Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN TECNOLOXÍAS DA INFORMACION
MENCIÓN EN ENXEÑARÍA DE COMPUTADORES

Starmind2: Entorno web para a visualización, análise e clasificación de espectros estelares

Estudante: Pablo Bugia López
Dirección: José Carlos Dafonte Vázquez
Dirección: Mª Alejandra Rodríguez Fernández

A Coruña, setembro de 2019.

Dedicado a uns e máis a outros, quen me aguantaron e máis quen non. Dalgunha maneira todos me axudaron.

Agradecementos

Agradezo a quen me axudaron a que algún día empezase a escribir esto sen ter máis que unhas breves nocións de Latex. Mención especial merecen os meus titores Alejandra e Carlos por aguantarme con tantas dúbidas, tantos correos e tantas reunións. Grazas a Carlos, alguén me brindou a oportunidade de facer un proxecto serio que me servise para as dúas mencións. Sen eles non sería posible nin siquiera empezalo. Tamén merece destacar a Minia Manteiga, a astrofísica do grupo de investigación, quen me sacou de dalgunha que outra situación de estancamento. Agradecementos para Ángel, pola súa gran colaboración co deseño da interface. Mención especial tamén merecen os compañeiros de laboratorio Dani, Marco, Arturo, Íker e máis Raúl.

Resumo

Este traballo de fin de grao implementa a evolución e reenxeñería dunha aplicación de escritorio adicada fundamentalmente a procesar e a clasificar espectros estelares, hacia un sistema web centralizado e distribuído que permita o uso simultáneo por parte de múltiples usuarios. Paraleliza zonas de código computacionalmente pesadas empregando a directiva *OpenMP*. Eliminouse a dependenza dun software propietario que dificultaba a utilización de estándares. O código finalmente quedou separado en capas.

Abstract

This end-of-grade work implements the evolution and re-engineering of a desktop application primarily focused on processing and classifying stellar spectra, towards a centralized and distributed web system that allows simultaneous uses by multiple users. Parallelize computationally heavy areas of code by undertaking *OpenMP* directive. I eliminated a dependency on proprietary software that made it difficult to use standards. The code is finally separated into layers.

Palabras chave:

Astronomía

Astrofísica

Espectros

Estrelas

Clasificación Morgan-Keenan

Metais en astrofísica

Luminosidade

VOtable

Sistema experto

Redes neuronais

Memoria compartida

Paralelización de código

openMP

openMPI

C++

Java

Python

Javascript

Django

Apache

UML

Metodoloxía

Keywords:

Astronomy

Astrophysics

Spectra

Stars

Morgan-Keenan classification

Metals in astrophysics

Luminosity

VOtable

Expert system

Neuronal networks

Shared memory

Code parallelization

openMP

openMPI

C++

Java

Python

Javascript

Django

Apache

UML

Methodology

Índice Xeral

Índice de Figuras

Índice de Táboas

Capítulo 1

Introdución

1.1 Introducción ao análise dos espectros estelares

Os obxectivos da astrofísica son o estudo da orixe, estructura e evolución dos obxectos celestes, polo cal compre recurrir a súa investigación cuantitativa e as leis físicas polas que se gobernan. Tendo en conta a lonxanía e a natureza dos fenómenos estudados dalle un aspecto singular a esta displina en comparación con outras ramas da física. A excepción a esto son os planetas e outros obxectos do sistema solar como poden ser asteroides..., no resto de corpos celestes non podemos elexir o instante ou lugar de observación nin influir, modificar propiedades ou mesmo obter mostras para realizar análises directos en laboratorio. Todo o que se coñece ata o de agora provén da súa observación totalmente pasiva, e unha grande parte desto provén do estudo detallado da radiación que emiten. A observación dos espectros estelares permite o suministro de información sobre a temperatura e outras condicións físicas das atmósferas estelares.

Dada a curiosidade e a importancia que se lle daba aos corpos celestas desde a antigüedade, dende hai moitos séculos a humanidade construiu edificacións de toda índole orientadas segundo a posición do sol nunha determinada época do ano, ou outros corpos celestes, a aparición de cometas sempre estivo enfocada a posibilidade de que ocurrisen fenómenos naturais de diversa índole, as estrellas podían servir para a navegación no mar...O centro deste proxecto será o estudo dun subconxunto dos corpos celestes, sendo as estrelas as escollidas. Aínda que existen de moitos tipos e tamaños, comparten moitas características comúns segundo a bibliografía consultada [1]. O enfoque deste proxecto irá hacia o estudo das estrelas, que só e posible a través dunha observación da radiación electromagnética que nos chega. Unha das fontes de información é captar a luz visible que emiten e descompoñela a través dun prisma nos seus cores fundamentais e logo medir o fluxo luminoso correspondente a cada lonxitude de onda. Esto é algo aparentemente moi sinxelo pero da unha idea bastante boa sobre a temperatura, a presenza de determinados elementos químicos coas súas proporcións e máis a idade

de moitas das estrelas existentes. Para que o espectro sexa o máis fiel posible é mellor que sexa obtido dende un satélite, pois a atmósfera terrestre engádelle ruído a radiación electromagnética provinte de diversos obxectos do universo. Además hai que ter en conta que aínda así é posible que os espectros conteñan ruído provocado polos efectos de atravesar zonas de po cósmico como poden ser nebulosas... Por outra parte sempre terán moitos menos ruído os espectros capturados dende un satélite que os obtidos na corteza terrestre, pois a atmósfera sempre lles engadirá algún ruído además de posibles desprazamentos en frecuencia.

1.2 Motivación

En canto aos motivos que me levaron a embarcarme nun proxecto relacionado coa astronomía, existen diferentes causas. Por un lado estaba o interese existente desde sempre en moitos temas relacionados conxuntamente coa física e astrofísica. Por outra banda quería facer un proxecto suficientemente amplo que me servise para optar a titulación de Grao en Enxeñería en Informática coas mencións en Tecnoloxías da Información e Enxeñaría de Computadores. A combinación destes dous factores, xunto co feito de que os titores levan moitos anos implicados en investigacións sobre o procesamento e a clasificacións de obxectos celestes, da lugar a me decidise por esta temática, aínda sendo plenamente consciente da carga de traballo.

Unha vez aceptado o reto, cando me presentaron a aplicación de escritorio para procesamento e clasificación de espectros estelares desenvolta no grupo hai anos, a idea en mente era poñela a disposición da comunidade científica, no ámbito da astrofísica computacional, implementando unha aplicación web centralizada que permita múltiples usuarios, además de permitir un procesado masivo de espectros. Por tanto, é necesario seguir un proceso de reenxeñería, observando as seguintes características e funcionalidades:

- Dispor dun sistema experto clasificador con reglas implementadas en OPS
- Uso de diferentes redes neuronais que refinan a primeira clasificación do sistema experto
- Identifica cando algún espectro está desplazado en frecuencia ou lonxitude de onda por efecto de varios factores como poden ser a atmósfera terrestre
- Xeración de gráficos cunha calidade moi boa
- Posibilidade de gardar facilmente as análises xeradas
- E outras moitas que poden pasar desapercibidas nos primeiros momentos

Vista desde o punto de vista actual, a aplicación ten dúas deficiencias importantes: O deseño do software non foi feito por capas, e foi implementada cunha ferramenta propietaria (Borland C/C++ Builder 5).

Probablemente o feito de non utilizar o deseño basado en capas foi motivado ao uso desa ferramenta que en principio non facilitaba esta labor. Outro factor que con bastante probabilidade motivou que non se empregasen capas foi o feito de dispoñer de moi pouco tempo para a súa construcción e os cambios posteriores. Outra posible pega é o feito de depender dunha ferramenta propietaria. No seu momento facilitou a creación de gráficos e o manexo da interfaz gráfica dando un aspecto visual moi bo pra aquela época. Podemos resumir as deficiencias atopadas nos seguintes aspectos:

- Dificulta moito a separación do código en capas
- Só se executa sobre sistemas Windows
- É unha aplicación de escritorio que precisa dunha instalación en cada equipo no que se vaia utilizar, ademáis de ser bastante laborioso de extender posibles actualizacións e melloras
- Presenta unha dificultade nada desprezable de actualización e mesmo defectos que se poidesen presentar debido fundamentalmente ao non estar separadas a lóxica de negocio coa parte de presentación e visualización
- Desaproveitamento dos actuais procesadores multicore e sistemas operativos de 64 bits

Independentemente do tema relacionado coa astronomía, había tamén dúas características que me interesaban moito do sistema Starmind:

1. O uso de redes neuronais nas súas moitas variantes seguen sendo unha forma eficaz de resolución de problemas de diversa índole
2. O feito de que a aplicación orixinal estaba codificada na linguaxe C++

Este último feito tamén resulta moi interesante, é unha linguaxe que non é impartida no grao, conservando a potencia do C orixinal é plenamente orientado a obxectos, con moitas máis características que Java, con tódalas ventaxas e algunhos inconvintes que poida levar consigo. Tamén está presente a idea de optimizar e paralelizar código empregando algunha das tecnoloxías existentes. Relacionado coa paralelización está o rendemento, non importando só o tempo de resposta, senón tamén o axeitado uso dos recursos computacionais. Outro aspecto interesante é a implementación de aplicacións con certo grado de seguridade e fiabilidade.

As aplicacións paralelas independentemente da súa tecnoloxía, son máis complexas a hora de desenvolver, pero a cambio aproveitase a natureza multinúcleo dos procesadores actuais.

Evidentemente existen tarefas que dada a súa natureza intrínseca de dependencia non poden ser paralelizadas. A tarefa de pasar un código secuencial a paralelo supón varios desafíos, como poden ser as dependencias entre instruccións, limitando en última instancia a paralelización en calquera das actuais tecnoloxías existentes hoxe en día.

1.3 Obxectivos

1. Construcción dunha aplicación web que permita múltiples usuarios con acceso simultáneo. Hoxe en día, nunha aplicación de ámbito científico, parece máis razoable que sexa web, eliminando a dependencia de determinadas plataformas de hardware e software.

2. Eliminación da dependencia da ferramenta Borland C/C++ Builder. Esta ferramenta seguramente no seu momento facilitou gran parte de traballo inicial, pero a simple vista non se lle atopan máis vantaxes. O non depender dunha ferramenta propietaria fai máis fácil a manteñibilidade e actualización segundo transcurra o tempo e evolucionen as tecnoloxías.

3. Uso da capacidade computacional das máquinas actuais (multinúcleos, ancho de palabra de 64 bits, instruccións SSE...) sen limitar a portabilidade do código a unhas determinadas plataformas de arquitectura ou mesmo sistema operativo. Este software non precisa dunha plataforma concreta, senón que está pensado para adaptarse as arquitectura existentes, evidentemente rendindo máis nunhas plataformas de hardware que noutras, pero traballando correctamente en calquera caso.

4. Creación dunha aplicación que non dependa do sistema operativo do usuario nin do software que teña instalado, non requerindo máis que un navegador web que cumpla determinados estándares. A maioría deses navegadores hoxe en día cumplen eses estándares.

5. A aplicación creada debe ser realmente escalable segundo o número de clientes. Se aumenta o número de expertos en astrofísica que a utilizan, precísase que escale ben en relación a cantidade e capacidade das CPUs e a cantidade de memoria dispoñible.

6. A aplicación precisa ter capacidade de administración básica de usuarios.

7. Uso dunha conexión cifrada *https* entre o navegador de cada usuario final e o servidor ou servidores da capa modelo, pois sempre fai máis difícil un ataque de tipo *man in the middle* ou mesmo *spoofing*. En moitos casos é obligatorio utilizar unha conexión *https* en vez de *http*.

