.

Una vegada un objecte processat ha estat rebut des del servidor, es procedeix a la seva conversió en objecte Java utilitzant el mètode *fromJson()* de la biblioteca Gson de Google, tal i com s'ha explicat en l'apartat anterior.

```
Set<String> strKey = TMP_objToDownload.keySet();
for(String idTask : strKey) {
    //check if is already download
    int id = TMP_objToDownload.get(idTask);
    if (!pairwiseSolved.containsKey(id)){
        //download the obj
        String JsonData=rCP.getProcessedObj(idTask);
        if (!JsonData.equals("")){
            //load object
            Gson gson = new Gson();
            compareSequences cs = (compareSequences) gson.fromJson(JsonData, compareSequences.class);
            pairwiseSolved.put(id, cs);
            cont++;
        }else{
            try {
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

Figura 3-51. Recepció i conversió dels objectes processats

4.Conclusions i futures ampliacions

Donada la gran proliferació de dispositius mòbils en el mon actual (més de 7,5 bilions de dispositius), i tenint en compte la creixent innovació tecnològica d'aquests dispositius, aquest projecte ha volgut treure profit de tot plegat.

Amb el desenvolupament d'aquest projecte, es demostra que és possible construir un sistema de computació distribuïda utilitzant la potencia dels dispositius mòbils com nodes de computació.

El projecte s'ha fet enfocat clarament cap el camp de la investigació, intentant oferir un sistema de computació distribuïda i lliure i de baix cost. Els sistemes actuals de computació distribuïda, normalment impliquen grans infraestructures que comporten grans costos. Aquest projecte podria oferir una alternativa de cost reduït en certs projectes concrets. Com és el cas d'alineament de seqüències d'ADN.

El projecte ha estat creat amb la filosofia de "volunteer computing", per tant la seva viabilitat està directament relacionada amb l'expansió del projecte a traves dels usuaris de dispositius mòbils. També la seva eficàcia, en termes de rendiment de la plataforma, està directament relacionada amb el número d'instal·lacions de l'aplicació en aquests dispositius. Això implica, que la plataforma serà més eficaç quan més gran sigui en numero de nodes existents.

La selecció de la plataforma Android per implemetar la part client d'aquest projecte ha estat propiciada principalment per dues raons.

D'una banda, la gran proliferació de dispositius amb aquest sistema operatiu el converteix en l'idoni per l'expansió de l'aplicació i per tant, per el creixement potencial en termes d'eficàcia i rendiment, tal i com s'ha explicat anteriorment.

D'altra banda, Android és open source, el que fa que no hagin tants impediments en temes legals, comercials o de llicencia i a la vegada ofereix una amplia API extensament documentada.

La plataforma Android també ha suposat un punt extra de complexitat en el projecte. Principalment en els temes de carregar classes dinàmicament en temps d'execució i en la serialització i deserialització d'objectes.

En un principi, el disseny estava enfocat per utilitzar el mecanisme RMI de Java on un objecte es pot exportar per que es pugui accedir a ell a través de la xarxa, però actualment Android no suporta RMI. Això va portar a descartar la idea inicial d'utilitzar RMI i valorar la utilització de serialitzar i deserialitzar els objectes. Les proves de serialització en Java i deserialització en Java per Android, tampoc van tenir èxit, ja que les funcions de Java estàndard per la deserialtzació d'objectes obliguen a conèixer la classe del objecte a priori i no la reconeixen si ha estat carregada dinàmicament en temps d'execució.

D'altra banda com ja s'ha explicat en aquesta memòria, Android utilitza una màquina virtual pròpia per compilar el codi font de Java (DVM), i això ha obligat a convertir els fitxers de classe, creats per la màquina virtual de Java (JVM) en el format .dex abans de que es puguin incorporar a l'aplicació d'Android.

Al final per solucionar els problemes de la serialització d'objectes s'ha fet servir la biblioteca de codi obert de Google, GSON. Aquesta biblioteca ofereix la serialitazció i deserialització d'objectes Java a la seva representació i notació JSON i és totalment compatible amb sistemes Android.

Tot això ha fet que la planificació inicial del projecte es veies modificada. Els temps d'arquitectura i disseny, implementació i proves, s'han vist incrementats en un 25%.

Aquest projecte pretén ser un punt de partida per explorar les possibilitats de la computació distribuïda sobre dispositius mòbils. Tenint això en ment, les millores possibles sobre el projecte actual són innumerables.

En la part que en aquest projecte s'ha anomenat client, es podrien afegir paràmetres de control, com per exemple, el control de la temperatura del processador dels dispositius, el control del nivell de bateria. També es podria modificar l'algoritme per treure profit de processadors multi-core, paral·lelitazant les tasques rebudes en els diferents nuclis del processador, etc.

En la part del servidor, les millores també poden ser importants, com per exemple que l'algoritme d'assignació de tasques tingui en compte la productivitat i rendiment del diferents clients a l'hora d'assignar les tasques a processar. També es podrien crear nous serveis web per poder oferir informació de l'estat d'un projecte en concret en un moment donat o un històric de les activitats que s'han dut a terme.

La part del projecte que s'ha implementat per provar el funcionament, en aquest cas l'aplicació per alienar seqüències d'ADN, també és àmpliament millorable. En principi, es podrien oferir més funcionalitats com alineament de seqüències d'ARN i proteïnes, més formats per acceptar l'entrada de seqüències, etc. També es podria replantejar la implementació de l'algoritme o inclús implementar algoritmes diferents.

5.Referències

[1] *Introduction to parallel computing*. https://computing.llnl.gov/tutorials/parallel_comp

[2] Alberto Ros, *Parallel and ditributed computing*, InTech, January 01, 2010

[3] M. Ben-Ari, *Principles of Concurrent and Distributed Programming*, 2nd Ed., Addison-Wesley, February 24, 2006

[4] Sriraman K R, Lead Architect, Research & Innovation, Altimetrik, *Grid computing on mobile devices, a point of view*. http://www.altimetrik.com/wp-content/uploads/2014/06/Grid-computing-on-mobile-devices.pdf

[5] Ketan B. Parmar, Nalinbhai N. Jani, Pranav S. Shrivastav, Mitesh H. Patel, *Mobile Grid Computing: Facts or Fantasy?* International Journal of Multidisciplinary Sciences and Engineering, vol. 4, no . 1, january 2013

[6] Rodrigo Santa Maria, *Alineamientos Múltiples de secuencias*. http://vis.usal.es/rodrigo/documentos/bioinfo/temas/5_Alineamientos%20m%C3%BAltiples.pdf

[7] Wikipedia, *Multiple Sequence Allignment*. http://en.wikipedia.org/wiki/Multiple_sequence_alignment

[8] Wikipedia, *Needleman-Wusch Algorithm*. http://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_algorithm

[9] Eric C. Rouchka, *Dynamic Programing*. http://www.avatar.se/molbioinfo2001/dynprog/dynamic.html

[10] Wikipedia, *Neighbor Joining*. http://en.wikipedia.org/wiki/Neighbor_joining

[11] Craig Larman, *UML y Patrones*, 2ª Ed. Pearson Educación S.A., Madrid 2004

[12] Wikipedia, *Facade Pattern*. http://en.wikipedia.org/wiki/Facade_pattern

[13] Wikipedia, *State Pattern*. http://en.wikipedia.org/wiki/State_pattern

[14] Oracle Corporation, *Jersey Restful Webservices in Java*. https://jersey.java.net/

[15] Red Hat Project, *Hibernate ORM*. http://hibernate.org/orm/

[16] Wikipedia, Dalvik. http://es.wikipedia.org/wiki/Dalvik

[17] Android Developer Tutorials, *DexClassLoader*.

http://developer.android.com/reference/dalvik/system/DexClassLoader.html

[18] Dr. Herong Yang, *"dx.bat –dex" Command. Converting .class files into .dex files.*

http://www.herongyang.com/Android/Project-dx-Command-Converting-class-Files-into-dex-File.html

[19] TutorialsPoint, *Android-Services.*

http://www.tutorialspoint.com/android/android_services.htm

[20] Vogella, *Android Services Tutorial*.

http://www.vogella.com/tutorials/AndroidServices/article.html

[21] Think Android, *Handling Screen O and, Screen OFF intents*.

https://thinkandroid.wordpress.com/2010/01/24/handling-screen-off-and-screen-on-intents/

[22] Google Project Hosting, *Google-Gson*. https://code.google.com/p/google-gson/

[23] TutorialsPoint, Java Multithreading.

http://www.tutorialspoint.com/java/java_multithreading.htm

[24] Oracle, *Java software*. https://www.oracle.com/java/index.html

[25] Oracle, *MySql*. https://www.mysql.com/

Annex 1. Estudi econòmic

En aquest apartat es detallen els costos que s'han produït a l'hora de portar a terme aquest projecte. No s'ha incorporat cap apartat de preus de venda o beneficis, ja que el projecte s'ha creat amb una finalitat sense cap ànim de lucre i amb la intenció d'oferir noves eines que aprofitin els recursos existents. Tot i això, la implementació d'aquest prototip a comportat uns costos que són els que aquí es detallen.

A1.1. Costos de material i infraestructures

Costos de materials i infraestructures			
Descripció	**Unitats**	**Preu unitari**	**Total**
Hardware per desenvolupament de codi	1	1000,00	1000,00
Servidor per el desplegament de l'aplicació servidora*	1	1200,00	1200,00
Servidor per el desplegament de l'aplicació proveïdora*	1	1200,00	1200,00
Dispositius mòbils amb sistema operatiu Android per desplegament de l'aplicació client	2	450,00	900,00
Total Costos de materials i infraestructures			4.300,00 €

Taula A1.1.1. Costos de materials i infraestructures

(*) El preu unitari es basa en una contractació anual (12 mesos)

A1.2. Costos de recursos humans

Totes les funcions detallades en la taula de costos de recursos humans, han estat assignades a un perfil professional d'Enginyer Tècnic Informàtic extern. Les unitats corresponen a hores de treball i el preu unitari correspon al preu/hora. El preu/hora s'ha extret de la informació estadística oferta per l'Institut Nacional d'Estadistica (INE, http://www.ine.es/prensa/np741.pdf)

Costos de recursos humans			
Descripció	**Unitats**	**Preu unitari**	**Total**
Recerca	112	24,00	2688,00
Documentació	150	24,00	3600,00
Anàlisi	80	24,00	1920,00
Disseny	90	24,00	2160,00
Programació i proves	800	24,00	19200,00
Total Costos de recursos humans			29.568,00 €

Taula A1.1.2. Costos de recursos humans

A1.3. Cost total

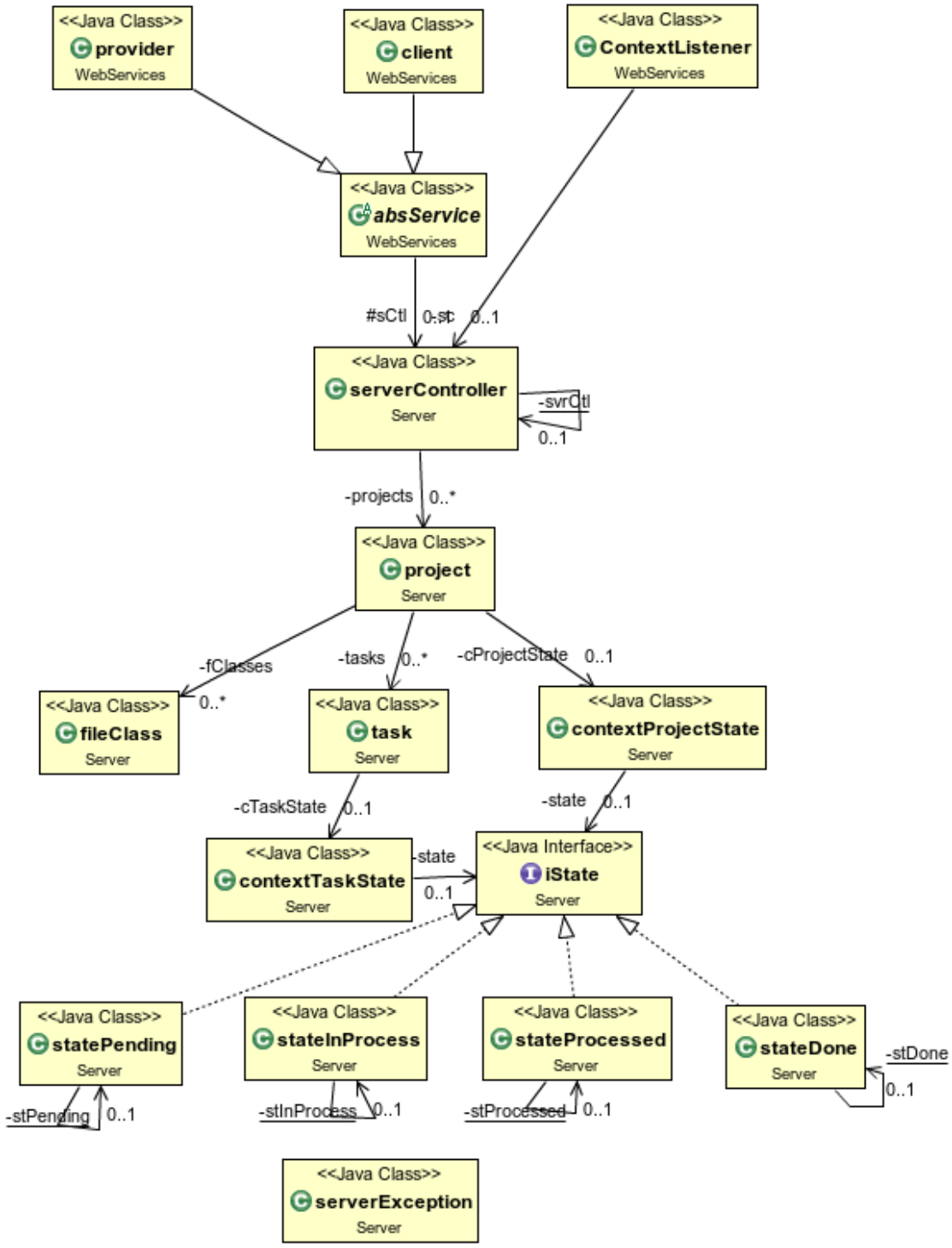
Cost Total	
Descripció	**Total**
Material i infraestrucutres	4300,00
Recursos humans	29568,00
Subtotal	33868,00
Despeses indirectes (19%)*	6434,92
Total	40.302,92 €

Taula A1.1.3. Cost total estimat

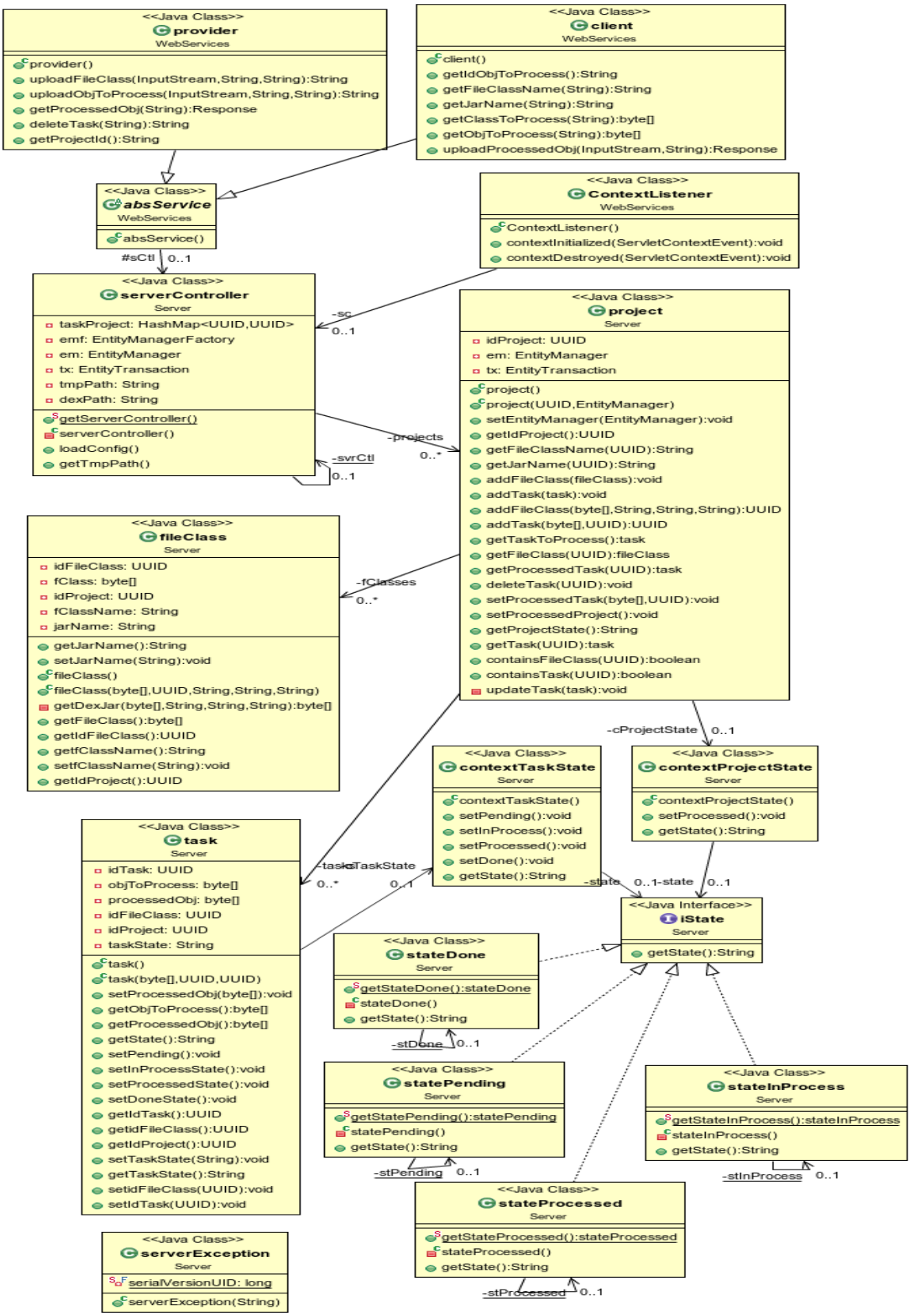
(*) El grup de despeses indirectes inclou despeses com: electricitat, lloguer d'oficina, etc. El conjunt d'aquestes despeses s'ha estimat que incrementen en un 19% el cost total del projecte.