8. Migración de sistemas expertos clasificadores e máis de tódalas redes neuronais empregadas.

Capítulo 2

Fundamentos teóricos

Neste capítulo tratarase sobre os fundamentos da análise espectral de estrelas, sobre o estado do arte dun proxecto e por último comentaranse brevemente as tecnoloxías empregadas. O autor deste TFG pensa que non se pode empezar a falar directamente dun software coma este, de tipo científico, sen ter unhas nocións moi básicas de diversos temas de física e astronomía, con todo é algo moi básico, se o lector precisa afondar algo máis nalgún dos temas, debe recurrir a literatura, sexa a bibliografía empregada [2], [1], [3]..., ou ben a calquera que trate sobre astrofísica cun grao de rigurosidade e exactitude.

Na primeira parte deste capítulo inténtanse explicar con certa rigurosidade aspectos da física, os cales permitirannos intuír algunhos aspectos do tema astronómico. Non se indicarán teoremas ou postulados que requiran unha base matemática ou física moi sólida cun nivel moi profundo, senón que se definiran da forma máis clara e concisa posible conceptos moi importantes en astronomía. Ademáis da definión de conceptos, tamén se indicarán rangos de diversas magnitudes, xa que os valores utilizados en astronomía distan moito aos que se soen empregar en calquera das outras disciplinas científicas.

Na segunda parte comentaranse tódolos traballos relacionados dos que se teña constancia que fan algo bastante similar ao que se pretende conseguir con Starmind2. Cando se inicia algo novo é importante saber se temos ou non referencias, a cantidade e a calidade delas, pois onde outros equipos tropezaron, aplicando coñecemento e planificación é máis fácil evitar a repetición deses problemas, pois xa son coñecidos ata certo grado de amtemán. Pero como se verá prácticamente non hai nada coñecido e suficientemente relacionado, salvo a aplicación de partida. Quitando Starmind orixinal non se atopan proxectos moi similares.

Na terceira parte falarase sobre as tecnoloxías empregadas, que son varias, e moi distintas entre si. Segundo se vaia avanzando na lectura iranse comprendendo os verdadeiros motivos que levaron ao equipo a decidir utilizar tantas, e tan diferentes entre sí.

2.1 Aspectos fundamentais sobre a análise espectral

2.1.1 Definicións dalgunhos termos moi empregados

Espectro electromagnético E a distribución enerxética do conxunto das ondas electromagnéticas dunha forma similar a mostrada na figura 2.1. Cando se refire a un obxecto denomínase espectro electromagnético ou simplemente espectro a radiación electromagnética que emite (espectro de emisión) ou absorve (espectro de absorción) unha sustancia.

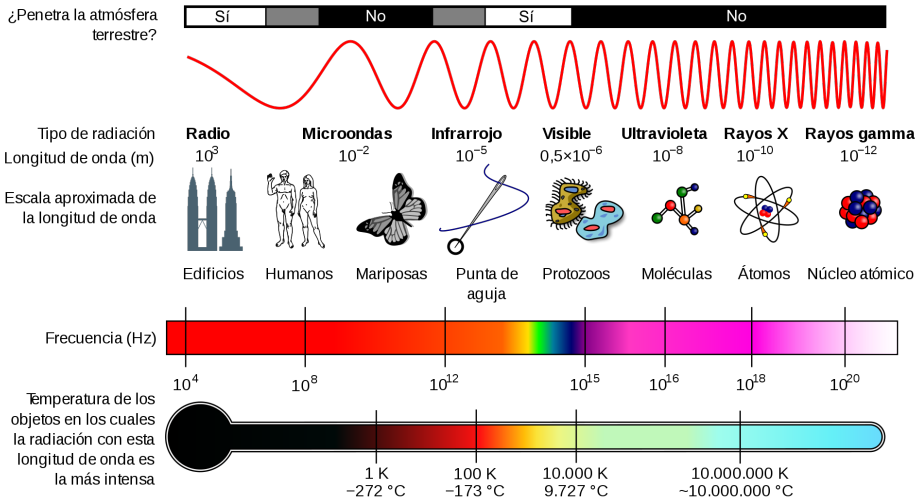


Figura 2.1: Espectro electromagnético

Unha maior lonxitude implica unha maior difusión. O espectro visible é o espectro de radiación electromagnética que é visible para o ollo humano, e vai dende a lonxitude de onda de 400 nm ata os 700 nm segundo o observado na figura 2.2.

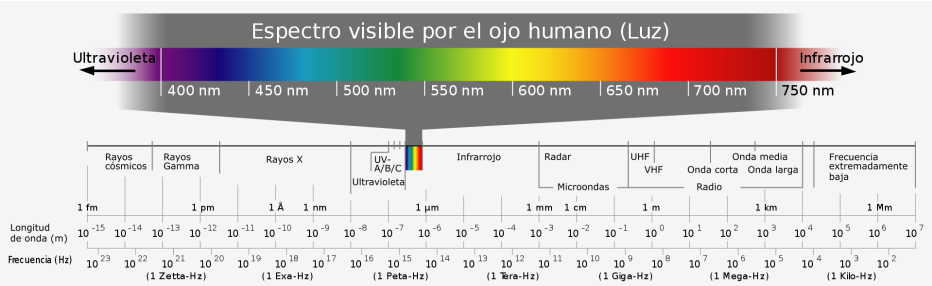


Figura 2.2: Espectro visible

Efecto Dopper: É o cambio de frecuencia aparente dunha onda producido polo movemento relativo da fonte respecto ao seu observador. Cando a fonte emisora vaise acercando cada vez máis ao observador, as lonxitudes de onda comprímense, e cando se alonxa as lonxitudes de onda estíranse. Un exemplo disto é cando unha persoa sabe perfectamente cando se acerca unha ambulancia e tamén cando se alonxa sen necesidade de contacto visual ningún debido ao son xenerado pola sirea. Cando se acerca a ambulancia ao observador, o son faise máis agudo e cando se alonxa faise máis grave como pode ser observado na figura 2.3. Este efecto é moi importante en astrofísica, pois adquire relevancia cando é aplicado as on-

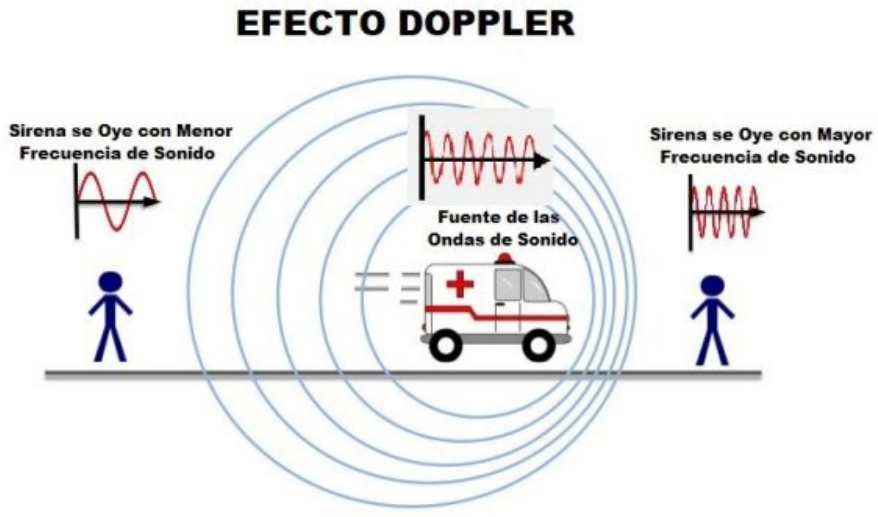


Figura 2.3: Efecto Dopper en ondas sonoras

das electromagnéticas, sobre todo na caso da luz visible. Evidentemente nesta disciplina non ten sentido aplicado ao son, posto que as ondas sonoras caracen de medio de propagación

no vacío. Cando un obxecto emite luz como poder ser unha estrela, galaxia..., sabemos se se acerca porque se comprimen as ondas de luz emitidas desprazándose hacia o azul. Se o corpo emisor alónxase alárganse as ondas desprazándose ata o vermello. Este efecto é mais intenso canto maior sexa a velocidade relativa entre o emisor e o receptor, o cal permite usar o efecto Dopper para calcular a velocidade de corpos celestes que emiten luz respecto a un observador terrestre ou situado nunha sonda de investigación.

Metais: En astrofísica considéranse metais a tódolos elementos máis pesados que o Helio, polo que neste traballo cando se fai referencia aos metais, de non dicirse ao contrario, será pra indicar tódolos elementos que teñen máis masa que o helio. Esto será de aplicación en todo este traballo.

Estrela: É un corpo formado por un cúmulo de materia en estado de plasma que irradia enerxía producida por reacciones de fusión nuclear que ocurren no seu núcleo. As súas propiedades dependen principalmente de súa masa e enerxía desprendida nas súas reaccións nucleares. O tamaño da estrelas clasificase entre 0.06 e 100 a masa do sol. O sol, como se verá máis adiante clasifícase como unha estrela enana.

Estrela binaria: É un sistema estelar composto de dúas ou máis estrelas que orbitan mutuamente alredor dun centro de masas común. Recente estudos suxiren que un elevado porcentaxe das estrelas son parte de sistemas que teñen polo menos dous astros. Os sistemas múltiples, poden ser ternarios, cuaternarios, ou incluso de cinco ou máis estrelas interactuando entre sí. Aínda que sexa un conxunto de máis de dúas tamén se denominan estrelas binarias, como é o caso de Alfa Centauri A y B e Próxima Centauri. Debido a gran cantidade de estrelas aparentemente binarias existentes no universo, foi necesario desenvolver formas para distinguir as que son verdaderamente binarias das que só parecen parecen selo, pero realmente é só unha cuestión óptica. Esta situación xurde cando dous astros separados por grandes distancias e sen relación gravitatoria mútua, pode ocorrer que se vexan moi cercanos dende a nosa perspectiva. Tamén se teñen dado casos nos que as estrelas de luminosidade cambiante parecen ser binarias eclipsantes cuando en realidade non o son. Se ben existen pares de estrellas orbitando tan lonxe unha de outra como para evolucionar de forma independente, en moitas ocasións as binarias atópanse a distancia tan relativamente corta que o seu proceso individual vese alterado polos cambios que sufre a súa compañeira, de forma que estos obxectos evolucionan coma un todo, creando obxectos que de non ser pola atracción mútua non serían posibles.

Galaxía: É un conxunto de estrelas, nubes de gas, planetas, po cósmico, matería oscura e enerxía unidas gravitacionalmente nunha estructura máis ou menos definida. A cantidade de estrelas que forman unha galaxia é enorme e vai das 107 das enamas ata as 1000 e pico das xigantes. Formando parte das galaxias existen subestructuras coma nebulosas, cúmulos estelares e sistemas estelares múltiples.

Púlsar: É un termo (do acrónimo en inglés pulsating star, que significa "estrela que emite radiación moi intensa a intervalos cortos e regulares"). É unha estrela de neutróns que emite radicación periódica. Este tipo de obxectos teñen un intenso campo magnético que induce a emisión destos pulsos de radiación electromagnética a intervalos regulares relacionados co periodo de rotación. Estos obxectos poden xirar sobre sí mesmos ata varios centos de veces por segundo de forma que un punto na súa superficie pode estar movéndose a velocidades da orde de 70000 km/s. De feito, xiran tan rápidamente que expanden o seu ecuador debido a súa velocidade vertixinosa. Estas estrelas teñen un tamaño entre 10 e 20 km, xa que a forza centrífuga xerada a esta velocidade é enorme e só o seu potente campo gravitatorio é capaz de evitar que se despedace.

Nebulosas: Son rexións do medio interestelar constituidas por gases (fundamentalmente hidróxeno e helio) xunto con elementos químicos en forma de po cósmico. Teñen unha gran importancia cosmolóxica porque moitas delas son os lugares onde nacen as estrelas por fenómenos de condensación e agregación da materia; Noutros casos trátase de restos de estrelas xa extintas ou en vías de extinción. As nebulosas asociadas con estrelas novas localízande nos discos das galaxías espirales e en calquera zona das galaxías irregulares, pero non se soen atopar en galaxías elípticas, xa que case non se producen fenómenos de formación estelar e están formadas por estrelas moi vellas. Un exemplo de nebulosa é a da imaxe 2.4.

Figura 2.4: Nebulosa cabeza de cabalo

Cúmulos estelares: É un grupo de estrelas atraídas entre sí debido a súa gravedade mútua. A clasificación tradicional inclúe dous tipos: Os cúmulos globulares e os cúmulos abertos (ou galácticos). Os globulares son agrupacións densas de centenares de miles ou millóns de estrelas de moita idade (máis de mil millóns de anos), mentres que os cúmulos abertos conteñen xeralmente centenares ou millares de estrelas novas (menos de cen millóns de anos) ou

de idade intermedia (entre cen millóns e mil millóns de anos). Os cúmulos abertos son disgregados ao largo do tempo pola súa interección gravitatoria con nubes moleculares no seu movemento pola galaxía mentres que os cúmulos globulares, máis densos, son máis estables fronte a súa disgregación (aínda que a largo prazo tamén acaban destruídos). Ademáis das diferenzas en número de estrelas (e polo tanto en masa tamén) e por idade entre os dous tipos distintos, tamén se distinguen pola súa metalicidade (os abertos son máis ricos en metais mentres que os globulares son pobres neles) e pola súa órbita (os abertos pertencen a poboación do disco da galaxia mentres que os globulares pertencen ao halo). Non existen diferenzas entre os tamaños dos núcleos de ambos, que é dunhos poucos pársecs. Unha clasificación moderna nas agrupacións estelares (cúmulos ou asociacións) debe incluir polo menos tres variables: Idade, masa e estado gravitacionan, ademáis de metalicidade e tipo de órbita. Un exemplo de cúmulo é o mostrado na imaxe 2.5.

Figura 2.5: Cúmulo estelar M31

Quasar: É unha fonte astronómica de enerxía electromagnética, que inclúe radiofrecuencia e luz visible. Son os obxectos celestes máis luminosos que poidamos observar. Estos fenómenos xurden cando un enorme furado negro, situado no núcleo dunha galaxia comeza a absorver toda a materia que se atope no seu radio de acción. Cando sucede esto por efecto da enorme velocidade de atracción formada, produce unha inmensa cantidade de enerxía, liberada en forma de ondas de radio, luz, infravermello, ultravioleta e raios X, o que os convirte nos obxectos celestes máis brillantes conocidos.

Liñas espectrais A luz ao atravesar un prisma descomponse en cores, pero en certos cores fanse ocos que parecen liñas escuras. Estos ocos denomínanse liñas espectrais. As liñas escuras da maioría das estrelas representan unha disminución da radiación electromagnética e certas lonxitudes de onda e denomínanse liñas de absorción A razón de por aquí este concepto

ven dos tempos de Isaac Newton, quen descompoñeu a luz solar por medio dun prisma dando lugar ao coñecido experimento, que está moi relacionado con este traballo. Este experimento é algo similar nostrado pola figura 2.6. Estes espectros obtidos a partir de observacións estelares como se verá dan moita información sobre as atmósferas estelares

Figura 2.6: Descomposición da luz

Liñas de emisión Cando se quenta un gas os átomos excítanse, pero só poden permanecer neste estado así un certo tempo, chamado vida media, ao cabo do cal desexcitanse e pasan ao estado fundamental emitindo fotóns. Estos fotóns representan enerxía a certas lonxitudes de onda que corresponden no espectro a lonxitudes de onda brillantes, é dicir, liñas de emisión. Un exemplo é a imaxe 2.7.

Figura 2.7: Liñas de emisión

Liñas de absorción Cando a radiación continua dun corpo a unha temperatura alta pasa a través dun gas frío, os átomos que constitúen o gas atrapan enerxía da radiación continua e excítanse a estados superiores de enerxía, son as liñas de absorción que no espectro visible aparecen como liñas escuras. Se un espectro continuo pasa a través do vapor dun elemento e despóis a través doutro vapor dun elemento diferente, ou a través dunha mezcla dos dous gases, o espectro de absorción que resulta mostra as liñas de absorción. Un exemplo é a imaxe 2.8.

Corpo negro É un obxecto teórico que absorve toda a luz e enerxía radiante que lle chega. Non refleza nin pasa ninguha da radiación incidente sobre el. A diferenza entre un corpo negro

Figura 2.8: Liñas de absorción

da materia escura é que un corpo negro emite luz constituindo un sistema físico idealizado para o estudo da radiación electromagnética. O nome do corpo negro veu dado polo físico Gustav Kirchhoff.

Todo corpo emite enerxía en forma de ondas electromagnéticas, sendo esta radiación máis intensa canto máis elevada sexa a temperatura do emisor. A enerxía radiante emitida por un corpo a temperatura ambiente é escasa e corresponde as lonxitudes de onda máis largas que as da luz visible, como pode ser a luz infravermella. Cando se aumenta a temperatura dun corpo non só aumenta a enerxía emitida, senón que se fai a través de lonxitudes de onda máis cortas. A esto é debido o cambio de color dun corpo cando se quenta. Os corpos non emiten igual intensidade a tódalas frecuencias, senón que seguen a lei de Planck. A mesma temperatura, a enerxía emitida depende da natureza da superficie. Según a lei de Kirchhoff establece que un corpo que é bo emisor de enerxía é tamén un bo absorvente de dita enerxía.

2.1.2 Magnitudes e rangos de valores empregados en astrofísica

Os rangos de magnitudes utilizados es astrofísica son moi diferentes aos utilizados en calquera outra disciplina dada a súa singularidade, por eso é interesante ter unhas nocións básicas sobre estos rangos antes de empezar a empregalos. Na maior parte dos casos son valores inimaxinables tanto na corteza coma no núcleo terrestres.

1. Unidades de distancia
    - 1 Å = 10^{-10} m (é moi utilizada pra as lonxitudes de onda)
    - A unidade de distancia é o parsec (1 pc = $3.086 \cdot 10^{13}$ km = 3.26 anos luz)
    - No universo extragaláctivo emprégase o megaparsec (1 Mpc = 10^6 pc)
2. Tamaño de estructuras
    - Graos de polvo interestelar de $2 \cdot 10^{-5}$ cm
    - Supercúmulos de galaxías de 10^{22} km
3. Temperaturas
    - Nas rexións de hidróxeno neutro 10 K
    - Nas explosións de supernovas 10^9 K

4. Densidades
    - Nas rexións de hidróxeno neutro $2 \cdot 10^{-26}\ g/cm^3$
    - Nos furados negros $10^{16}\ g/cm^3$
5. Campos magnéticos
    - No medio interestelar 10^{-10} T
    - Nos estrelas de neutróns 10^{8} T

Estes rangos parecen moi grandes, pero poden ser aínda moito maiores se temos en conta as condicións iniciais do universo [3]

2.1.3 Clasificacións estelares

A clasificación estelar é a clasificación en función das súas características espectrais. A radiación electromagnética procedente dunha estrela é analizada mediante a división por un prisma ou por unha rede de difracción nun espectro, mostrando así o arcoiris de cores (coincide co espectro electromagnético visual) entremezclado con liñas de absorción. Cada liña indica un ión dun determinado elemento químico, xunto coa intensidade da liña determina a abundancia dese ión. A abundancia relativa dos diferentes ións varía coa temperatura do fotoesfera. A *clase espectral* dunha estrela é un código corto que resume o estado da ionización, dando unha medida obxectiva sobre a temperatura da fotoesfera e a súa densidade.

A maioría das estrelas están actualmente clasificadas polo sistema de Morgan-Keenan (MK) na figura 2.9, utilizando as letras O, B, A, F, G, K e M, unha secuencia que abarca desde as máis quentes (tipo O) as máis frías (tipo M). Cada clase de letra subdivídese usando un díxito numérico co 0 para as máis quentes e un 9 para as máis frías dentro desa mesma letra. A secuencia inicial foi ampliada con clases doutras estrelas e outros obxectos parecidos as estrelas que non encaixan no sistema clásico como son a clase D (enanas brancas) e clase C (estrelas de carbono).

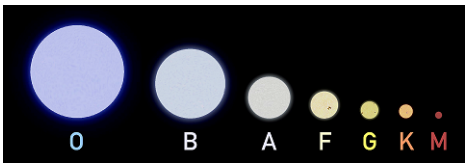


Figura 2.9: Clasificación en función da temperatura

No sistema MK, engádese unha *clase de luminosidade* a clase espectral usando números romanos segundo a figura 2.10. Está basado no ancho das liñas de absorción no espectro dunha estrela, as cales varían coa densidade da atmósfera distinguíndose as xigantes vermellas das

enanas. A clase de luminosidade 0 ou as estrelas Ia+ son hiperxigantes, a clase de estrelas I para a superxigantes, a clase II para as xigantes brillantes, a clase III para xigantes regulares, a clase IV para as subxigantes e a clase V para as estrelas da secuencia principal, a clase sd son subenanas e a clase D para enanas blancas. A clase espectral completa para o sol é G2V, indicando que é unha estrela da secuencia principal cunha temperatura aproximada de 5800K.

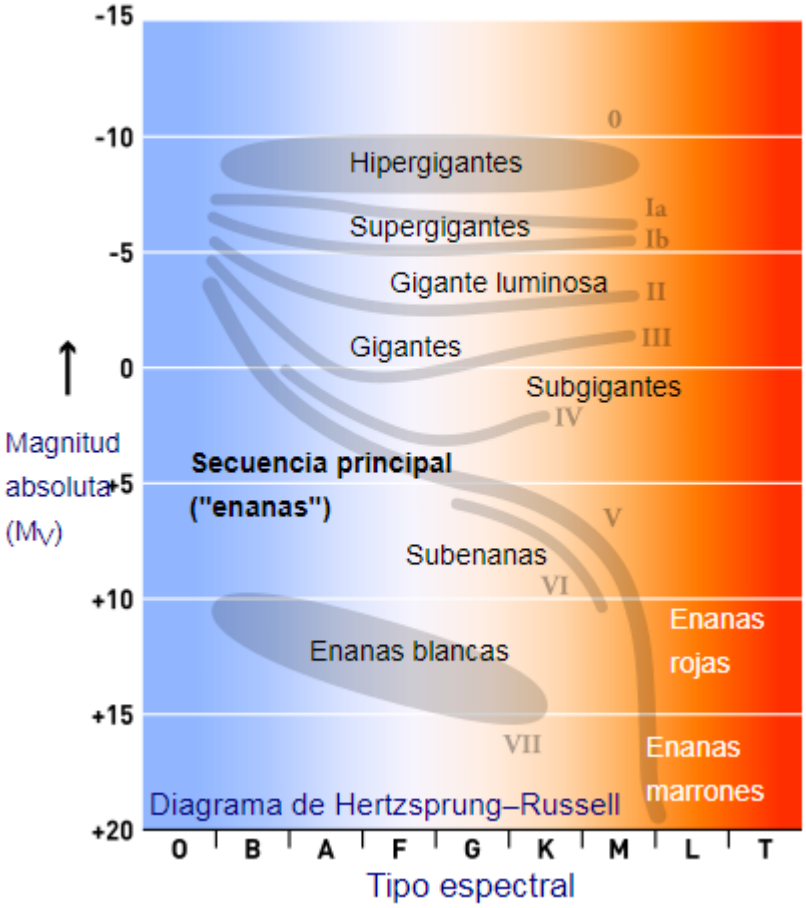


Figura 2.10: Clasificación en clases de luminosidade

2.1.4 Aspectos previos ao análise dos espectros estelares

Se ben se trata de reconstruir unha aplicación para o tratamento e análise de espectros estelares é importante contextualizala nun ámbito no que se vaia usar. A orixe desta disciplina é a experiencia de Newton de descompoñer a luz solar coa axuda dun prima nunha banda continua de cores denominada espectro. Máis adiante Wollanston en 1802 detecta sete liñas escuras no espectro solar, pero non foi ata unha década máis tarde cando Fraunhoffer observou e mediu cuidadosamente as posicións de máis de 500 liñas escuras. Posteriormente Kirchhoff, nos seus traballos de laboratorio, chegou a conclusión de que as liñas escuras se atribúen a transicións específicas dos átomos excitados. Isto facilitou a rápida identificación de moitos elementos químicos na atmósfera do sol.