Annex 2. Documentació tècnica de l'aplicació amb funcions de servidor

A2.1. Diagrama de domini



A2.2. Diagrama de classes



A2.3. Casos d'us

A2.3.1. Rebre fitxer de classe

<table>
<tr><td>Nom cas d'ús</td><td colspan="2">P1. Rebre fitxer de classe</td></tr>
<tr><td>Descripció</td><td colspan="2">L'aplicació rep un fitxer .class de l'aplicació proveïdora</td></tr>
<tr><td>Actor principal</td><td colspan="2">Proveïdor</td></tr>
<tr><td>Pre-condició</td><td colspan="2">El proveïdor té un identificador de projecte vàlid.</td></tr>
<tr><td>Flux bàsic</td><td colspan="2"><table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1</td><td>El proveïdor fa arribar el fitxer .class, el nom de la classe i el projecte al que pertany al servidor.</td></tr>
<tr><td>2</td><td>El servidor converteix el fitxer .class en un fitxer .dex d'Android i l'empaqueta en un fitxer .jar</td></tr>
<tr><td>3</td><td>El servidor guarda a la base de dades l'objecte creat amb els seu identificador</td></tr>
<tr><td>4</td><td>El servidor posa l'objecte creat a disposició dels clients.</td></tr>
</table></td></tr>
<tr><td>Post-condició</td><td colspan="2">El servidor retorna l'identificador del objecte creat</td></tr>
<tr><td>Flux alternatiu</td><td colspan="2"><table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1, 2, 3, 4</td><td>El servidor retorna una excepció indicant el detall concret de l'error perquè l'aplicació proveïdora actuï en conseqüència</td></tr>
</table></td></tr>
</table>

A2.3.2. Rebre objecte a processar

<table>
<tr><td>Nom cas d’ús</td><td colspan="2">P2. Rebre objecte a processar</td></tr>
<tr><td>Descripció</td><td colspan="2">L’aplicació rep un objecte de l’aplicació proveïdora per que el processin les aplicacions clients</td></tr>
<tr><td>Actor principal</td><td colspan="2">Proveïdor</td></tr>
<tr><td>Pre-condició</td><td colspan="2">El proveïdor té un identificador de projecte vàlid i coneix l’identificador del objecte fileClass al que pertany.</td></tr>
<tr><td>Flux bàsic</td><td colspan="2"><table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1</td><td>El proveïdor fa arribar l’objecte, l’identificador de la classe i el projecte al que pertany al servidor.</td></tr>
<tr><td>2</td><td>El servidor crea un nou objecte que conté l’objecte rebut i les seves relacions (projecte i classe) , l’assigna un identificador i el posa en estat “pendent de processar”</td></tr>
<tr><td>3</td><td>El servidor guarda a la base de dades l’objecte creat amb els seu identificador</td></tr>
<tr><td>4</td><td>El servidor posa l’objecte creat a disposició dels clients.</td></tr>
</table></td></tr>
<tr><td>Post-condició</td><td colspan="2">El servidor retorna l’identificador del objecte creat</td></tr>
<tr><td>Flux alternatiu</td><td colspan="2"><table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1, 2, 3, 4</td><td>El servidor retorna una excepció indicant el detall concret de l’error perquè l’aplicació proveïdora actuï en conseqüència</td></tr>
</table></td></tr>
</table>

A2.3.3. Enviar objecte processat

Nom cas d'ús	**P3. Enviar objecte processat**
Descripció	El servidor envia un objecte processat cap el proveïdor
Actor principal	Proveïdor
Pre-condició	El proveïdor coneix l'identificador del objecte processat.
Flux bàsic	**Pas** / **Acció** 1 — El proveïdor fa una petició al servidor per obtenir l'objecte processat fent servir l'identificador de l'objecte. 2 — El servidor comprova si l'objecte demanat pertany a un projecte vàlid 3 — El servidor comprova que l'objecte demanat estigui en estat processat