O descubrimento da fotografía levou a un rápido avance na aplicación da espectroscopia a astronomía. En 1863 Huggins obten os dous primeiros espectros estelares abrindo unha nova era na astronomía.

Por outra parte a observación dos espectros estelares permite o suministro de información sobre a temperatura e outras condicións físicas das atmósferas estelares [2], [1], así que será o obxectivo primario polo que se rexirá este proxecto. Esto vai dirixido a astrofísicos, astrónomos e en menor medida, a físicos e a astrónomos aficionados. Como as estrelas son un subconxunto significativo de obxectos celestes o proxecto irá encamiñado ao procesado e a clasificación de espectros estelares, dando características coma o seu tipo segundo a temperatura, luminosidade, bandas de diferentes elementos, bandas de enerxía...

Por último convén ter en conta o que pasa coa maioría das estrelas, excepto do sol, que se pode observar con máis detalle durante os eclipses totais de sol, e estudo dos espectros é a única forma de obter coñecemento das estrellas, pois hai que ter en conta que están moi lonxe do sistema solar, e non se coñece outra forma de saber as súas propiedades máis que forma totalmente pasiva. Aínda que no hipotético caso de que se poidesen enviar sondas fora do sistema solar, a unha velocidade algo próxima a da luz, a sonda tardaría miles ou millóns de anos en chegar a unha distancia relativamente cercana aos obxectos a observar. Supoñendo que estivese a unha distancia correcta do obxecto a estudar, canto tempo tardaría en chegarnos a terra a información a velocidade da luz? Coas leis físicas que explican a maioría dos fenómenos que ocorren no universo, é algo totalmente inviable, o cal dalle toda a importancia ao estudo dos espectros que recibimos de obxectos celestes moi lonxanos. É importante aclarar que as veces referímonos co termo obxecto estelar en vez de estrela. O termo obxecto abrangue a tódolos corpos celestes, sexan clasificados ou sexan sen clasificar. Unha definición de estrela moi simple é a de obxecto celeste que emite luz visible xunto con outras formas de radiación dentro do conxunto do espectro electromagnético, é dicir, é algo activo frente a algo pasivo como poden ser os planetas, nebulosas, asteroides...

2.2 Estado da arte

A aplicación de escritorio da que se parte sentou un precedente fundamental e vital para a construcción da versión a implementar neste proxecto. É moi importante dispoñer dunha versión de referencia nun traballo destas características, pois é un tema en certa medida bastante descoñecido e complexo para o autor. Starmind orixinal emprega técnicas tan diversas que ao autor deste traballo, de non ser por esa referencia, non podería ter conseguidos a meirante parte dos obxectivos de Starmind2. Esto é debido en parte a que é un software de tipo científico, e nunha enxeñería de calquera tipo impártense moitos máis aspectos técnicos que puramente científicos.

O autor quedou abraiado polo número tan grande de funcionalidades e tan complexas dende o primeiro momento que viu a aplicación orixinal funcionando. Consta de diferentes sistemas clasificadores que non deixan de estar vixentes na actualidade, ademáis dunha representación visual bastante ben lograda e atractiva. Ademáis débese ter en conta que naquela época non existían tantas ferramentas, para facer unha aplicación de tal calibre, probablemente fose inviable chegar a bo porto empregando unicamente ferramentas de software libre. Aínda sendo moi exhaustivos na análise do software orixinal, só se lle poden por defectos na forma na que foi desenvolto e implementado, sendo o único criticable da versión orixinal.

Pódese afirmar sen rodeos e con rigurosidade, que sen esta aplicación de referencia non sería posible o presente TFG. Nexte contexto búscanse proxectos relacionados sexan PFC, TFG, TFM, teses e artigos. Non hai un sistema clasificador de estrelas automatizado como pretende ser o aquí abordado, por tanto convérteo en algo innovador con tódolas vantaxes e inconvintes que esto ten.

Se ben existen outras implementacións que tratan sobre diferentes aspectos da astrofísica, polo momento non se atoparon traballos que teñan unha funcionalidade similar ao programa orixinal. Existe algunha aplicación cun grao de similitude significativa dun PFC de hai cerca dunha década [4], dedicándose fundamentalmente a almacenar espectros nunha base de datos, para posteriormente ser utilizados con facilidade. Aínda que os obxectivos sexan relacionados con este proxecto, en canto se pasa un pouco ao detalle vese que é un sistema completamente diferente, e mesmo poderían considerarse como complementarios, pois mentres o PFC en cuestión dedícase a súa almacenaxe nunha base de datos para logo poder consultalos fácilmente por parte de astrónomos e astrofísicos, o obxectivo deste traballo é a construcción dun sistema que clasifique os espectros de forma automatizada, e sí, unha vez clasificados poderían ser almacenados nese sistema que foi obxecto dun PFC. Un TFM aparentemente algo máis relacionado [5] que o anterior, precisa dunha lectura con certo grao de detalle para saber a onde chegaron e a onde non.

As técnicas de IA empregadas serán sistemas expertos e redes neuronais de diferentes

tipos, se ben, as redes neuronais véñense utilizando dende hai tempo, o seu uso na clasificación de espectros estelares non se produce ata a versión orixinal de Starmind.

As redes neuronais xa estaban entrenadas da versión orixinal, clasificando correctamente moitos e variados espectros. Esto facilitou moito o traballo, entrenar redes neuronais é un proceso lento, e moi complexo computacionalmente, a pesar de tódalas tecnoloxías que hai hoxe en día. É importante recalcar que non houbo que facer todo, que había unha guía inicial moi boa con tódolos problemas puntuais que poido haber.

Desde o punto de vista do autor, o que se plantexa é un desafío, excepto a aplicación Starmind orixinal, non hai nada suficientemente parecido cunhos requisitos funcionais e non funcionais similares que permitan ter algunha referencia orientativa. Ademáis aquí importan moito os requisitos non funcionais, non se pode limitar a funcionalidade a que só cumpla os requisitos funcionais sen máis, senón que ten que cumplir unhos requisitos mínimos de eficiencia, é dicir, satisfacendo correctamente os non funcionais.

2.3 Tecnoloxías empregadas

Para abordar o TFG foi necesario combinar tecnoloxías moi variadas sen utilizar ninguna delas cunha complexidade excesiva. A complexidade en sí está en integrar tantas técnicas diferentes, e non en utilizar tres linguaxes de programación, redes neuronais de varios tipos e sistemas expertos. É importante distinguir entre as tecnoloxías que se utilizaron na construcción de Starmind orixinal e nas utilizadas para Starmind2, que como se explica nos seguintes parágrafos están interrelacionadas.

Na aplicación orixinal utilizouse a linguaxe C++, o entorno de desenvolvemento C++ Builder 5, redes neuronais de diferentes tipos...

Por outra banda en Starmind2 empréganse máis linguaxes e máis tecnoloxías. O feito de estar as redes neuronais xa entrenadas en C++, foi o que quitou definitivamente a idea de reimplementar a lóxica de negocio da aplicación en Java, a pesar de haber esa idea en principio. Java actualmente polos motivos que sexa, é moito máis utilizado que C++, pero o feito de estar as redes neuronais implementadas e entrenadas en C++, motivou o abandono da idea de volver a codificar a lóxica de negocio en Java. A aplicación divídese fundamentalmente en tres capas segundo o patrón arquitectónico Modelo Vista Controlador. O capa modelo segue implementada en C++, a capa controlador foi construída en C++ para este traballo e finalmente a vista foi feita en Python e Javascript. En principio ao seguir mantendo a capa modelo en C++ parecía algo arcaico pero logo demostrouse que fora un acerto, pois hai métodos que clasifican varios espectros de maneira continuada ata que finalmente dan o resultado, son bastante costosos computacionalmente e grazas a que non están en Java, xunto con que os seus bucles conteñen a maioría das súas iteraccións independentes, posibilitou a súa para-

lelización empregando openMP. Esto último en sentido estricto non é correcto actualmente, existe unha implementación de opemMP para Java http://www.omp4j.org/, pero de todas formas é algo moi novo e non se usa de maneira significativa.

Fundamentalmente empregouse C++ para as partes máis costosas computacionalmente, a parte de que as redes neuronais xa estivesen nesa linguaxe, procurouse implementar todo o posible dentro de C++, sobre todo polo tema da eficiencia. Empregouse Python por ser plenamente compatible con servidores web de calibre tal como *Apache* ou *Ninx*, ademáis de que a linguaxe Python favorece o uso de metodoloxías áxiles, xunto con que préstase moito a obligar a construcción do código de forma ordenada, pola súa peculiar forma de indicar un bloque, sendo coa tabulación a única maneira posible de facelo. Ademáis é multiparadigma, algo que axuda moito a unha aplicación deste tipo. Na capa vista utilizouse Javascript, fundamentalmente porque é unha forma moi usada para dar comportamento dinámico as páxinas web permitindo interactuar fácilmente, é dicir, entre outras características poder saber a posición no gráfico espectral ao manter o rato enriba del, poder facer zoom entre dous puntos indicados encima do gráfico...

2.3.1 Tecnoloxías e plataformas empregadas

En cantos as tecnoloxías empregadas para representar os datos ao usuario úsanse HTML 5, CSS 3. Por outra parte a lóxica de negocio será executada nun servidor *http* coma Apache. Por outra banda para a paralelización de métodos costosos computacionalmente usaranse as directivas OpenMP. Para o paso de datos entre a API feita en C++ e Python utilízase Swig, pois é o máis cómodo que se atopou que permite o paso de parámetros, aínda que non sexan tipos elementais como poden ser *list<string>*. A continuación explicaranse as características máis salientables de cada unha destas tecnoloxías.

Apache É un servidor HTTP de código aberto para plataformas Unix (BSD, GNU/Linux...), Microsoft Windows, Macintosh... que implementa o protocolo HTTP/1.1 é a noción de sitio virtual segundo a normativa RFC 2616. Cando comenzou o seu desenvolvemento en 1995 basouse no código do popular NCSA HTTPd 1.3, pero máis tarde foi reimplementado totalmente de novo. O servidor Apache é desenvolto e mantido por unha comunidade de usuarios baixo a supervisón de Apache Software Foundation dentro do proxecto HTTD Server (httpd). Este será o servidor empregado cando o software final estea listo e pase a producción.