Post-condició	El servidor retorna l'objecte demanat
Flux alternatiu	**Pas** / **Acció** 1, 2, 3 — El servidor retorna una excepció indicant el detall concret de l'error perquè l'aplicació proveïdora actuï en conseqüència

A2.3.4. Eliminació d'un objecte processat

Nom cas d'ús	**P4. Eliminació d'un objecte processat**
Descripció	L'aplicació rep una petició d'eliminació d'un objecte processat
Actor principal	Proveïdor
Pre-condició	El proveïdor coneix l'identificador del objecte processat.
Flux bàsic	**Pas** / **Acció** 1 — El proveïdor fa una petició d'eliminació d'un objecte processat fent servir l'identificador de l'objecte 2 — El servidor comprova si l'objecte demanat pertany a un projecte vàlid 3 — El servidor posa l'objecte en estat finalitzat.
Post-condició	El servidor retorna un codi per indicar que totes les accions s'han fet correctament
Flux alternatiu	**Pas** / **Acció** 1, 2, 3 — El servidor retorna una excepció indicant el detall concret de l'error perquè l'aplicació proveïdora actuï en conseqüència

A2.3.5. Enviar identificador d'objecte a processar

<table>
<tr><td>Nom cas d'ús</td><td colspan="2">C1. Enviar identificador d'objecte a processar</td></tr>
<tr><td>Descripció</td><td colspan="2">Obtenir un identificador d'objecte per processar</td></tr>
<tr><td>Actor principal</td><td colspan="2">Client</td></tr>
<tr><td>Pre-condició</td><td colspan="2">El client té disposició de processar un objecte</td></tr>
<tr><td>Flux bàsic</td><td colspan="2"><table><tr><th>Pas</th><th>Acció</th></tr><tr><td>1</td><td>El client fa una petició per obtenir un identificador d'un objecte per processar-ho</td></tr><tr><td>2</td><td>El servidor busca objectes en estat "pendent" dins dels projectes actius</td></tr><tr><td>3</td><td>El servidor canvia l'estat de l'objecte trobat a "en procés"</td></tr></table></td></tr>
<tr><td>Post-condició</td><td colspan="2">El servidor retorna l'identificador de l'objecte a processar</td></tr>
<tr><td>Flux alternatiu</td><td colspan="2"><table><tr><th>Pas</th><th>Acció</th></tr><tr><td>1, 3</td><td>El servidor retorna una excepció indicant el detall concret de l'error perquè l'aplicació client actuï en conseqüència</td></tr><tr><td>2</td><td>El servidor busca objectes en estat "en procés" dins dels projectes actius.</td></tr></table></td></tr>
</table>

A2.3.6. Enviar nom de classe

<table>
<tr><td>Nom cas d'ús</td><td colspan="2">C2. Enviar nom de classe</td></tr>
<tr><td>Descripció</td><td colspan="2">Enviar el nom de la classe a la que pertany un objecte en concret</td></tr>
<tr><td>Actor principal</td><td colspan="2">Client</td></tr>
<tr><td>Pre-condició</td><td colspan="2">El client coneix l'identificador del objecte del qual vol obtenir el nom de la classe a la que pertany.</td></tr>
<tr><td>Flux bàsic</td><td colspan="2"><table><tr><th>Pas</th><th>Acció</th></tr><tr><td>1</td><td>El servidor rep una petició de per obtenir el nom de la classe d'un objecte fent servir l'identificador de l'objecte</td></tr><tr><td>2</td><td>El servidor comprova que l'objecte pertany a un projecte vàlid</td></tr><tr><td>3</td><td>El servidor busca en el projecte el nom de classe al que pertany l'objecte</td></tr></table></td></tr>
<tr><td>Post-condició</td><td colspan="2">El servidor retorna el nom de la classe de l'objecte</td></tr>
<tr><td>Flux alternatiu</td><td colspan="2"><table><tr><th>Pas</th><th>Acció</th></tr><tr><td>1, 2, 3</td><td>El servidor retorna una excepció indicant el detall concret de l'error perquè l'aplicació client actuï en conseqüència</td></tr></table></td></tr>
</table>

A2.3.7. Enviar nom de fitxer .jar

<table>
<tr><td>Nom cas d'ús</td><td>C3. Enviar nom de fitxer .jar</td></tr>
<tr><td>Descripció</td><td>Enviar el nom del fitxer .jar en el que està empaquetada una classe en concret</td></tr>
<tr><td>Actor principal</td><td>Client</td></tr>
<tr><td>Pre-condició</td><td>El client coneix l'identificador del objecte del qual vol obtenir el nom del fitxer .jar on està empaqueta la classe a la que pertany.</td></tr>
<tr><td>Flux bàsic</td><td>
<table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1</td><td>El servidor rep una petició de per obtenir el nom del fitxer .jar fent servir un identificador d'objecte</td></tr>
<tr><td>2</td><td>El servidor comprova que l'objecte pertany a un projecte vàlid</td></tr>
<tr><td>3</td><td>El servidor busca en el projecte el nom del fitxer .jar de la classe al que pertany l'objecte</td></tr>
</table>
</td></tr>
<tr><td>Post-condició</td><td>El servidor retorna el nom del fitxer .jar</td></tr>
<tr><td>Flux alternatiu</td><td>
<table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1, 2, 3</td><td>El servidor retorna una excepció indicant el detall concret de l'error perquè l'aplicació client actuï en conseqüència</td></tr>
</table>
</td></tr>
</table>

A2.3.8. Enviar fitxer .jar

<table>
<tr><td>Nom cas d'ús</td><td>C4. Enviar fitxer .jar</td></tr>
<tr><td>Descripció</td><td>Enviar el fitxer .jar amb la classe de l'objecte a processar</td></tr>
<tr><td>Actor principal</td><td>Client</td></tr>
<tr><td>Pre-condició</td><td>El client coneix l'identificador del objecte del qual vol obtenir el fitxer .jar</td></tr>
<tr><td>Flux bàsic</td><td>
<table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1</td><td>El servidor rep una petició de per obtenir el fitxer .jar d'un objecte en concret fent servir l'identificador de l'objecte</td></tr>
<tr><td>2</td><td>El servidor comprova que l'objecte pertany a un projecte vàlid</td></tr>
<tr><td>3</td><td>El servidor busca en el projecte el fitxer .jar de l'objecte demanat</td></tr>
</table>
</td></tr>
<tr><td>Post-condició</td><td>El servidor retorna el fitxer .jar</td></tr>
<tr><td>Flux alternatiu</td><td>
<table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1, 2, 3</td><td>El servidor retorna una excepció indicant el detall concret de l'error perquè l'aplicació client actuï en conseqüència</td></tr>
</table>
</td></tr>
</table>

A2.3.9. Enviar objecte a processar

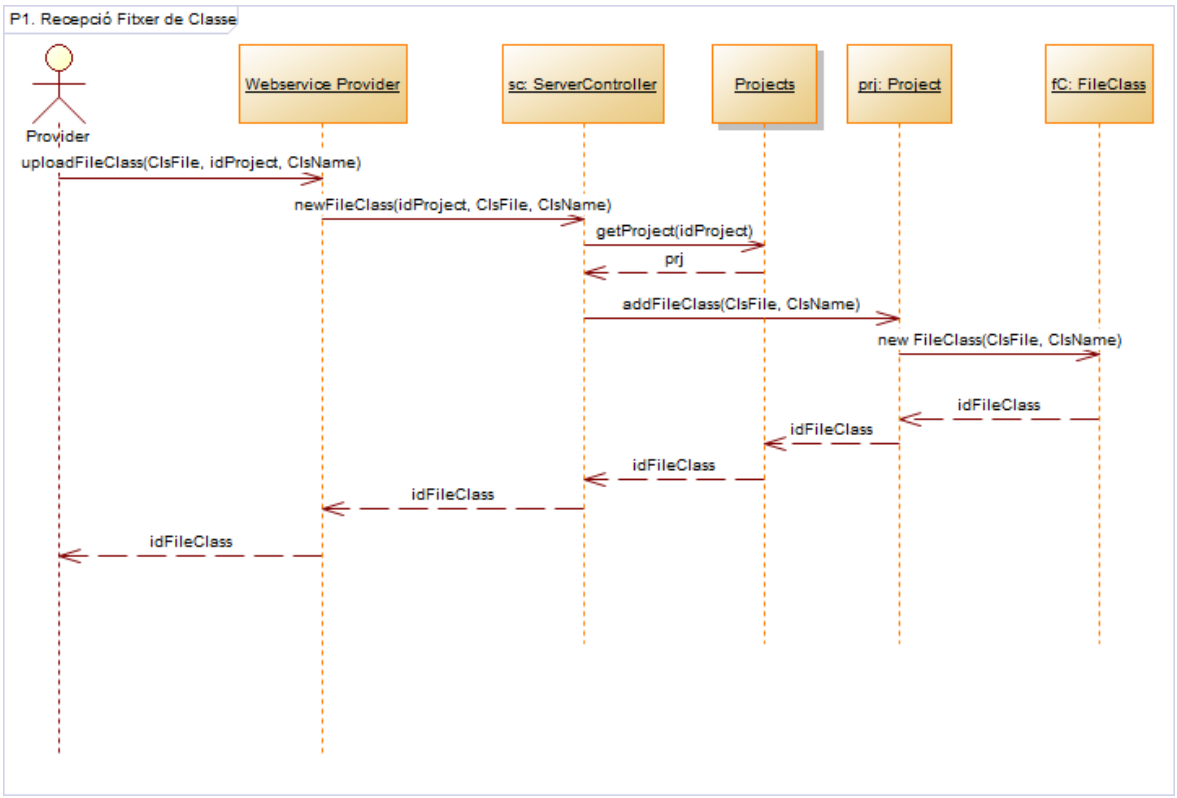
<table>
<tr><td>Nom cas d'ús</td><td>C5. Enviar objecte a processar</td></tr>
<tr><td>Descripció</td><td>Enviar l'objecte per que sigui processat pel client.</td></tr>
<tr><td>Actor principal</td><td>Client</td></tr>
<tr><td>Pre-condició</td><td>El client coneix l'identificador del objecte que vol processar.</td></tr>
<tr><td>Flux bàsic</td><td>
<table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1</td><td>El servidor rep una petició de per obtenir un objecte concret fent servir el seu identificador.</td></tr>
<tr><td>2</td><td>El servidor comprova que l'objecte pertany a un projecte vàlid.</td></tr>
<tr><td>3</td><td>El servidor comprova que l'objecte està en estat "pendent" o "en procés"</td></tr>