OpenMP É unha interfaz de programación de aplicacións (API) para a programación multiproceso de memoria compartida en múltiples aplicacións. Permite engadir concorrencia aos programas escritos en C, C++ e Fortran sobre a base do modelo de execución fork-join. Basicamente é unha directiva que permite a parelelización de bucles no que os resultados dunha iteración non dependen dos resultados de anteriores, sobre arquitecturas de memoria compartida como son os típicos equipos de escritorio ou mesmo os portátiles. Esta técnica non

ten sentido utilizala de maneira illada en arquitecturas de memoria distribuída como pode ser un cluster, para iso empréganse outras técnicas como pode ser OpenMPI [6], o que si podería ser posible e razoable, é utilizar OpenMPI para repartir os tarefas que se executarán dentro de cada nodo, e dentro de cada un deles aproveitar a súa arquitectura multicore con OpenMP.

```
#pragma omp parallel for default(shared) private(j, nameStar,
    errors) schedule(dynamic, chunk)
for (j = 0 ; j < spectraList.size() ; j++) {
    #pragma omp critical
    {
        strcpy(nameStar, it->c_str());
        it++;
    }
    if (TEstrella::CreaEstrellaDeFichero(nameStar, &catalogo[j],
    errors)) {
        catalogo[j]->CalcularParametros(errors);
        #pragma omp critical
        {
            indexCatalog.push_back(j);
            i++;
        }
    }
    errors.clear();
```

Unha parte de código paralelizado 2.3.1 de Starmind2 utilizando esta directiva.

Swig O *Simplified Wrapper and Interface Generator* (SWIG) é unha ferramenta de software libre pra conectar aplicacións ou bibliotecas escritas en C/C++ con linguaxes de scripting como son Lua, Perl, PHP, Python, R, Go... A intención disto é poder usar obxectos, métodos ou mesmo funcións escritas en C/C++ por outras linguaxes de programación de scripting pasando tipos de datos complexos. O programador escribe o ficheiro de interfaces contendo unha lista de clases cos seus métodos ou funcións en C/C++ para facelas visibles ao intérprete. Swig compilará este ficheiro xerando código en C/C++ e máis na linguaxe destino. Swig xerará código para as funcións e métodos con argumentos simples, mentres que os parámetros complexos (por exemplo int *, list<string> &...) deben ser escritos polo programador. A ferramenta Swig creará o código que une C++ e a linguaxe destino. [7]

Ajax acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono e XML), é unha técnica de desenvolvemento web para crear aplicacións interactivas que se executan no cliente, é dicir, o navegador dos usuarios mentras se mantén a comunicación asíncrona en segundo plano co servidor segue respondendo aos eventos do usuario sen quedar bloqueado. Desta forma además é posible realizar cambios nas páxinas sen necesidade de recargalas, mellorando a interactividade, velocidade e usabilidade das aplicación, incluso reduce o tráfico de datos entre cliente e servidor. Unha cuestión importante é que os datos que devolve o

servidor non teñen porque ser en formato XML, poden ser en JSON ou mesmo nun formato propio definido polo programador. En Starmind2, practicamente úsase Ajax para todo menos para cando o usuario inicia sesión, péchaa ou cando sube un ou máis ficheiros. De non existir esta tecnoloxía complicaría moito a execución, pois calquera cambio que involucrase ao servizo remoto, daría lugar a recargar a páxina enteira consumindo moito ancho de banda, descedendo significativamente o rendemento da aplicación e mesmo a experiencia de uso que sería menos agradable. Na imaxe 2.11 móstrase como funciona de maneira simplificada.

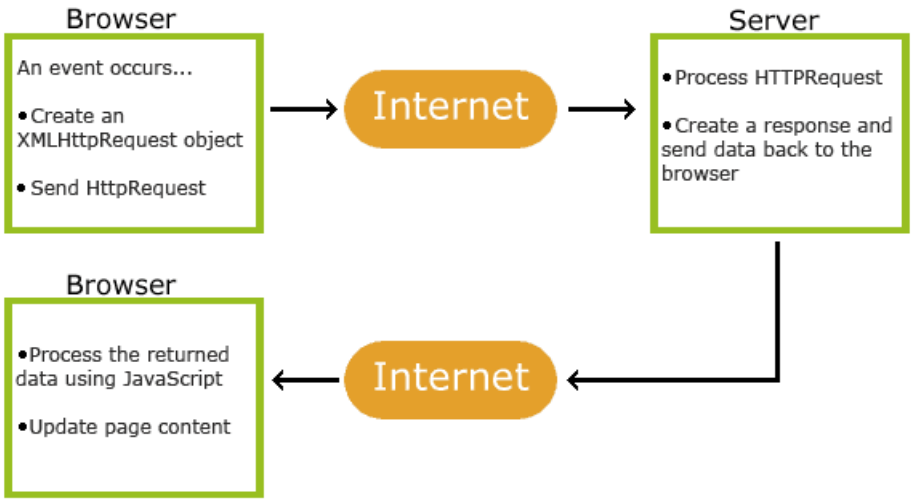


Figura 2.11: Diagrama que mostra o funcionamento de Ajax

CSS é unha linguaxe de marcado de diseño gráfico para definir e crear a representación dun documento estructurado escrito nunha linguaxe de marcado. É moi usado pra establecer o diseño visual dos documentos web, e interfaces de usuario escritas en HTML e XHTML, pode ser aplicado a cualquier documento XML, incluindo XHTML, SVG, XUL, RSS... Tamén permite aplicar follas de estilo non visuales como as follas de estilo auditivas. Xuntos con HTML e JavaScript, CSS é unha tecnoloxía usada por moitos sitios web para crear páxinas visualmente atractivas, interfaces de usuario para aplicacións web e interfaces de usuario para aplicacións móbiles. CSS está diseñado principalmente para marcar a separación do contido do documento coa forma de presentación. Isto permite a separación entre o contido do documento e a súa forma de presentación, tamén búscase mellorar a accesibilidade do documento, proveer máis flexibilidade e control na especificación das características de representación. Tamén permiten que varios documentos HTML compartan o mesmo estilo usado dunha ou varias follas de estilo CSS e reducir a complexidade e a repetición de código na estructura de documento. A separación do formato e do contigo fai posible representar o mesmo documento marcado en diferentes estilos para diferentes métodos de renderizado, como en pantalla, en impresión,

en voz (mediante un navegador de voz ou lector de pantalla, e dispositivos táctiles basados no sistema Braille). A especificación CSS describe un esquema prioritario para determinar que reglas de estilo son aplicadas cando hai máis dunha regra pra un elemento en concreto. Estas regras son aplicadas con un sistema chamado de cascada, de modo que as prioridades son calculadas e asignadas as regras, así que os resultados son totalmente deterministas.

2.3.2 Linguaxes de programación e bibliotecas empregadas

- C++
- Python3
- Javascript

A continuación exporanse brevemente as características máis salientables de cada unha delas.

C++ é unha linguaxe de programación compilada deseñada en 1979 por Bjarne Stroustrup. A intención do seu creador foi extender a linguaxe de programación C mecanismos que permitan a manipulación de obxectos. Neste sentido, dende o punto de vista das linguaxes orientadas a obxectos, C++ é unha linguaxe híbrida a diferenza de Java que é puramente orientada a obxectos. Unha peculiaridade é a posibilidade de redefinir operadores e de poder crear novos tipos que se comporten coma tipos fundamentais. Ademáis o uso da memoria é responsabilidade do programador, non hai posibilidade de ter un recolector de basura, o cal posibilita o desenvolvemento dunhas aplicacións moi eficientes desde o punto de vista computacional ao non haber recolector automático. O feito de non ter recolector de basura obliga a ser máis ordeados co código. Como é unha linguaxe fortemente tipada admite herencia múltiple. As características de poder redefinir operadores foron eliminadas en Java. [8] [9]

Python é unha linguaxe de programación interpretada ca filosofía da facer fincapé en que a sintaxe favoreza un código lexible. É unha linguaxe de programación multiparadigma, o que significa que non forza aos programadores a adoptar un estilo particular de programación, máis ben permite varios estilos; Programación orientada a obxectos, programación imperativa e programación funcional. Están soportados outros paradigmas mediante o uso de extensións. Usa tipado dinámico e conteo de referencias para a administración da memoria. Unha característica destacable é a resolución dinámica de nomes; É dicir, o que enlaza un método é o nome dunha variable durante a execución dun programa. Outro obxectivo importante é a facilidade de extensión, pódense escribir facilmente novos módulos en C/C++, en definitiva trátase dunha linguaxe multiparadigama. É unha linguaxe interpretada, usa tipado dinámico e é multiplataforma. [10]

Javascript é unha linguaxe de programación interpretada, dialecto do estándar ECMAScript. Soe ser habitual referirse a el abreviadamente como JS. Defínese coma orientado a

obxectos, basado en prototipos, imperativo, debilmente tipado e dinámico. Utilízase principalmente do lado do cliente (client-side), implementado como parte dun navegador web permitindo melloras na interfaz de usuario e páxinas web dinámicas. Deseñouse cunha sintaxe similar a C, aínda que adopta nomes e convencións de Java. Sen embargo Java e JavaScript teñen semánticas e propósitos moi diferentes. Tódolos navegadores modernos interpretan o código JS, dándolle unha implementación do Document Object Model (DOM) para poder interactuar cunha páxina web. Tradicionalmente utilizábase en páxinas web HTML para realizar operacións e unicamente no marco da aplicación cliente pero actualmente é utilizado para enviar e recibir información a algún servidor coa axuda doutras tecnoloxías como AJAX. JS interprétase no axente de usuario ao mesmo tempo que as sentenzas vanse descargando xunto o código HTML. [11]

nvd3 é unha biblioteca implementada en javascript que permite debuxar gráficos interactivos cunha calidade moi aceptable de forma moi parecida a como se faría en Matlab, é dicir, de forma xinxela sin ter que facer grandes traballos para representar funcións simples como poden ser representar espectros estelares e moitos dos seus compoñentes. Despóis de intentar primeiro usar *vis.js* para pintar os gráficos, tiña unha complexidade tan elevada para pintar unha simple función matemática dunha soa variable, que houbo que buscar outras alternativas, elixindo finalmente nvd3.

2.3.3 Ferramentas para o desenvolvemento

C++ Builder é un entorno de desenvolvemento rápido de aplicacións na linguaxe C++ para Windows que inicialmente era propiedade da empresa *Borland* e actualmente é da empresa *Embarcadero Technologies*. C++ Builder combina a biblioteca Visual Component Library (VCL) é o IDE escrito en Delphi cun compilador de C++. O seu ciclo de lanzamento é anual e inclúe ferramentas que permiten o seu desenvolvemento visual de arrastrar e soltar compoñentes sobre a aplicación. É importante destacar que ata o de agora este IDE ademáis de só executarse en Windows, só permite facer aplicacións a citada plataforma. Outro aspecto destacable é que incorpora moitos obxectos que non están nos estándares de ISO de C++, co cal é fácil que os programadores usen clases que non forman parte do estándar ISO, co cal están limitando a súa portabilidade. Esta ferramenta só se usará pra extraer a lóxica da aplicación orixinal, posibles depuracións, pois a idea é deixar de depender definitivamante dela. Aínda que se marca coma unha ferramenta de desenvolvemento é só para extraer funcionalidades e depurar código. En Starmind2 non se está utilizando propiamente coma ferramenta de desenvolvemento en sentido estricto. [12]

Django é un framework de desenvolvemento web en código aberto, escrito en Python, que respecta o patrón arquitectónico Modelo Vista Controlador (MVC), aínda que non os denomina exactamente así, senón que desta maneira: Modelo, plantilla e vista. Foi desenvolto na

súa orixe para xestionar varias páxinas orientadas a noticias de World Company de Lawrence e foi liberado ao público baixo licencia BSD en xulio do 2005. En xunio de 2008 anunciouse que a recién formada Django Software Foundation faríase cargo de Django no futuro, e así continua ata o día de hoxe. A meta principal de Django é facilita-la creación de sitios web complexos, pois pon énfase no reuso, na conectividade e a extensabilidade de componentes, no desenvolvemento rápido e o principio "Non te repitas" (DRY). Python é usado en todas as partes do framework, incluso nas configuracións, ficheiros e nos modelos de datos. Desde a propia Django Software Foundation recálcase que é para desenvolvemento e non para producción, neste caso deberá empregarse un servidor que cumpla cunhas determinadas especificacións. De non haber algo similar a Django habería que construir a aplicación en Apache directamente ou similares, con tódolos inconvintes que esto conleva; Non se pode empregar nun servidor que estea en producción, de cada vez que se fagan cambios dependendo da complexidade destes habería que reiniciar o servidor correspondente. Hoxe en día non sería tan catastrófico, pois existen máquinas virtuais que funcionan perfectamente en computadores de hoxe en día, non obstante, tería unha carga de traballo sensiblemente maior este proxecto de non existir un framework coma Django. [13] [14]

Igual resalta que non se utilizasen IDEs coma Eclipse, pero efectivamente non se usou, porque o analista e o programador non son expertos no seu uso, e polo tanto non quixeron complicarse a vida adquirindo unhos coñecementos mínimos para que o seu uso resultase cómodo e fluído. En vez de utilizar un IDE complexo úsase un editor de código coma *gedit* que resalte a linguaxe empregada sexa C++, Python ou Javascript e que permita gardar o código xerado coa codificación UTF-8. Outro factor que fixo prescindir de Eclipse foi a facilidade de compilación que da a ferramenta *make*, sen a cal o proceso de compilación sería lento e tedioso, pois conta de varios pasos intermedios.

Capítulo 3

Estudo alternativas

A hora de abordar este traballo xurdiron varias propostas sobre que linguaxes e tecnoloxías escoller entre varias existentes, tendo en conta as vantaxes e limitacións de cada unha delas. O problema non era nada sinxelo, a pesar de estar claro dende o principio que non se podía utilizar unha única linguaxe ao tratarse dunha aplicación de natureza distribuída. Ao tratarse dun programa deste tipo necesítanse dúas linguaxes polo menos, unha na parte de servidor e outra na parte do cliente, e soe ser inaudito que fosen na mesma linguaxe aínda que desde o punto de vista teórico non existan problemas para lograr algo dese estilo.

Tendo en conta os obxectivos nomeados na sección 1.3, é moi importante telos claros para recurrir a eles en calquera punto de proxecto no que haxa grandes dúbidas ou diferenzas entre os integrantes do equipo. Por suposto calquera das seguintes posibilidades está analizada tendo en conta que en calquera delas haberá como mínimo dúas capas; Por un lado a do servidor e por outro a do cliente. O feito de precisar polo menos dúas capas é algo irrenunciable, pois dende o principio se partiu de que a parte cliente será unha parte web, e por outra parte a de servidor, que será a que manexe toda a lóxica da aplicación. Por tanto o primeira decisión a que nos enfrentamos coa versión orixinal de Starmind foron as seguintes alternativas:

1. Actualizala, además de engadirlle varias funcionalidades, a unha versión actual desa ferramenta (Embarcadero Builder C/C++)

2. Implementar un servizo remoto con C++ Builder e logo facer unha capa web que utilice ese servizo

3. Reimplementala desde practicamente cero sen utilizar ferramentas propietarias

3.1 Pros e contras de ambas posibilidades

A primeira opción é a máis fácil en canto a recursos tanto de man de obra coma de tempo, sen embargo non se deixa de depender dunha ferramenta propietaria que emprega moitos

componentes que non son estándar, además de que por si soa non consegue arranxar o feito de que non estea implementada mediante o uso de capas. Además o feito de seguir sen obligar a utilizar determinados estándares non é nada beneficioso. Se nalgún momento desaparece algún compoñente propietario utilizado en Starmind2, hai que sustituílo por algún dos estándares coa cantidade de recursos que pode supoñer. Esto non é estar en contra do software propietario senón que se intenta analizar as consecuencias desde unha perspectiva o máis ampla e aberta posible. Non se propón ou prioriza o uso dunha determinada ferramenta en función de que sexa propietaria ou libre, pero si en funcións das súas características como son, curva de aprendizaxe, facilidade de uso, plataformas sobre as que corre... De por sí soa, esta opción non soluciona o problema de ter un servicio centralizado que permita que exporte as súas funcionalidades por medio dunha API.

A segunda opción xa ten algo moito máis sustentable que a primeira, en C++ Builder só se implementaría a capa servizo, construindo unha API que permita acceder a súas funcionalidades, xunto con un cliente web que se comunicaría co servizo. A idea é moito máis atractiva que a primeira, polo menos xa hai unha separación do software en capas, pero a capa servizo segue anclada a un sistema operativo Windows, además de seguir dependendo dunha ferramenta propietaria de desenvolvemento que non facilitaría unha posible parelización dalgunhas das súas funcionalidades. Con todo é unha posibilidade moito máis atractiva que a primeira, pois só segue dependendo do Borland para o servizo, aínda que parece que as vantaxes non son moitas, se se analiza con detemento compróbase que é unha grande mellora, polo menos xa nos libramos de moitos compoñentes propietarios desta ferramenta, como moito teremos compoñentes de tipo *TSocktet* e *Tthread*, co cal a porcentaxe de dependenza da ferramenta igual baixou dende un 80% ata un 30%. Además ao ter implementada a capa servizo con eles, é moito máis fácil planificar a súa migración a outras plataformas. Queda no aire o grao de aproveitamento desta solución das arquitecturas multicore, instruccións SIMD... aínda que si aproveitarían os actuais 64 bits de ancho de palabra.

A terceira opción é a máis desexable dende o punto de vista da modularidade e manteñibilidade do sistema, además de ser a que pode aproveitar actualmente e no futuro as posibles melloras no hardware e no software. Ao implementalo practicamente desde cero usaranse capas, o lóxico hoxe en día, empregando principios e patróns de diseño. Un patrón arquitectónico que se pasou por alto na versión orixinal foi o patrón MVC, unha simple visual ao código de non máis de dez minutos deixará a patente acuñada. O feito de non empregar principios e patróns de desenvolvemento parece que aforra carga de traballo, pero é só unha falsa ilusión. Non hai que olvidar que todo esto ten un alto custe de recursos, sendo o tempo o máis demandado esta opción. En resumen, ten un alto custe en recursos do proxecto, pero a cambio ten grandes vantaxes como son ter un código modular, además de que no futuro requerirá de menos recursos para manterse actualizado, engadido facilidade a hora de corri-

xir posibles erros, además de poder seguir estando plenamente operativo o producto final segundo transcurra o tempo e se actualice o sistema operativo, as bibliotecas, as ferramentas de compilación... Estas melloras son debidas ao cumprimento de certos estándares que non van desaparecer así por así, ao non depender de nada propietario senón de organizacións coma ANSI, IEEE... Aínda que sexa a versión que aparentemente teña máis probabilidades de dar máis fallos e dificultades ao principio, o cal é discutible, garantiza que os posibles fallos e actualizacións serán moito máis adsequibles que coas opcións 1 e 2.

O feito de escoller a terceira opción sabendo de antemán que a carga de traballo sería bastante maior non estivo exenta de polémicas decisións. A aplicación orixinal estaba implementada básicamente en C++ utilizando dialectos de Borland C++ Builder que non eran problema en sí, pero había que ter en conta que ao non empregar C++ estándar pois habería algún que outro problema co que non se contaba inicialmente, xa que hai moitos obxectos ou compoñentes que non forman parte do estándar C++. A idea de construir un servicio web en C++ era moi acolledora, pero a hora de mirar servicios remotos implementados nesta linguaxe chegouse a conclusión de que non era algo fácilmente asumible. Por outra banda estaba a tentadora idea de reimplementar o código en Java, hoxe en día existen multitude de servizos remotos implementados nesta linguaxe, pero dado o feito de que C++ é unha linguaxe híbrida a diferenza de Java que é puramente orientado a obxectos, co además de que integra un sistema experto clasificador, xunto con redes neuronais xa entrenadas, pois fixo botar por terra a idea de reimplementala en Java, además se temos en conta que nesta última linguaxe non se dispón de structs tan empregadas aquí. Outro dos aspectos polo que nos decantamos en manter a capa modelo en C++ é a facilidade de parelelizar diferentes rexións do código utilizando OpenMP, algo do que se empregásemos Java non deixaría de ser posible con tecnoloxías como omp4j [15] que son aínda moi novas, estando case en fase experimental . Finalmente a capa modelo foi reimplementada en C++ polas razóns aquí expostas, que C++ sea memos utilizado que Java non quere dicir que non sexa útil para unha aplicación de ámbito científico. Elixindo Java non sería ningún problema o sistema operativo onde se vaia executar o servidor, pero ao utilizar C++ estándar non hai problema en atopar un compilador de certa calidade que se execute sobre diferentes sistemas operativos como Windows, linux, FreeBSD, MacOS...

Se se analizan ben as diferenzas entre Java e C++, obsérvase de que a principal diferenza desde o punto de vista do programador entre ambas linguaxes é o manexo da memoria e máis as excepcións. Se ben en C++ o programador debe administrar a memoria, non hai recolector de basura, e neste último non hai diferenza entre excepcións comprobadas (nas que o programador está obrigado a capturalas) e non comprobadas, sendo todas deste tipo en C++, non hai grandes diferenzas en canto ao deseño e implementación dunha aplicación en C++ ou en Java. Aínda que C++ parece que ten inconvintes moi salientables non son tales, que o uso da memoria sexa responsabilidade do programador tamén lle da a oportunidade

de deseñar un código moi eficiente, o das excepcións non é tal inconvinte, onde se saiba que se poden producir métense dentro dun *try* co seu corresponte *catch*, además de que existe un xenérico que captura calquera excepción que poida ocurrir dentro do programa. Outro dos aspectos discutidos de C++ é a posibilidade de empregar herencia múltiple, que pode complicar o código, ben é certo que pode crear complicacións innecesarias ao igual que en Java se temos moitas interfaces. Un código pode ser complicado ou máis sinxelo sin que influia en gran medida a linguaxe na que estea codificado, é dicir, a linguaxe claro que afecta ao rendemento da execución, pero a calidade do código segue influíndo independentemente da linguaxe utilizada. Polo momento con Cuda podemos parelizar código tanto en C++ como en Java, pero está moito máis explotado para C++.