</table>
</td></tr>
<tr><td>Post-condició</td><td>El servidor retorna l'objecte demanat.</td></tr>
<tr><td>Flux alternatiu</td><td>
<table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1, 2, 3</td><td>El servidor retorna una excepció indicant el detall concret de l'error perquè l'aplicació client actuï en conseqüència</td></tr>
</table>
</td></tr>
</table>

A2.3.10. Rebre objecte processat

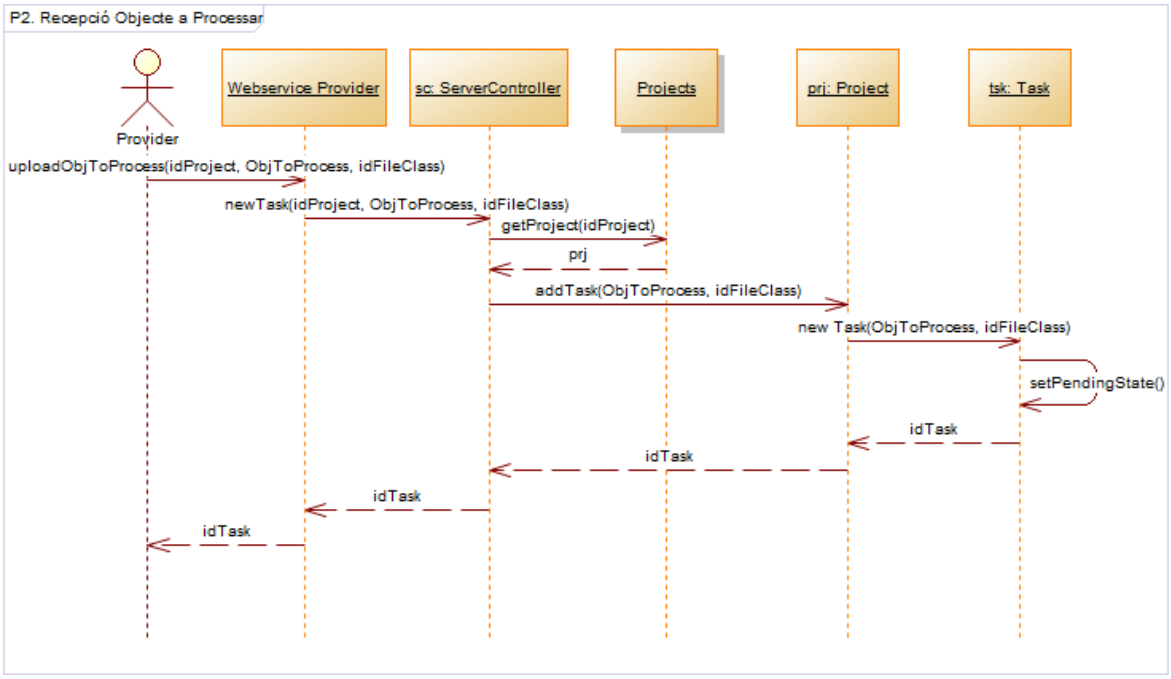
<table>
<tr><td>Nom cas d'ús</td><td colspan="2">C6. Rebre objecte processat</td></tr>
<tr><td>Descripció</td><td colspan="2">Rebre un objecte prèviament processat</td></tr>
<tr><td>Actor principal</td><td colspan="2">Client</td></tr>
<tr><td>Pre-condició</td><td colspan="2">El client coneix l'identificador del objecte.</td></tr>
<tr><td>Flux bàsic</td><td colspan="2"><table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1</td><td>El servidor rep l'objecte processat i l'identificador de l'objecte al que pertany</td></tr>
<tr><td>2</td><td>El servidor comprova que l'objecte pertany a un projecte vàlid</td></tr>
<tr><td>3</td><td>El servidor comprova que l'objecte està en estat "pendent" o "en procés"</td></tr>
<tr><td>4</td><td>El servidor incorpora l'objecte processat a l'objecte del projecte.</td></tr>
<tr><td>5</td><td>El servidor marca posa l'objecte del projecte en estat "processat"</td></tr>
</table></td></tr>
<tr><td>Post-condició</td><td colspan="2">El servidor retorna un codi indicant que totes les accions s'han completat correctament.</td></tr>
<tr><td>Flux alternatiu</td><td colspan="2"><table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1, 2, 3, 5</td><td>El servidor retorna una excepció indicant el detall concret de l'error perquè l'aplicació client actuï en conseqüència</td></tr>
</table></td></tr>
</table>

A2.4. Diagrames de seqüència

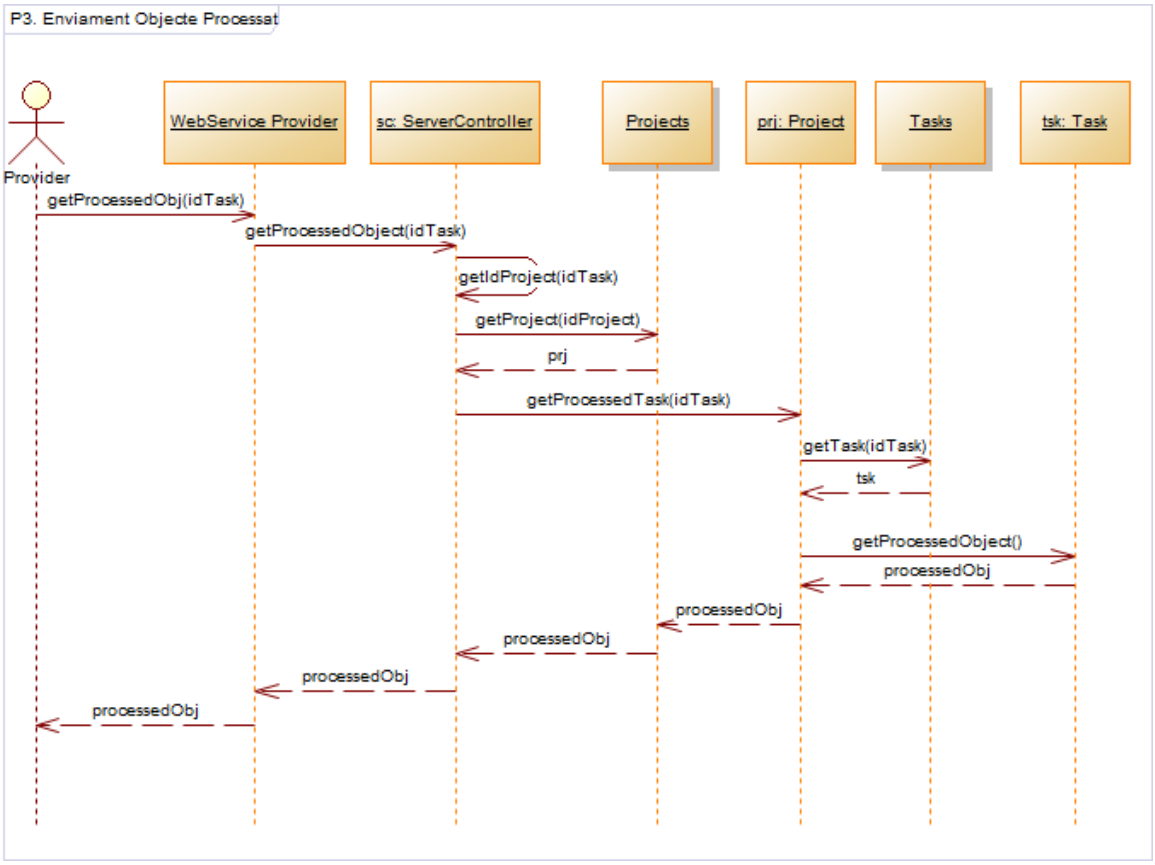
A2.4.1. Rebre fitxer de classe



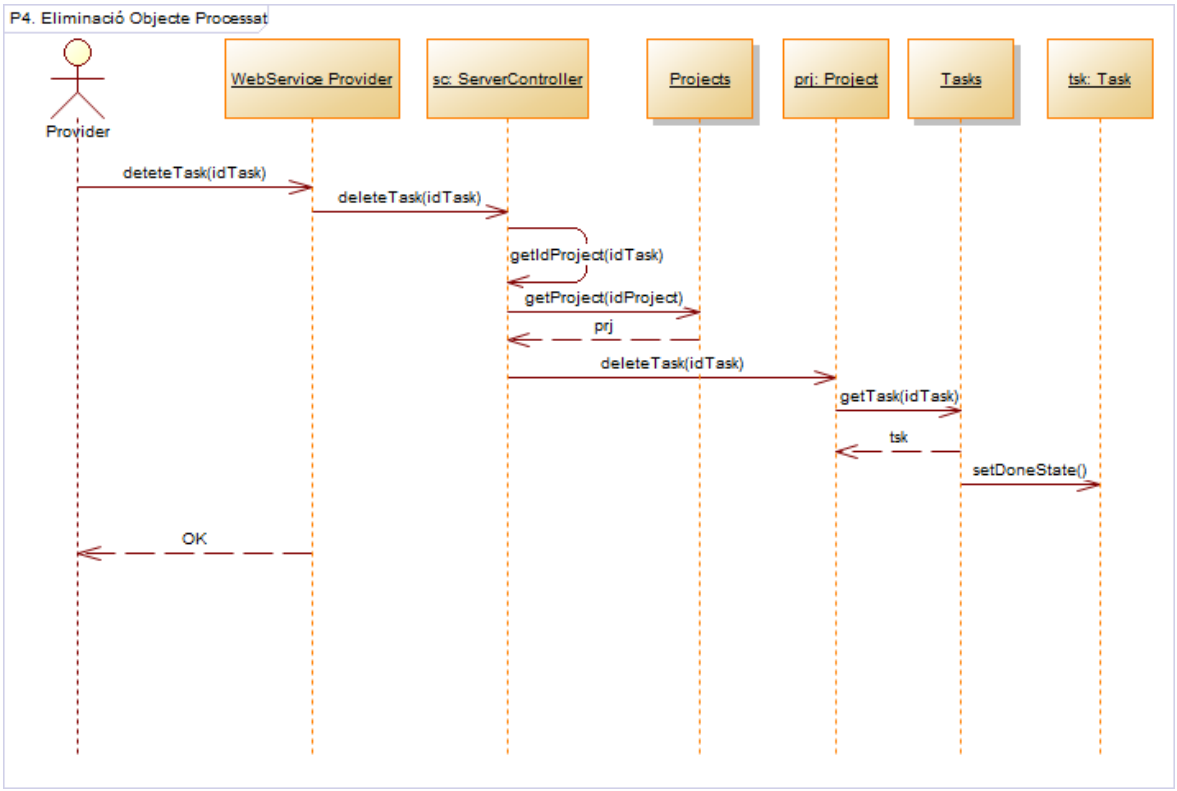
A2.4.2. Rebre objecte a processar



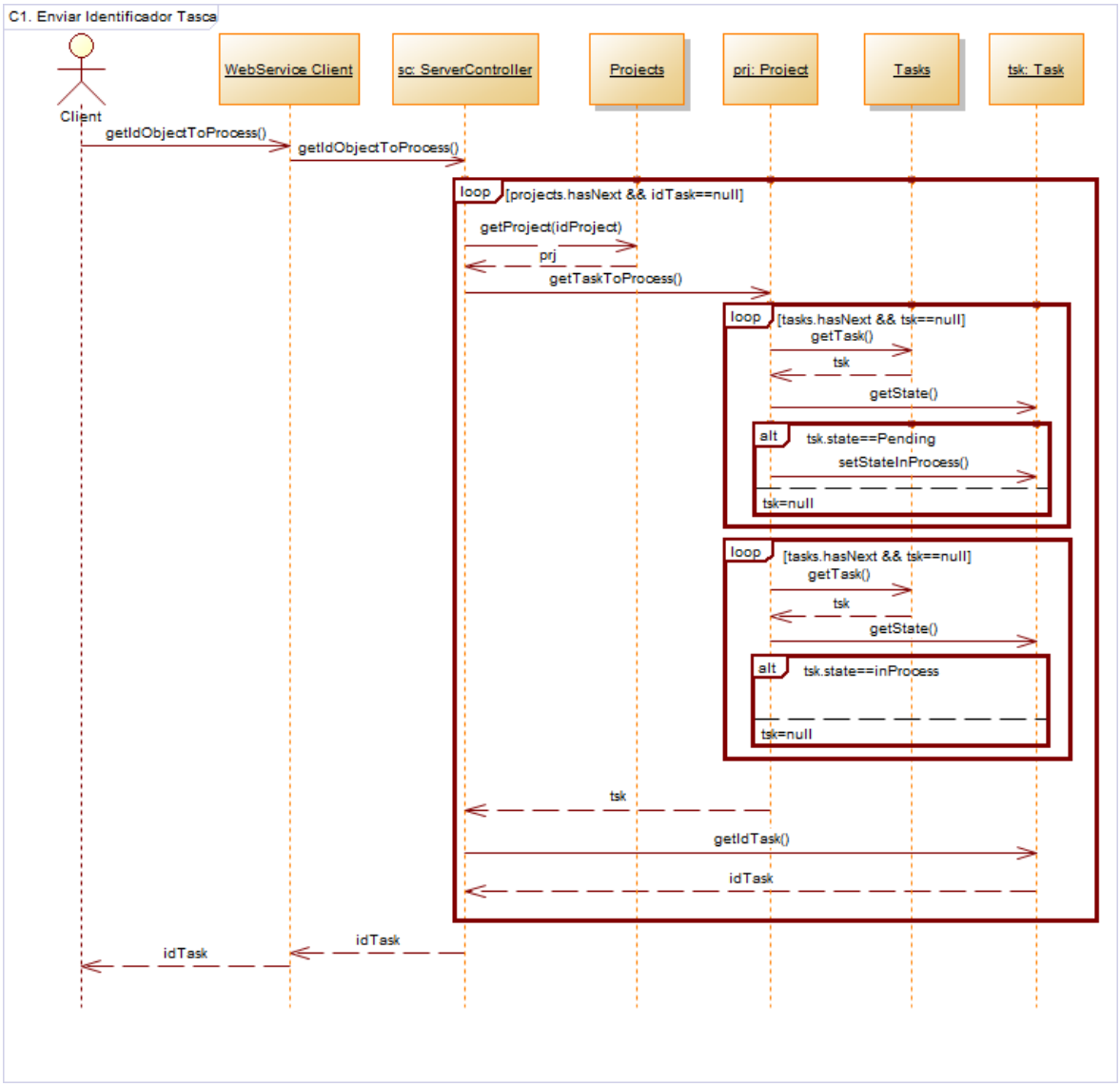
A2.4.3. Enviar objecte processat



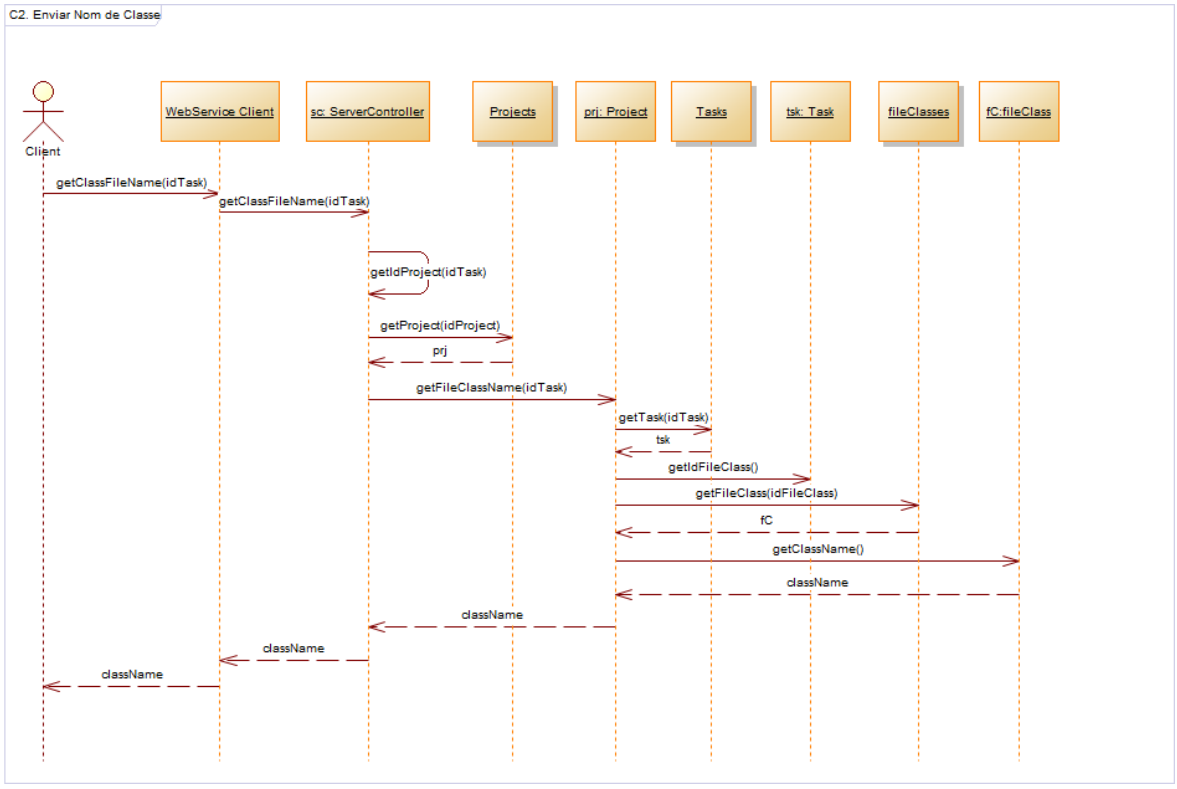
A2.4.4. Eliminació d'un objecte processat



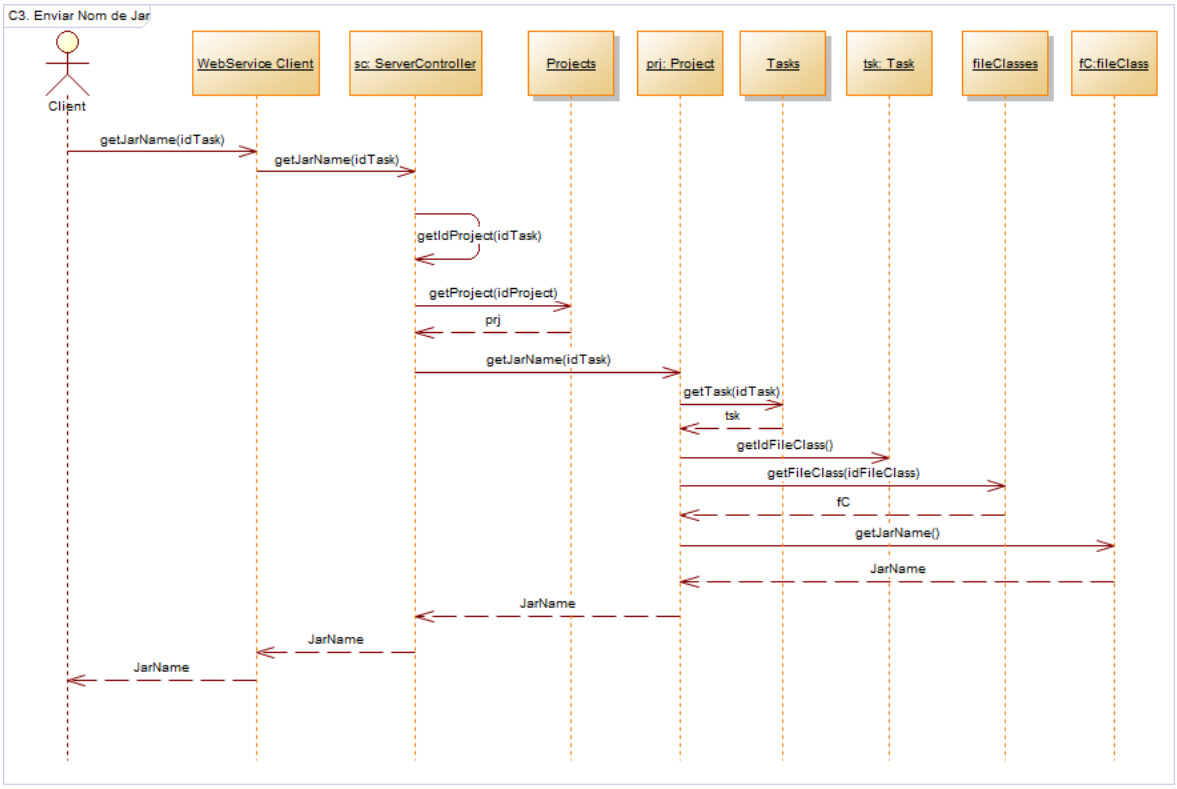
A2.4.5. Enviar identificador d'objecte a processar



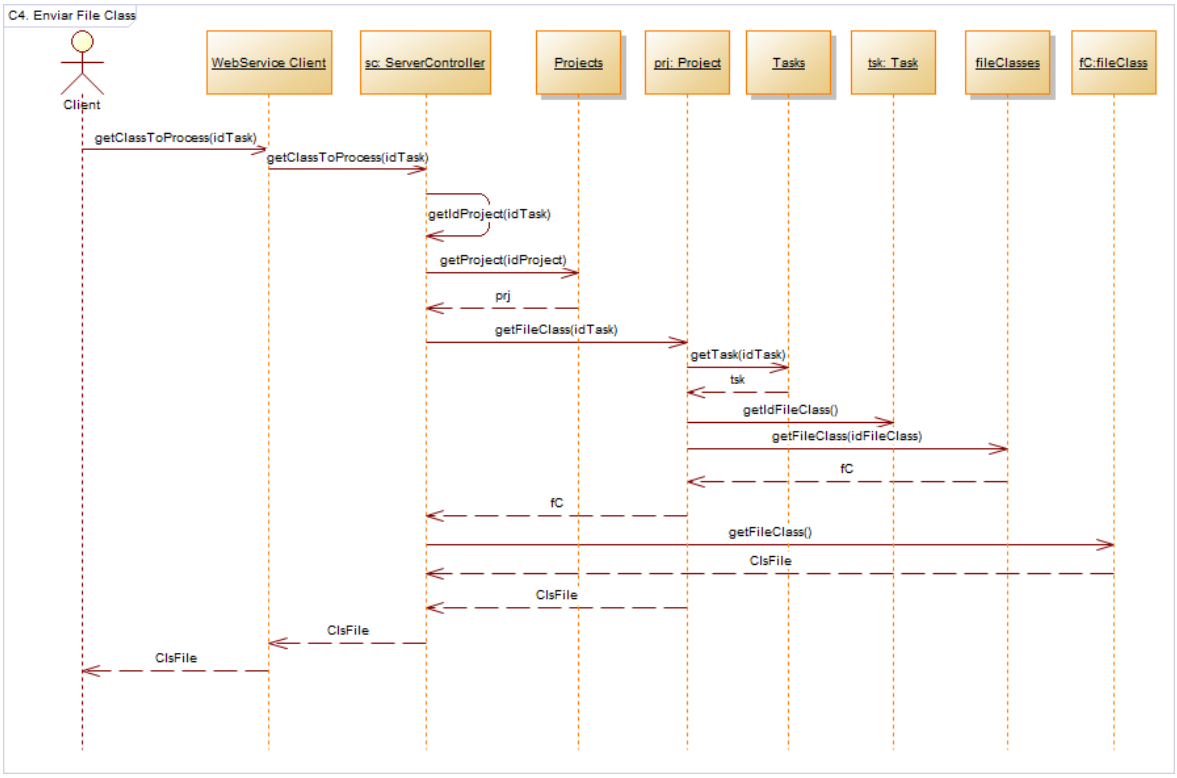
A2.4.6. Enviar nom de classe



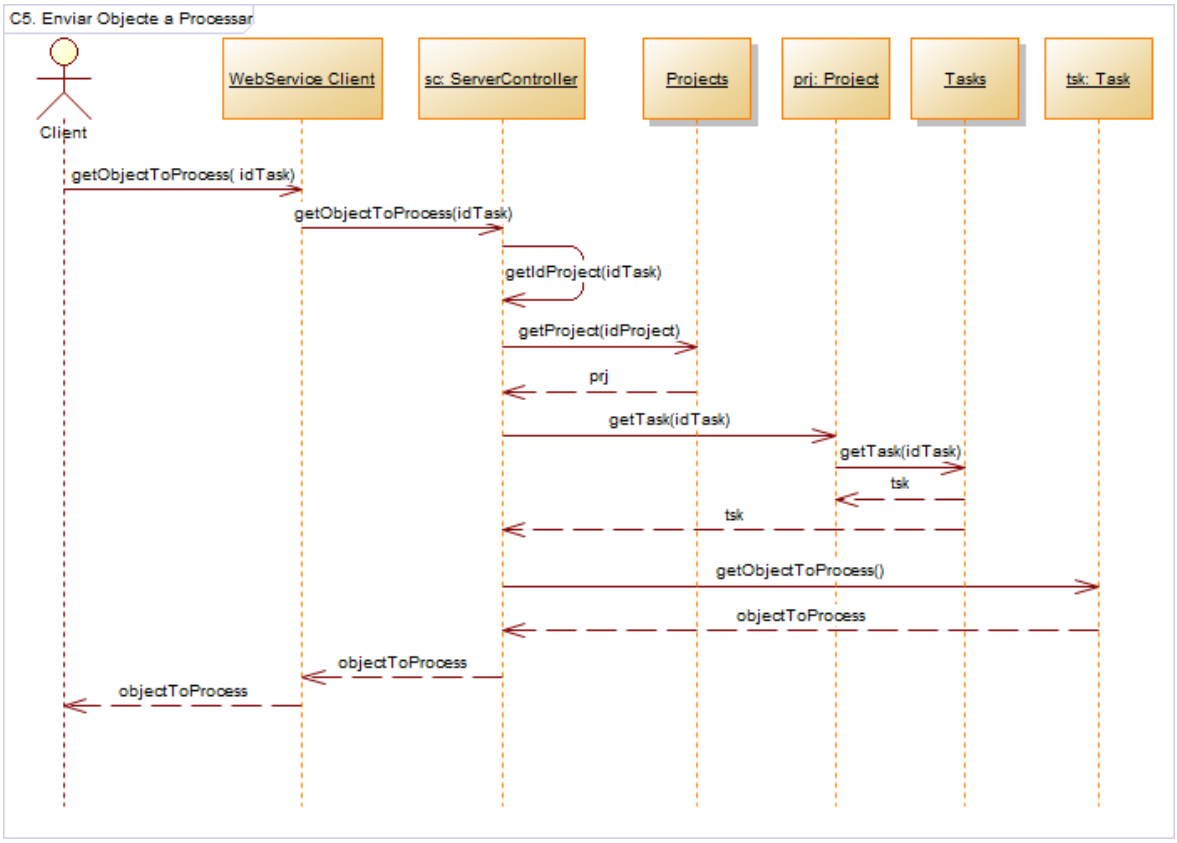
A2.4.7. Enviar nom de fitxer .jar



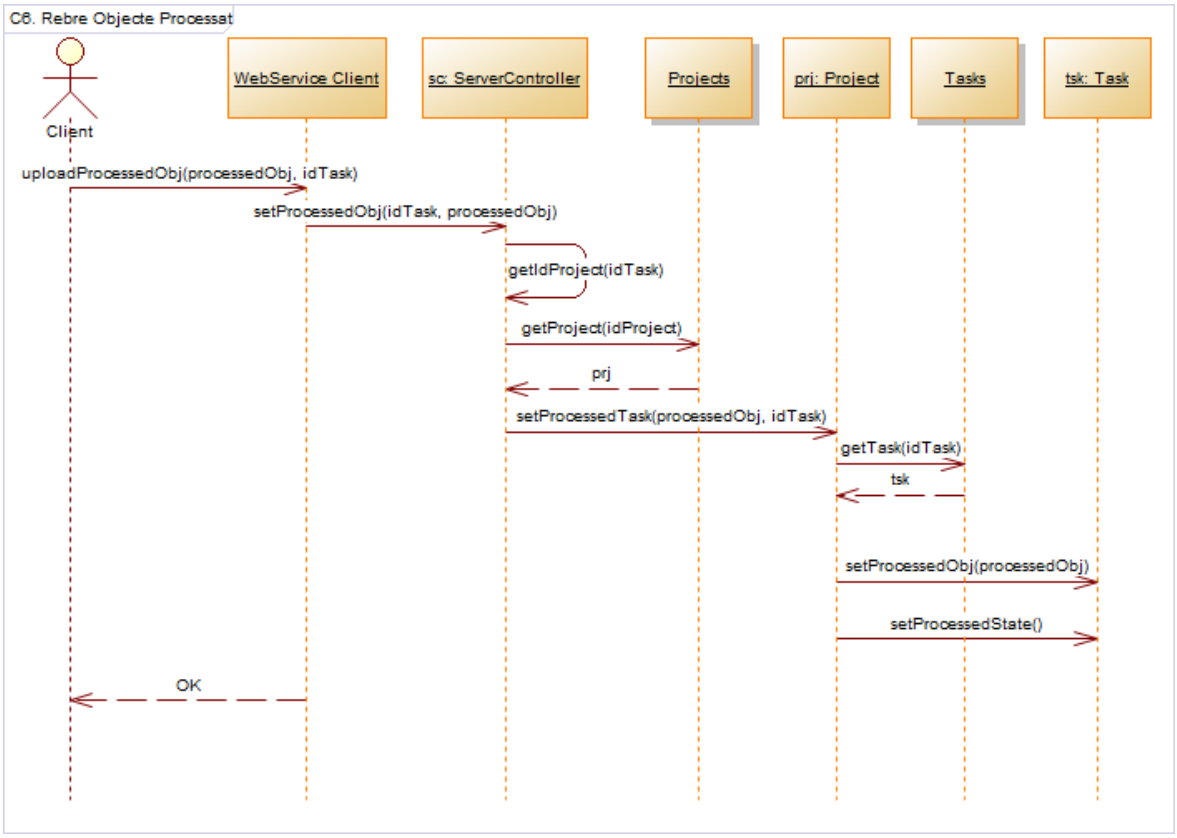
A2.4.8. Enviar fitxer .jar



A2.4.9. Enviar objecte a processar

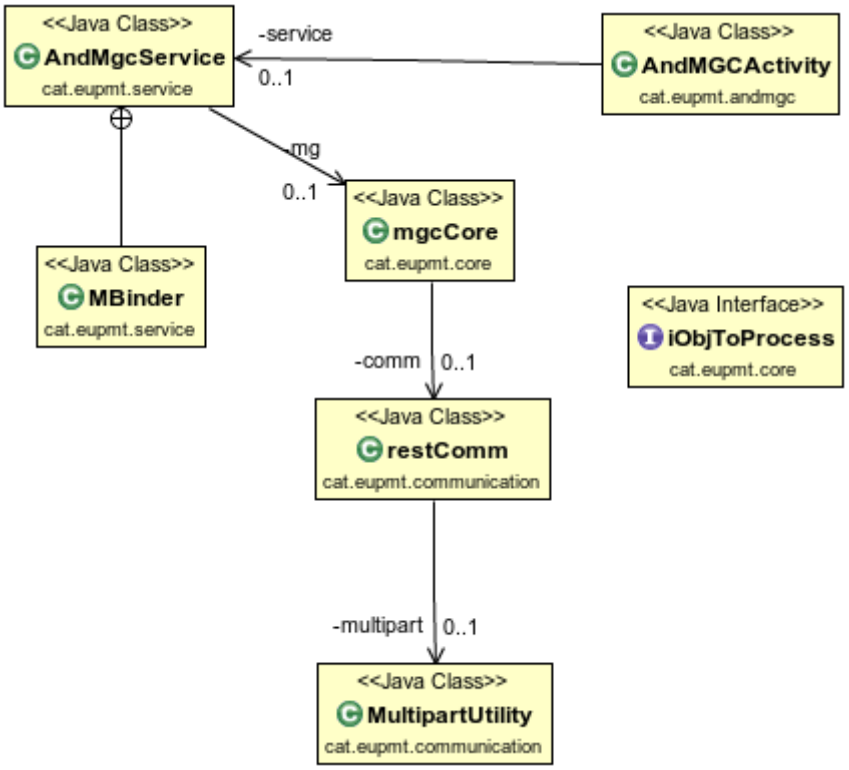


A2.4.10. Rebre objecte processat

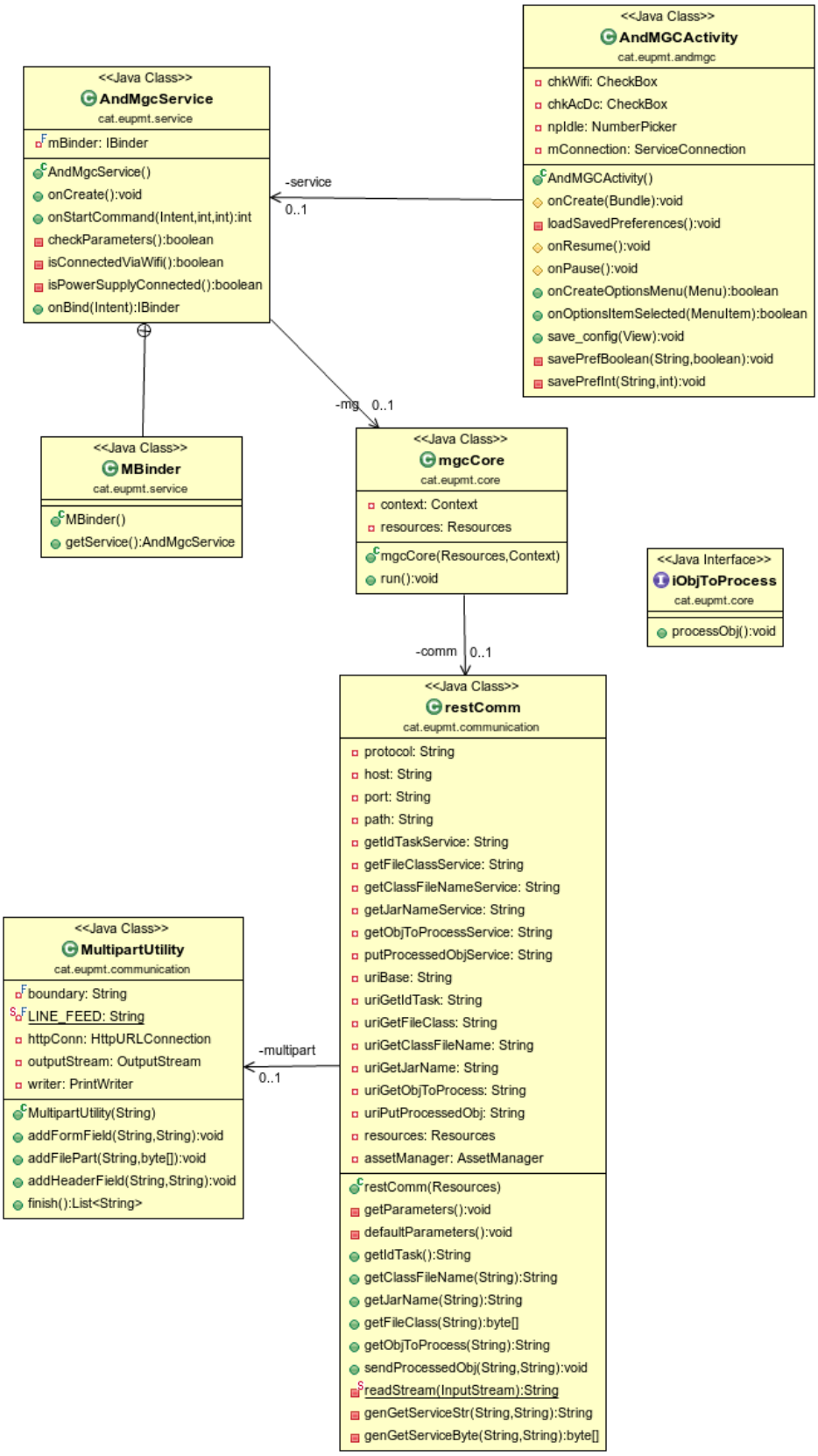


Annex 3. Documentació tècnica de l'aplicació amb funcions de client

A3.1. Diagrama de domini



A3.2. Diagrama de classes



A3.3. Casos d'us

A3.3.1. Modificar paràmetres

<table>
<tr><td>Nom cas d'ús</td><td colspan="2">AC1. Modificar paràmetres</td></tr>
<tr><td>Descripció</td><td colspan="2">Modificar els valors dels paràmetres de l'aplicació</td></tr>
<tr><td>Actor principal</td><td colspan="2">Usuari</td></tr>
<tr><td>Pre-condició</td><td colspan="2">--</td></tr>
<tr><td>Flux bàsic</td><td colspan="2"><table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1</td><td>L'aplicació mostra la pantalla amb els diferents paràmetres i els seus respectius valors</td></tr>
<tr><td>2</td><td>L'usuari modifica els valors segons el seu criteri</td></tr>
<tr><td>3</td><td>L'usuari guarda els paràmetres modificats</td></tr>
</table></td></tr>
<tr><td>Post-condició</td><td colspan="2">Els nous valors són enviats al servei que gestiona l'execució del subsistema de processament</td></tr>
<tr><td>Flux alternatiu</td><td colspan="2"><table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1, 2, 3</td><td>Es mostra un missatge d'error informant a l'usuari de la incidència.</td></tr>
</table></td></tr>
</table>

A3.3.2. Executar subsistema de processament

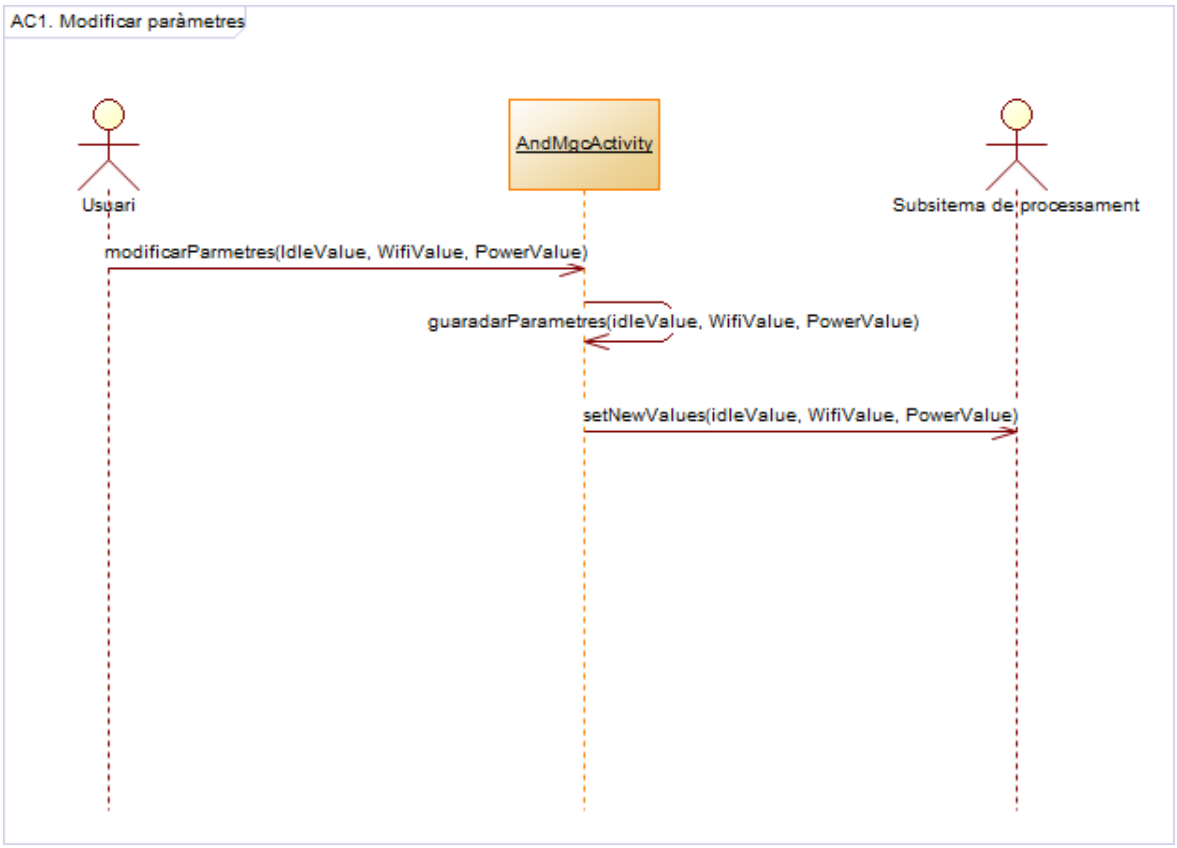
<table>
<tr><td>Nom cas d’ús</td><td colspan="2">AC2. Executar subsistema de processament</td></tr>
<tr><td>Descripció</td><td colspan="2">L’aplicació comprova si es donen les circumstancies i executa el subsistema de processament</td></tr>
<tr><td>Actor principal</td><td colspan="2">Servei de gestió del subsistema de processament</td></tr>
<tr><td>Pre-condició</td><td colspan="2">El servei detecta que el dispositiu ha entrat en estat d’inactivitat</td></tr>
<tr><td>Flux bàsic</td><td colspan="2"><table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1</td><td>Comprova si la connexió a la xarxa de dades via wifi es correspon amb el valor del paràmetre de configuració</td></tr>
<tr><td>2</td><td>Comprova si la connexió a la xarxa elèctrica es correspon amb el valor del paràmetre de configuració</td></tr>
<tr><td>3</td><td>Comprova si el temps d’inactivitat supera el temps informat al paràmetre que indica quan temps d’inactivitat és necessari per executar el subsistema de processament</td></tr>
</table></td></tr>
<tr><td>Post-condició</td><td colspan="2">Executar el subsistema de processament</td></tr>
<tr><td>Flux alternatiu</td><td colspan="2"><table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1, 2, 3</td><td>S’atura el procés fins que es tornin a donar les circumstancies especifiques.</td></tr>
</table></td></tr>
</table>

A3.3.3. Subsistema de Processament

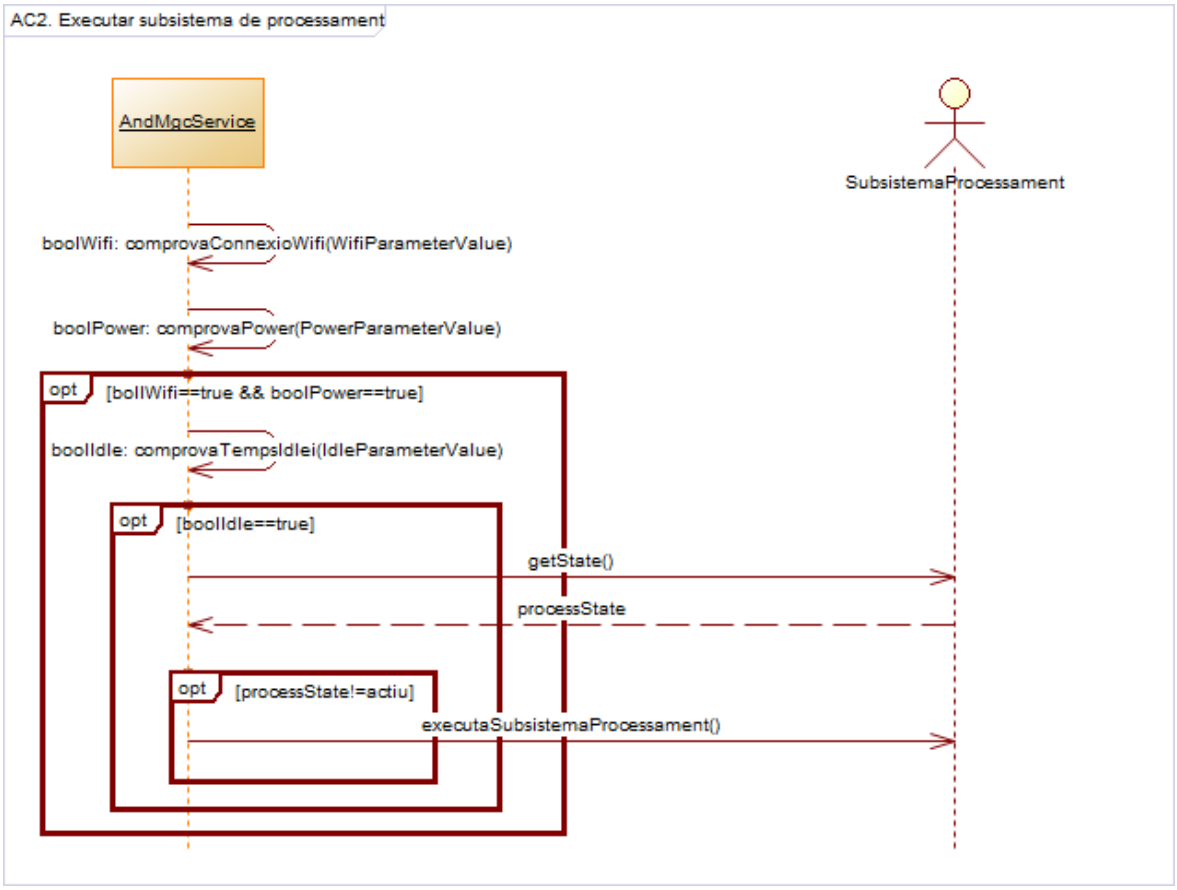
<table>
<tr><td>Nom cas d'ús</td><td>AC3. Subsistema de Processament</td></tr>
<tr><td>Descripció</td><td>Obté objectes dels servidor, els processa i els retorna</td></tr>
<tr><td>Actor principal</td><td>Servei de gestió del subsistema de processament</td></tr>
<tr><td>Pre-condició</td><td>Es compleixen els requisits indicats en els paràmetres de control.</td></tr>
<tr><td>Flux bàsic</td><td>
<table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1</td><td>S'obté un identificador de tasca a processar</td></tr>
<tr><td>2</td><td>S'obté el nom de la classe del objecte a processar</td></tr>
<tr><td>3</td><td>S'obté el nom de l'arxiu .jar que conté la classe del objecte a processar</td></tr>
<tr><td>4</td><td>S'obté l'arxiu .jar que conté la classe del objecte a processar</td></tr>
<tr><td>5</td><td>Es carrega la classe del objecte a processar</td></tr>
<tr><td>6</td><td>S'obté l'objecte a processar</td></tr>
<tr><td>7</td><td>Es carrega l'objecte a processar</td></tr>
<tr><td>8</td><td>Es processa l'objecte carregat</td></tr>
</table>
</td></tr>
<tr><td>Post-condició</td><td>S'envien els resultats a l'usuari</td></tr>
<tr><td>Flux alternatiu</td><td>
<table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1, 2, 3, 4, 5, 6, 7, 8</td><td>En qualsevol moment que deixi de complir-se la pre-condició o en qualsevol excepció durant el procés, aquest s'aturarà</td></tr>
</table>
</td></tr>
</table>

A3.4. Diagrames de seqüència

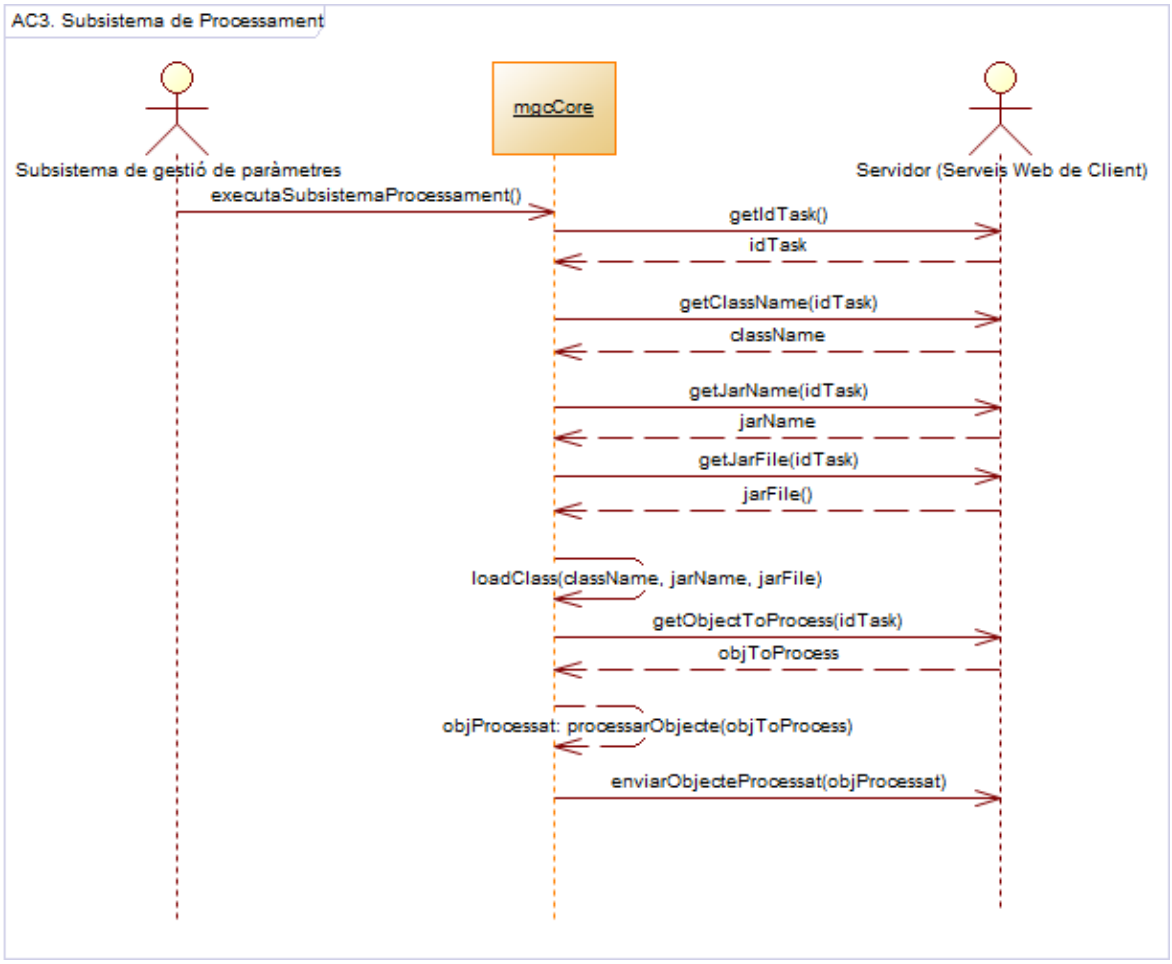
A3.4.1. Modificar paràmetres