Un dos aspectos polos que se tardou en decidir foi que servidor web empregar, había varias opcións moi interesantes, por unha banda incrustar un pequeno servidor web coma https://www.gnu.org/software/libmicrohttpd/ e por outra utilizar un entorno de desenvolvemento coma Django que logo permite que o servizo en producción sexa prestado por servidores de calibre coma Apache, Nginx... Tengo en conta estas dúas opcións baralladas, decantámonos pola segunda, porque aínda que a primeira idea é moi interesante inicialmente, existía o temor de como escalaría o servidor http libmicrohttpd, co agravante de que as comprobacións dos tipos de datos nos parámetros terían que ser feitas practicamente desde cero (implicaría máis tempo de desenvolvemento e probablemente algún que outro retraso), además coa vantaxe de que ao ir implementada en Python, facilita moito o paso de datos entre a capa de visualización e a capa controlador. A complicación maior desta opción foi o paso de determinados parámetros como son arrais de enteiros e flotantes, lista de strings ou mesmos strings, pero unha vez solucionados este problemas démonos conta de que foi unha boa decisión manter o núcleo en C++. Además Django é un servidor http bastante lixeiro pra desenvolvemento e probas, non require de moita capacidade de procesamento nin moita memoria, a súa curva de aprendizaxe é bastante sinxela ao empregar unha linguaxe coma Python. Además facilita moito o uso de bases de datos sinxelas sen andar creando unha clase por cada entidade existente na base de datos. Django además ocupa un lugar privilexiado nas ferramentas que se soen utilizar nas metodoloxías áxiles.

O que pode resultar algo compricado é o paso de Django a Apache, pero tamén é certo que existe moita documentación ao respecto, o que fará que o proceso de migración non sexa algo de ir descubrindo todo paso a paso desde o principio, senón que consistirá en seguir as guías e solucionar algún que outro problema que poida haber. A idea de deixar Django como servidor de producción é totalmente descartable polos seguintes motivos:

- A propia documentación oficial de Django indícao moi claro, só é para desenvolvemento, nunca para producción [13]
- Django non permite as conexións *https* que é un dos obxectivos fundamentais do pro-

xecto

- Tampouco está moi documentado sobre cantos usuarios simultáneos podería manexar dando unha mínima garantía de servizo

3.2 Alternativas elexidas

En resumen, polas motivos discutidos e analizados no presente capítulo, tómanse a seguintes decisións:

1. Implementarase desde practicamente cero a capa modelo en C++, aproveitando as redes neuronais xa entrenadas, o sistema experto e obtendo a lóxica de negocio da versión orixinal
2. Utilizarase o patrón MVC
3. Utilizaranse ferramentas libres que cumpran certos estándares
4. Na parte do servidor utilizaremos Python pola súa compatibilidade coa maioría dos servidores http
5. Como servidor http empregarmeos Apache, pois é un servidor con ampla presencia en internet, ademáis dunha certa fiabilidade dada por moitos anos de uso
6. Empregarase Python3 e non a versión 2, porque esta última ten data de fin de soporte
7. A parte de servidor será compilada con *g++* coa plena axuda da ferramenta *make*
8. Empregarase *Boostrap* para construir unha interface de usuario visualmente atractiva, ademáis de reducir parte de código Javacript, pois para tarefas que se repiten no tempo e en diferentes páxinas boostrap xa o ten previsto, dando moitas funcionalidades válidas por defecto.
9. Para a codificación de calquera parte de Starmind empregarase o editor de código *gedit*, pois é lixeiro, resalta ben as tres linguaxes empregadas e permite a codificación UTF-8

Unha vez elexidas as diferentes alternativas, xa está estamos en condicións de continuar o proxecto, é aínda que esta fase non sexa estrictamente moi laboriosa, sí que ten unha enorme responsabilidade, pois as decisións aquí tomadas afectarán non só durante o ciclo de desenvolvemento, senón que tamén afectarán ao ciclo de vida do producto, de forma que tomar ben o camiño entre as diferentes alternativas é algo crucial.

Capítulo 4

Planificación e custes

Antes de pasar aos detalles en si do proxecto Starmind2, é útil dar unhas definicións breves e concisas de certos aspectos da xestión de proxectos, pois o autor pensa que poden facilitar a comprensión do resto do capítulo. Pódese definir un proxecto coma unha pauta para desenvolver un producto final, mentres que un proceso é unha pauta a seguir para desenvolver un proxecto ou mesmo un conxunto de pasos para lograr un determinado fin. Outra definición de proxecto pode ser a de unha actividade humana que transforma os requisitos dun cliente ou usuario nun producto software entregable. Un proxecto está formado por un conxunto de actividades relacionadas para lograr un fin específico, con un comezo e fin claros, suxeito a tres restriccións importantes: Tempo, presuposto e alcance. En calquera tipo de proxecto independentemente do seu tamaño haberá que lidiar con estes tres factores. Para obter un producto final, que cumpla as espectativas para as que foi concebido, é necesario mellorar a xestión do proxecto optimizando a previsión de control dos seguintes factores:

- **Esforzo**: É algo que afecta directamente ao equipo de desenvolvemento.
- **Tempo**: Afecta tanto ao equipo de desenvolvemento coma ao usuario final.
- **Custe**: Debe estar compensado co producto resultante.

Pódese pensar en deixar a responsabilidade de que o proxecto cumpla as espectativas satisfactoriamente sen máis que en aumentar a calidade das probas. Pero o problema é que as probas non soen atopar máis dun 50% dos erros existentes [16], o cal deixa claro, que non é suficiente con só mellorar a calidade das probas. A solución máis eficaz é empregar procesos de calidade en tódalas fases da que se compón, e mesmo a de utilizar as probas en practicamente tódalas fases das que se compón. O tema das probas tratarase en detalle no tema seguinte.

Este capítulo irá dividido nas seguintes seccións: Planificación estimada e real, na que se falará das diferenzas entre a planificación e o que sucedeu realmente, o diagrama de Gantt, qué

é un clásico na xestión de proxectos, o cal é unha ferramenta gráfica que ten como obxectivo indicar o tempo de dedicación previsto para as diferentes tarefas ou actividades ao largo dun tempo total determinado, sen indicar as relacións existentes entre actividades, por último o cálculo de custes, algo totalmente esencial e necesario, se non se controlan os custes é algo similar a afirmar que non se controla un proxecto.

4.1 Planificación estimada

A planificación estimada sempre se debe facer sen ser excesivamente optimistas nin excesivamente pesimistas, hai que intentar ser o máis realistas posibles intentado ver os parecidos con proxectos que teñan unha certa similitude ou sexan dun ámbito similar. Pero neste caso é moi difícil encontrar algo similar (según o indicado na sección 2.2) é a experiencia na dirección de proxectos reais desta envergadura é prácticamente inexistente. Con todo as tarefas, planificadas da mellor forma posible polo autor, son as mostradas na imaxe 4.3.

Con todo a liña base está pensada desde unha forma bastante razoable, cóntase que poida haber algún problema, pero que non van alargar demasiado o camiño crítico. Con respecto ao camiño crítico, pódese afirmar que tódalas tarefas son críticas, pois as seguintes dependen das anteriores, un retraso nunha supón o retraso no comezo da seguinte, co cal irá propagando sucesivamente o retraso ata chegar ao final. Esto é debido fundamentalmente a que todo o proxecto nunha grande parte foi levado a cabo pola mesma persoa facendo os papeis de analista e programador, eso sí, contando coa axuda dos titores, que fan o rol de xefe de proxecto, en caso de serios problemas ou adversidades. Como se verá na seguinte sección, é imposible prever tódolos contratempos que poida haber, nin aínda que foramos excesivamente pesimistas poderiamos chegar a prevelos, así que non queda outra que facer unha planificación usando criterios de racionalidade, xunto co estudiado na asignatura xestión de proxectos, sentido común e a pouca experiencia que poidamos ter dentro do ámbito do desenvolvemento do software. Ante a dúbida e a inexperiencia debe usarse a bibliografía, se esto non chega, o sentido común, habitualmente permítenos saír satisfactoriamente de diversos incidentes e atrancos.

Segundo as directrices do anterior parágrafo foi establecida a seguinte liña base 4.4.

Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesora
	Starmind2	**105 días**	**mié 02/01/19**	**mié 29/05/19**	
	Implementación dun servizo remoto	**36 días**	**mié 02/01/19**	**mié 20/02/19**	
	Comprobación do funcionamento da versión orixinal	10 días	mié 02/01/19	mar 15/01/19	
	Extracción da lóxica de negocio	4 días	mié 16/01/19	lun 21/01/19	3
	Deseño das funcionalidades	10 días	mar 22/01/19	lun 04/02/19	4
	Implementación das funcionalidades	12 días	mar 05/02/19	mié 20/02/19	5
	Proba das funcionalidades do servizo remoto	**29 días**	**jue 21/02/19**	**mar 02/04/19**	
	Planificación das probas	12 días	jue 21/02/19	vie 08/03/19	2
	Execución das probas	2 días	lun 11/03/19	mar 12/03/19	8
	Conclusións das probas	15 días	mié 13/03/19	mar 02/04/19	9
	Implementación da capa web	**11 días**	**mié 03/04/19**	**mié 17/04/19**	**7**
	Implementación representación gráfica	7 días	mié 03/04/19	jue 11/04/19	
	Redeseño da interfaz usuario	4 días	vie 12/04/19	mié 17/04/19	12
	Verificación, validación e posta en marcha de tódolos módulos	**22 días**	**jue 18/04/19**	**lun 20/05/19**	**11**
	Conexión dos diferentes módulos	10 días	jue 18/04/19	mié 01/05/19	
	Probas de integración	12 días	jue 02/05/19	lun 20/05/19	15
	Paralelización	**7 días**	**mar 21/05/19**	**mié 29/05/19**	**14**
	Implementación do paralelismo	3 días	mar 21/05/19	jue 23/05/19	
	Probas dos métodos paralelos	4 días	vie 24/05/19	mié 29/05/19	18
	Elaboración dunha documentación técnica	105 días	mié 02/01/19	mié 29/05/19	2CC;17FF

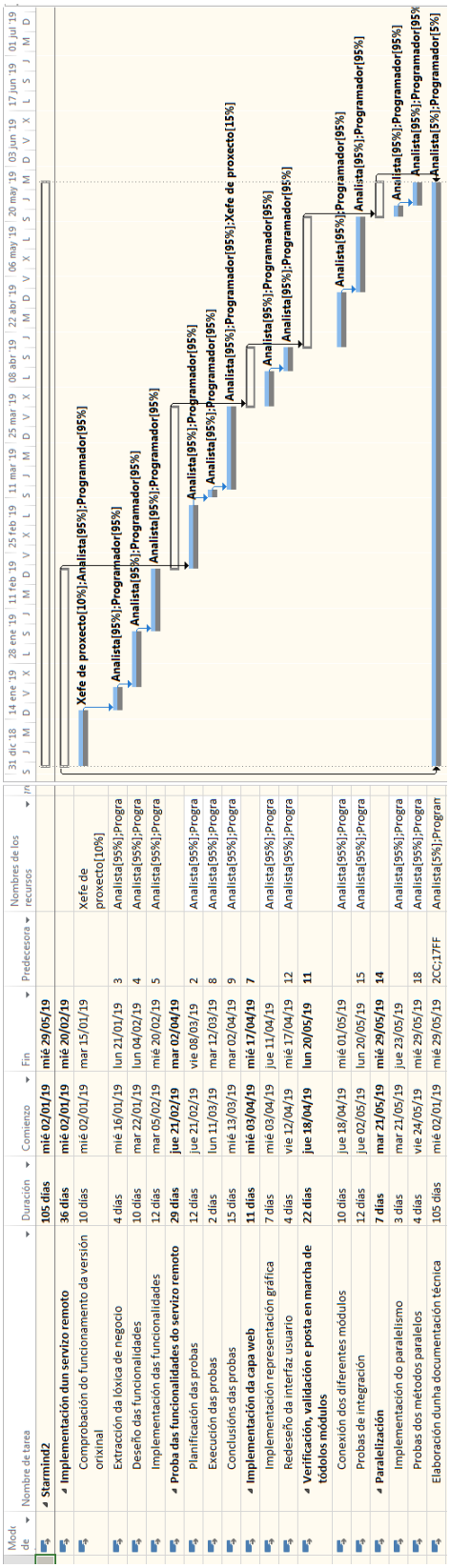
Figura 4.1: Lista de tarefas sen asignar recursos