A3.4.2. Executar subsistema de processament

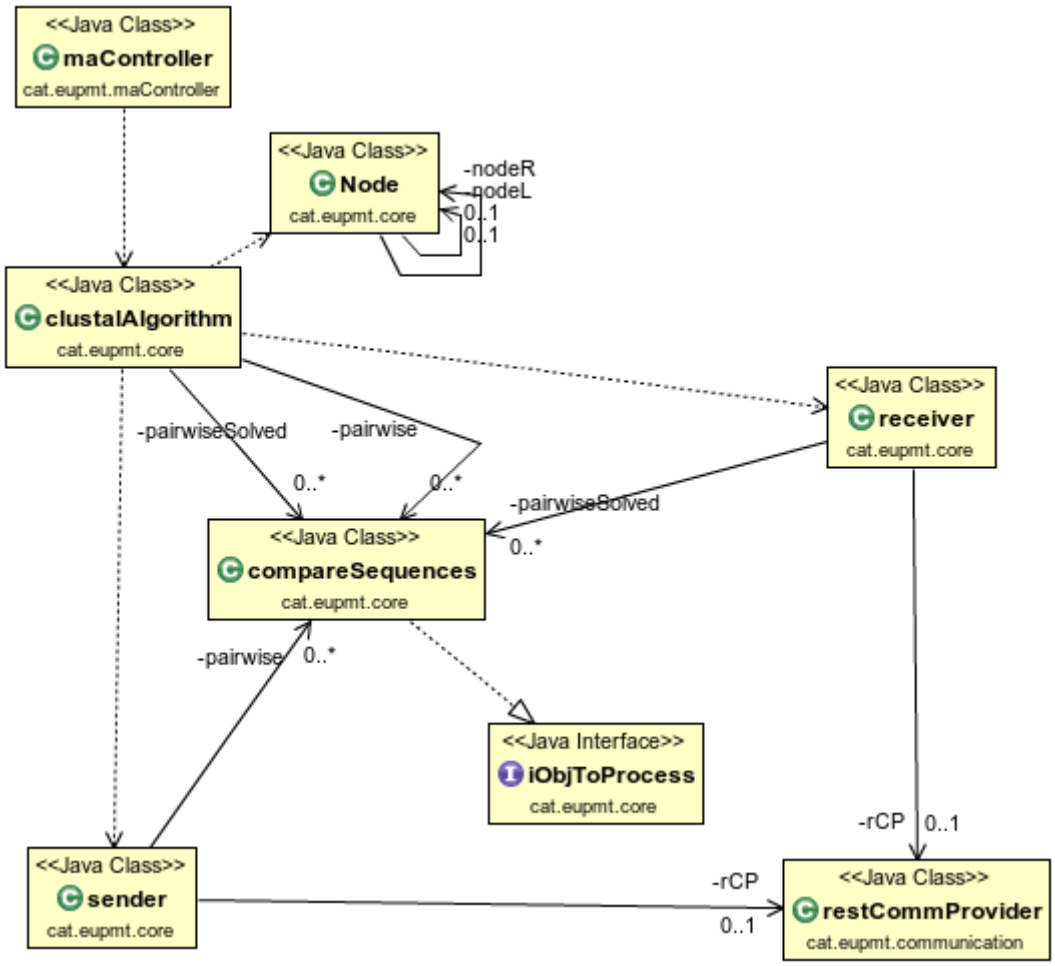


A3.4.3. Subsistema de processament

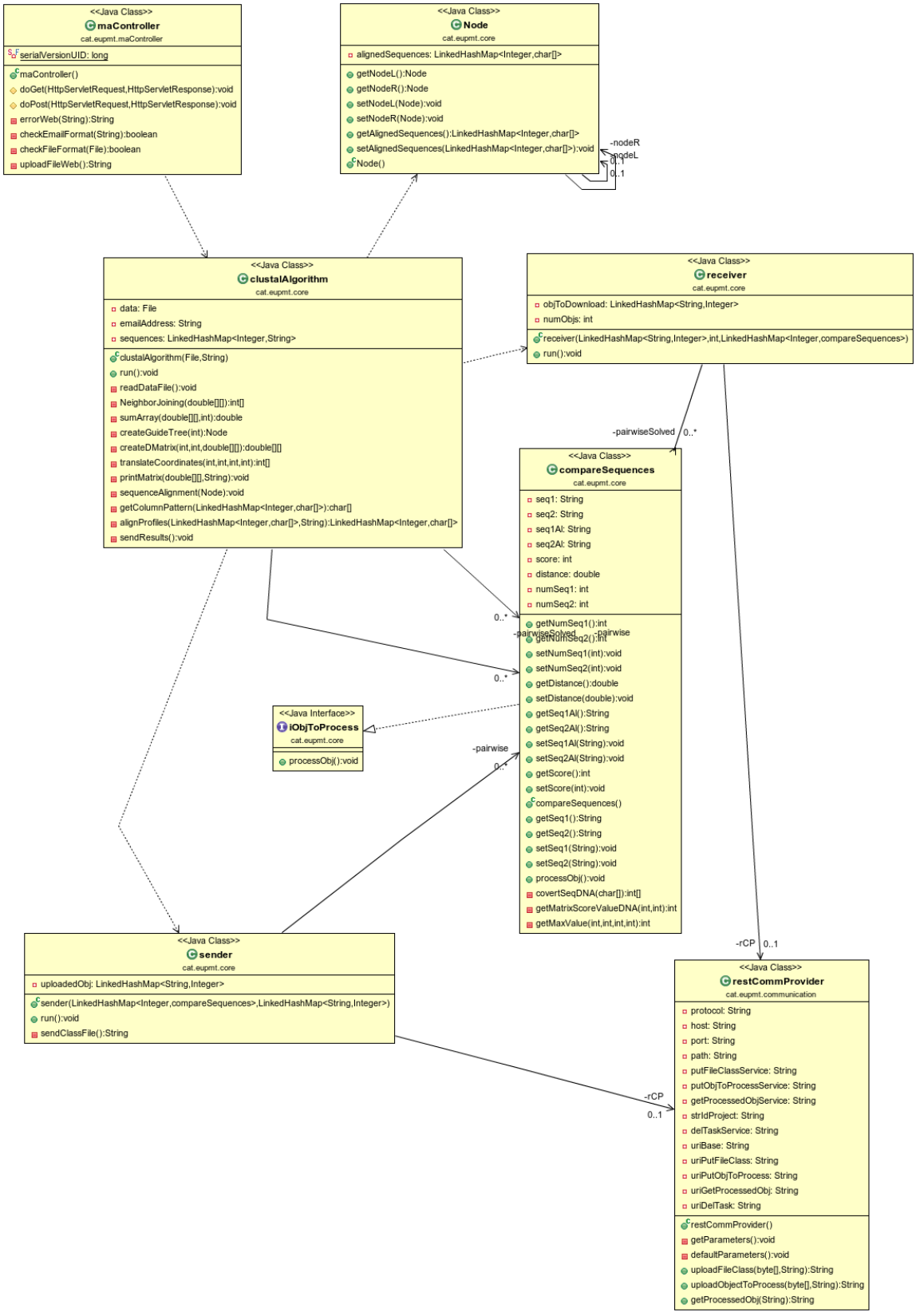


Annex 4. Documentació tècnica de l'aplicació amb funcions de proveïdor

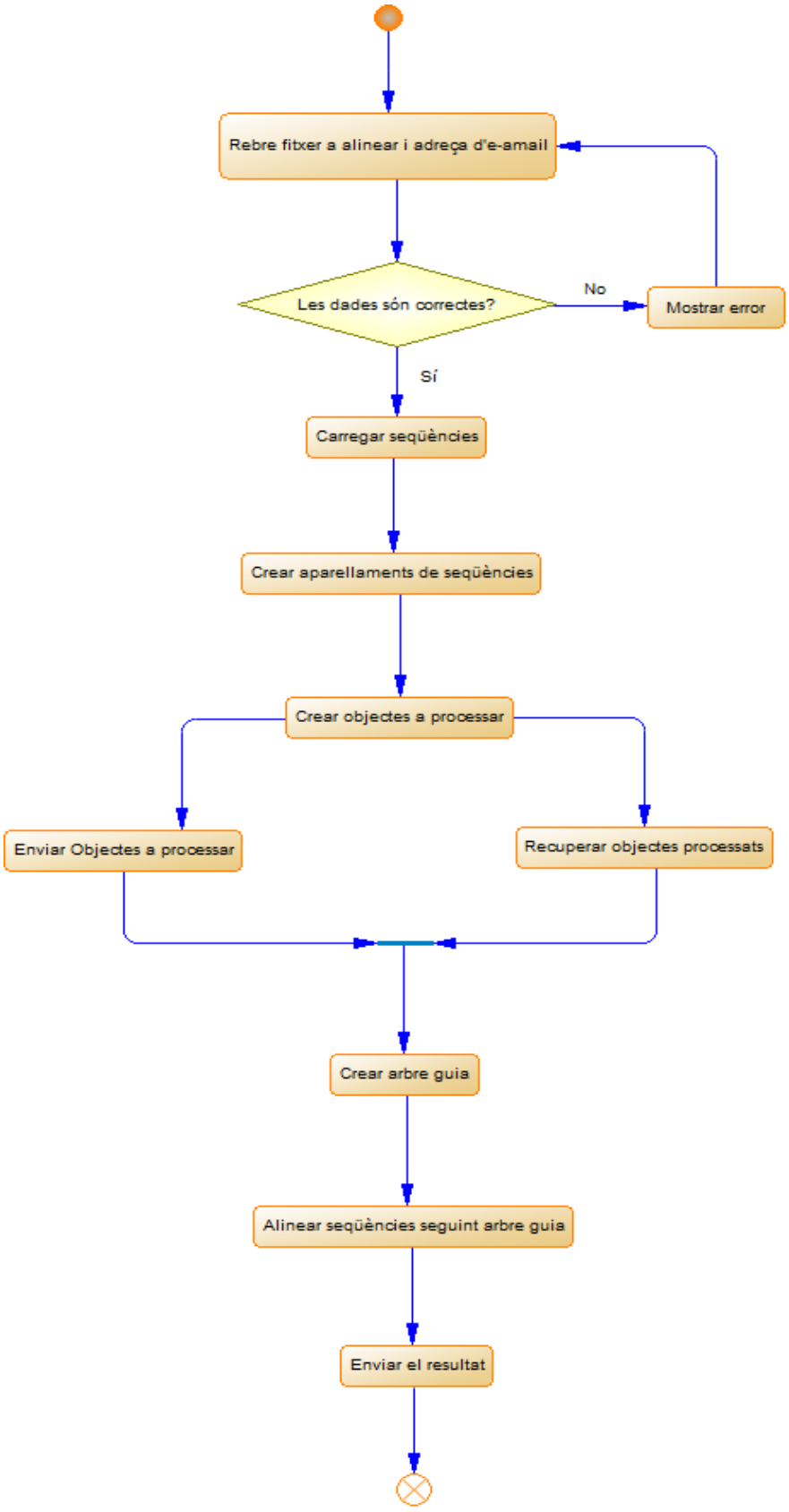
A4.1. Diagrama de domini



A4.2. Diagrama de classes



A4.3. Diagrama d'activitats



A4.4. Casos d'us

A4.4.1. Recepció fitxer fasta

<table>
<tr><td>Nom cas d'ús</td><td>Al1. Recepció fitxer fasta</td></tr>
<tr><td>Descripció</td><td>L'aplicació rep un fitxer que conté les seqüències a alinear</td></tr>
<tr><td>Actor principal</td><td>Usuari</td></tr>
<tr><td>Pre-condició</td><td>L'usuari a omplert els camps de fitxer i adreça d'e-mail del formulari d'enviament de dades</td></tr>
<tr><td>Flux bàsic</td><td>
<table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1</td><td>L'aplicació rep el fitxer i l'adreça d'e-mail a traves d'un formulari web.</td></tr>
<tr><td>2</td><td>Es comprova que l'arxiu sigui un arxiu en format Fasta correcte.</td></tr>
<tr><td>3</td><td>Es comprova que l'adreça de correu estigui en un format correcte</td></tr>
<tr><td>4</td><td>S'envien les dades al subsistema de processament</td></tr>
</table>
</td></tr>
<tr><td>Post-condició</td><td>Es mostra un missatge al usuari informant de que el procés està en marxa.</td></tr>
<tr><td>Flux alternatiu</td><td>
<table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>2</td><td>Es mostra un missatge d'error informant a l'usuari de la incidencia i el procés no continua.</td></tr>
</table>
</td></tr>
</table>

A4.4.2. Preparació d'objectes

<table>
<tr><td>Nom cas d'ús</td><td colspan="2">Al2. Preparació d'objectes</td></tr>
<tr><td>Descripció</td><td colspan="2">L'aplicació prepara els objectes que s'han de processar</td></tr>
<tr><td>Actor principal</td><td colspan="2">Controlador</td></tr>
<tr><td>Pre-condició</td><td colspan="2">El controlador a validat les dades.</td></tr>
<tr><td>Flux bàsic</td><td colspan="2"><table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1</td><td>S'extrauen les seqüències d'ADN de l'arxiu Fasta</td></tr>
<tr><td>2</td><td>Es creen els aparellaments de les seqüències</td></tr>
<tr><td>3</td><td>Es creen els objectes que seran enviats per que siguin processats amb els aparellaments creats.</td></tr>
</table></td></tr>
<tr><td>Post-condició</td><td colspan="2">Executa els subsistemes d'enviament i recepció d'objectes</td></tr>
<tr><td>Flux alternatiu</td><td colspan="2"><table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1, 2, 3</td><td>S'envia un missatge al controlador informant de l'excepció i el procés s'atura.</td></tr>
</table></td></tr>
</table>

A4.4.3. Enviar objectes a processar

<table>
<tr><td>Nom cas d’ús</td><td colspan="2">Al3. Enviar objectes a processar</td></tr>
<tr><td>Descripció</td><td colspan="2">L’aplicació envia els objectes per que siguin processats</td></tr>
<tr><td>Actor principal</td><td colspan="2">clustalAlgorithm</td></tr>
<tr><td>Pre-condició</td><td colspan="2">Els objectes estan creats.</td></tr>
<tr><td>Flux bàsic</td><td colspan="2"><table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1</td><td>S’envia l’arxiu .class al que pertanyen els objectes al servidor de parel·lelització</td></tr>
<tr><td>2</td><td>El servidor confirma que ho ha rebut correctament i retorna un identificador del objecte de Classe</td></tr>
<tr><td>3</td><td>S’envia l’objecte al servidor de parel·lelització perquè l’objecte sigui processat</td></tr>
<tr><td>4</td><td>El servidor confirma que ho ha rebut correctament i retorna un identificador del objecte a processar</td></tr>
<tr><td>5</td><td>L’identificador es guarda en un espai comú per que el subsistema de recepció d’objectes processats el pugui reclamar al servidor de paral·lelització.</td></tr>
</table></td></tr>
<tr><td>Post-condició</td><td colspan="2">--</td></tr>
<tr><td>Flux alternatiu</td><td colspan="2"><table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1, 3</td><td>Es torna a intentar l’enviament mentres no s’hagi superat el numero màxim d’intents i en cas contrari s’atura el procés i s’envia una excepció</td></tr>
</table></td></tr>
</table>

A4.4.4. Recepció d'objectes processats

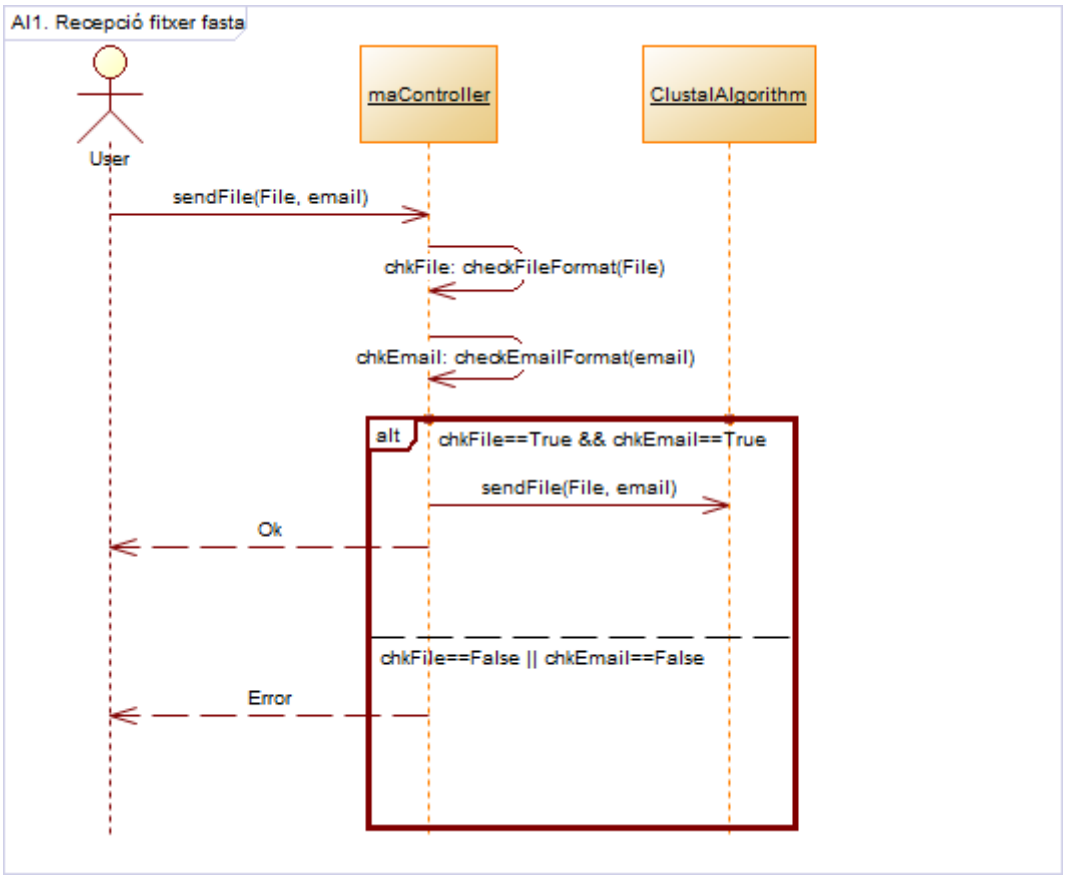
<table>
<tr><td>Nom cas d'ús</td><td colspan="2">Al4. Recepció d'objectes processats</td></tr>
<tr><td>Descripció</td><td colspan="2">Obté els objectes processats des del servidor</td></tr>
<tr><td>Actor principal</td><td colspan="2">ClustalAlgorithm</td></tr>
<tr><td>Pre-condició</td><td colspan="2">Hi ha objectes pendents de rebre</td></tr>
<tr><td>Flux bàsic</td><td colspan="2"><table><tr><th>Pas</th><th>Acció</th></tr><tr><td>1</td><td>Es tria un identificador d'objecte enviat a processar.</td></tr><tr><td>2</td><td>S'envia una petició de descàrrega del objecte identificat.</td></tr><tr><td>3</td><td>Es descàrrega l'objecte processat</td></tr></table></td></tr>
<tr><td>Post-condició</td><td colspan="2">Es marca l'objecte com descarregat per no tonar-lo a demanar</td></tr>
<tr><td>Flux alternatiu</td><td colspan="2"><table><tr><th>Pas</th><th>Acció</th></tr><tr><td>2</td><td>En cas de no obtenir resposta o que aquesta sigui nul·la es torna al pas 1.</td></tr></table></td></tr>
</table>

A4.4.5. Alineament múltiple de seqüències

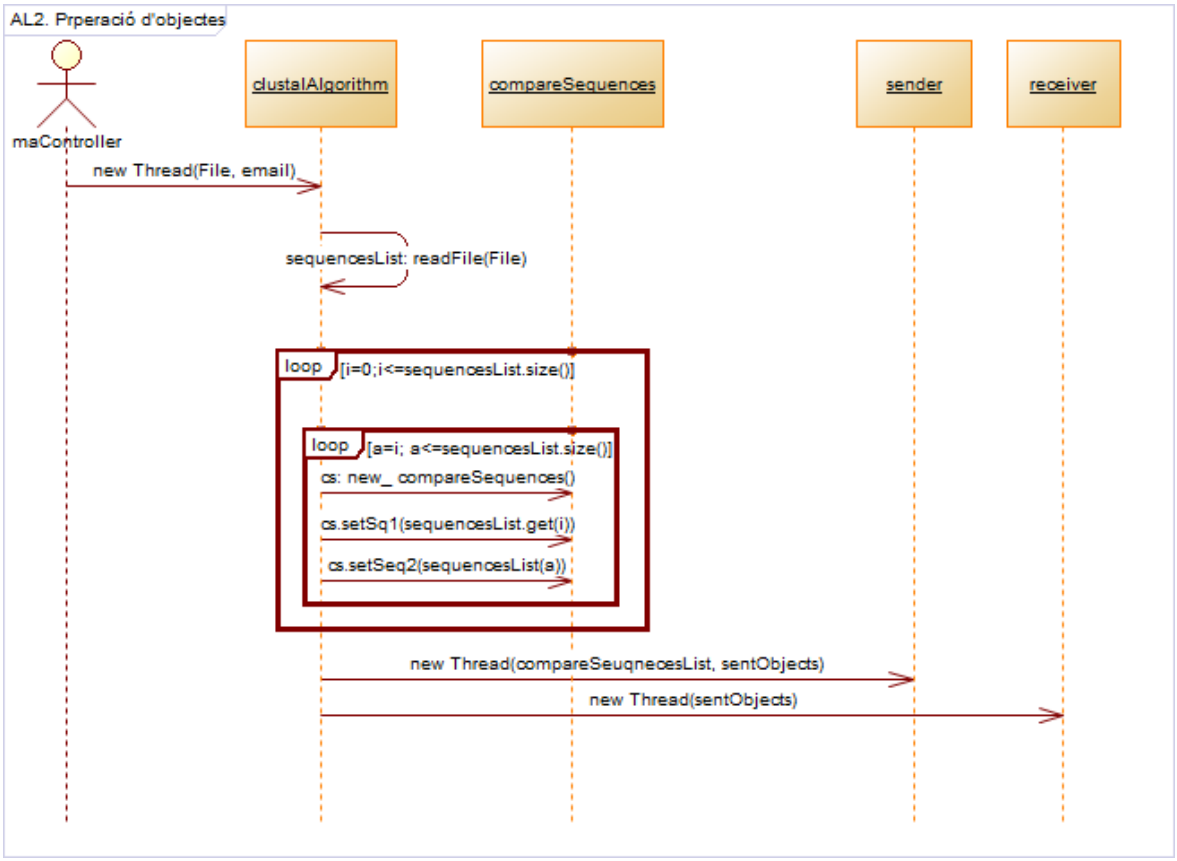
<table>
<tr><td>Nom cas d’ús</td><td colspan="2">Al5. Alineament múltiple de seqüències</td></tr>
<tr><td>Descripció</td><td colspan="2">S’alineen les seqüències d’ADN com descriuen els passos 2 i 3 del Algoritme Feng - Doolittle</td></tr>
<tr><td>Actor principal</td><td colspan="2">Aplicació</td></tr>
<tr><td>Pre-condició</td><td colspan="2">Els subsistemes d’enviament i recepció d’objectes han finalitzat la seva tasca.</td></tr>
<tr><td>Flux bàsic</td><td colspan="2"><table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1</td><td>Es crea la matriu de distancies a partir de la puntuació dels aparellaments.</td></tr>
<tr><td>2</td><td>Es crea l’arbre guia a partir de la matriu de distancies</td></tr>
<tr><td>3</td><td>Es fa l’alineament seqüencial utilitzant l’arbre guia per indicar l’ordre en el que s’han d’anar alineant les seqüències.</td></tr>
</table></td></tr>
<tr><td>Post-condició</td><td colspan="2">S’envia el resultat de l’alineament a l’adreça d’e-mail del usuari.</td></tr>
<tr><td>Flux alternatiu</td><td colspan="2"><table>
<tr><th>Pas</th><th>Acció</th></tr>
<tr><td>1, 2,3</td><td>Es retorna un missatge indicat l’excepció i s’atura el procés.</td></tr>
</table></td></tr>
</table>

A4.5. Diagrames de seqüència

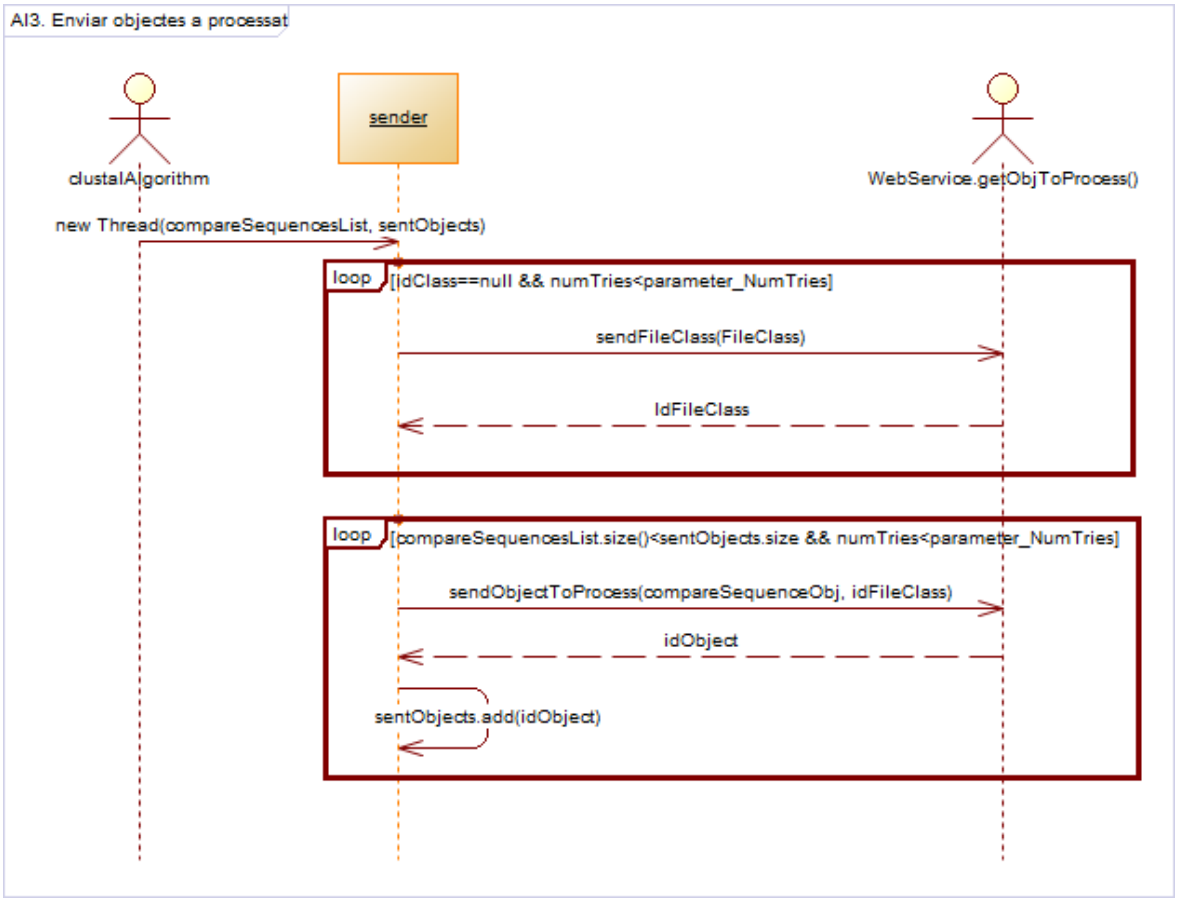
A4.5.1. Recepció fitxer Fasta



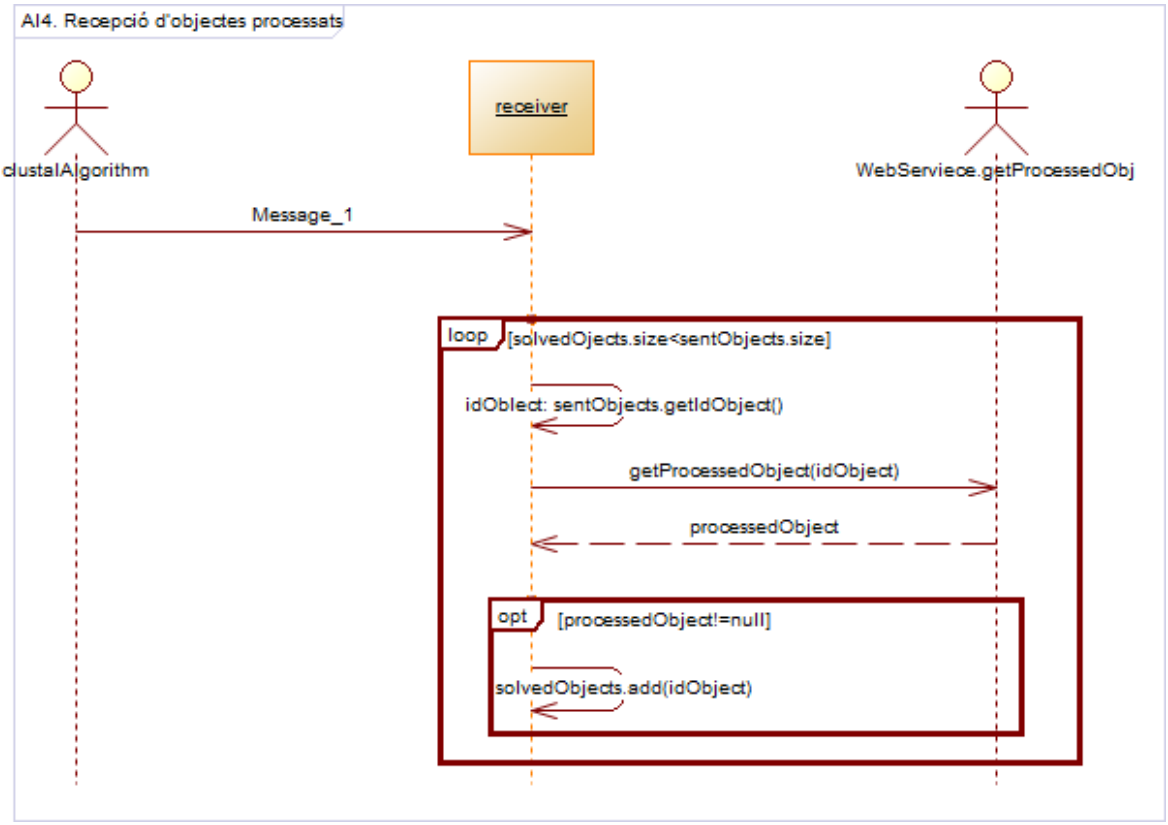
A4.5.2. Preparació d'objectes



A4.5.3. Enviar objectes a processar



A4.5.4. Recepció d'objectes processats



A4.5.5. Alineament múltiple de seqüències