Nombre del recurso	Tipo	Etiqueta de	Iniciales	Capacidad máxima	Tasa estándar	Tasa horas extra	Costo/U:	Acumu	Calendario base
Xefe de proxecto	Trabajo		X	100%	27,46 €/hora	27,46 €/hora	0,00 €	Prorrateo	Estándar
Analista	Trabajo		A	100%	26,94 €/hora	26,94 €/hora	0,00 €	Prorrateo	Estándar
Programador	Trabajo		P	100%	19,30 €/hora	19,30 €/hora	0,00 €	Prorrateo	Estándar

Figura 4.2: Lista de recursos coa súa capacidade e custe

Nombre de tarea	Duración	Comienzo	Fin	Predece	Nombres de los recursos
Starmind2	**105 días**	**mié 02/01/19**	**mié 29/05/19**		
Implementación dun servizo remoto	**36 días**	**mié 02/01/19**	**mié 20/02/19**		
Comprobación do funcionamento da versión orixinal	10 días	mié 02/01/19	mar 15/01/19		Xefe de proxecto;Analista;Program
Extracción da lóxica de negocio	4 días	mié 16/01/19	lun 21/01/19	3	Analista;Programador
Deseño das funcionalidades	10 días	mar 22/01/19	lun 04/02/19	4	Analista;Programador
Implementación das funcionalidades	12 días	mar 05/02/19	mié 20/02/19	5	Analista;Programador
Proba das funcionalidades do servizo remoto	**29 días**	**jue 21/02/19**	**mar 02/04/19**		
Planificación das probas	12 días	jue 21/02/19	vie 08/03/19	2	Analista;Programador
Execución das probas	2 días	lun 11/03/19	mar 12/03/19	8	Analista;Programador
Conclusións das probas	15 días	mié 13/03/19	mar 02/04/19	9	Analista;Programador
Implementación da capa web	**11 días**	**mié 03/04/19**	**mié 17/04/19**	**7**	
Implementación representación gráfica	7 días	mié 03/04/19	jue 11/04/19		Programador
Redeseño da interfaz usuario	4 días	vie 12/04/19	mié 17/04/19	12	Programador
Verificación, validación e posta en marcha de tódolos módulos	**22 días**	**jue 18/04/19**	**lun 20/05/19**	**11**	
Conexión dos diferentes módulos	10 días	jue 18/04/19	mié 01/05/19		Analista;Programador
Probas de integración	12 días	jue 02/05/19	lun 20/05/19	15	Analista;Programador
Paralelización	**7 días**	**mar 21/05/19**	**mié 29/05/19**	**14**	
Implementación do paralelismo	3 días	mar 21/05/19	jue 23/05/19		Analista;Programador
Probas dos métodos paralelos	4 días	vie 24/05/19	mié 29/05/19	18	Analista;Programador
Elaboración dunha documentación técnica	105 días	mié 02/01/19	mié 29/05/19	2CC;17FF	Analista;Programador

Figura 4.3: Lista de tarefas cos seus recursos



Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesora	Nombres de los recursos
	Starmind2	**105 días**	**mié 02/01/19**	**mié 29/05/19**		
	Implementación dun servizo remoto	**36 días**	**mié 02/01/19**	**mié 20/02/19**		
	Comprobación do funcionamento da versión orixinal	10 días	mié 02/01/19	mar 15/01/19		Xefe de proxecto[10%]
	Extracción da lóxica de negocio	4 días	mié 16/01/19	lun 21/01/19	3	Analista[95%];Progra
	Deseño das funcionalidades	10 días	mar 22/01/19	lun 04/02/19	4	Analista[95%];Progra
	Implementación das funcionalidades	12 días	mar 05/02/19	mié 20/02/19	5	Analista[95%];Progra
	Proba das funcionalidades do servizo remoto	**29 días**	**jue 21/02/19**	**mar 02/04/19**		
	Planificación das probas	12 días	jue 21/02/19	vie 08/03/19	2	Analista[95%];Progra
	Execución das probas	2 días	lun 11/03/19	mar 12/03/19	8	Analista[95%];Progra
	Conclusións das probas	15 días	mié 13/03/19	mar 02/04/19	9	Analista[95%];Progra
	Implementación da capa web	**11 días**	**mié 03/04/19**	**mié 17/04/19**	**7**	
	Implementación representación gráfica	7 días	mié 03/04/19	jue 11/04/19		Analista[95%];Progra
	Redeseño da interfaz usuario	4 días	vie 12/04/19	mié 17/04/19	12	Analista[95%];Progra
	Verificación, validación e posta en marcha de tódolos módulos	**22 días**	**jue 18/04/19**	**lun 20/05/19**	**11**	
	Conexión dos diferentes módulos	10 días	jue 18/04/19	mié 01/05/19		Analista[95%];Progra
	Probas de integración	12 días	jue 02/05/19	lun 20/05/19	15	Analista[95%];Progra
	Paralelización	**7 días**	**mar 21/05/19**	**mié 29/05/19**	**14**	
	Implementación do paralelismo	3 días	mar 21/05/19	jue 23/05/19		Analista[95%];Progra
	Probas dos métodos paralelos	4 días	vie 24/05/19	mié 29/05/19	18	Analista[95%];Progra
	Elaboración dunha documentación técnica	105 días	mié 02/01/19	mié 29/05/19	2CC;17FF	Analista[5%];Program

Figura 4.4: Liña base xunto co diagrama de Gantt

4.2 Seguimento do proxecto

Entrando con certo nivel de detalle nos dous seguimentos efectuados, empézanse a ver as diferenzas entre a planificación estimada e o trancurso real do proxecto.

4.2.1 Seguimento ao 50%

No seguimento efectuado ao 50% mostrado na figura 4.5, o retraso na tarefa *Comprobación do funcionamento da versión orixinal* 4.5 é debido a que o xefe do proxecto, que é a persoa que máis sabe do tema pois implementou a versión orixinal de partida, non estaba dispoñible por estar formando parte doutros proxectos relacionados co ámbito da astronomía. Esto aínda que pareza raro retrasou o comezo do proxecto, pois sen unha versión de referencia que funcionase, era impensable empezar por algún lado que non fose a documentación. As tarefas afectadas son todas as seguintes, pois todas forman parte do camiño crítico.

Se non fose suficiente isto, na tarefa *Conclusión das probas* 4.5 comezou cun retraso de dous días motivado polas diferenzas entre os membros do equipo. Os integrantes non acababan de encaixar as grandes diferenzas de valores usando o mesmo código fonte inicial das probas de Starmind2, segundo se empregase o compilador *g++* ou o compilador *bcc32.exe*. As diferenzas non se podían pasar por alto, excepto pechando os ollos. Case puxeron en perigo a continuidade do proxecto Starmind2. Finalmente, con moitas dificultades e reticencias, chegou a haber un acordo que permitiu saír da situación de estancamento.

4.2.2 Seguimento ao 100%

No seguimento final do proxecto 4.6 mostrado na figura 4.6 obsérvase que fundamentalmente que en vez de acabar na data 29/05/2019, xa se intuía un retraso co seguimento ao 50%, acabou o 17/06/2019, motivado a posteriores retrasos que aconteceron despóis do paso de ecuador de Starmind2. Na tarefa *Probas de integración* os resultados que daba, unha vez integrados tódolos compoñentes, a versión final, mostraban unhos valores completamente diferentes aos dados pola versión orixinal de Starmind. Esto era algo herdado da tarefa anterior *Conclusión das probas* na que a solución parcial tomada polos membros do equipo foi que se ignorasen, o que non era máis que un parche no seu momento. Chegados a este a punto, non se podían parchear as diferenzas existentes, había que tomar unha decisión, pois ou se solucionaban esas diferenzas, ou era a fin de Starmind2. A sitúción era crítica, chegou a haber sete días de retraso, moito desánimo por parte de algún membro do equipo, pero finalmente grazas a unha idea feliz dun deles, conseguiron solucionarse esas diferenzas de valores. A este acordo, que foi coma un fito, tamén axudou que houbese un artigo cunhos valores de referencia, de forma que lle diron unha confianza ao equipo, que ata o de agora faltaba. Con esta solución as diferencias eran plenamente aceptables por parte de tódolos membros.

Con respecto ao tema da paralelización, convén destacar que non ocasionou retrasos, en gran parte porque era a última fase do proxecto que precisaba que tódalas demáis xa estivesen finalizadas. O feito de que o resto das tarefas xa estivesen listas, xunto coa habilidade e a boa disposición dos integrantes analista e programador en temas de paralelización a baixo nivel (sen chegar a paralelismo a nivel de instrucción) propiciou que non se alargasen máis do previsto, a pesar de ser algo delicado, pois houbo que repensar grandes bucles con moitas variables e mesmo protexer algunhas delas de accesos concurrentes. A pesar de que non houbo retrasos nesta tarefa, supuxo un esforzo extra do analista e máis do programador, pois aínda que están acostumados a esas técnicas, aquí aparecen un montón de variables de tipo obxecto, que aínda que en esencia haxa que protexer da mesma maneira que se fai con calquera tipo de variable compartida, sexa con semáforos ou mútex, non é exactamente o mesmo, pois a tecnoloxía empregada está basicamente pensada e deseñada para paralelizar bucles con iteracións independentes, a diferenza da creación de procesos ou threads, o cal é algo totalmente diferente.

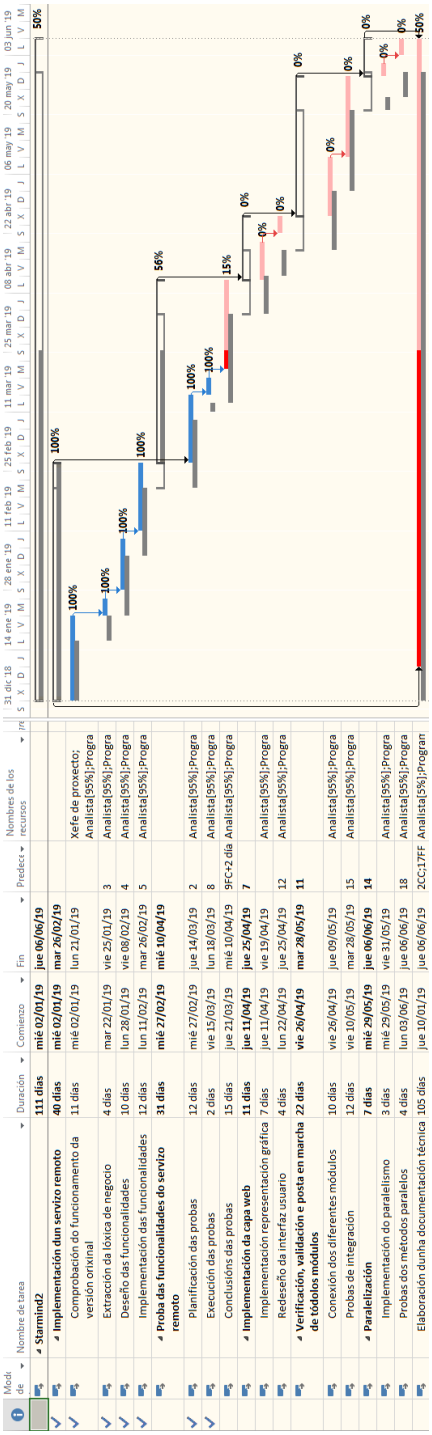


Figura 4.5: Seguimento ao 50%

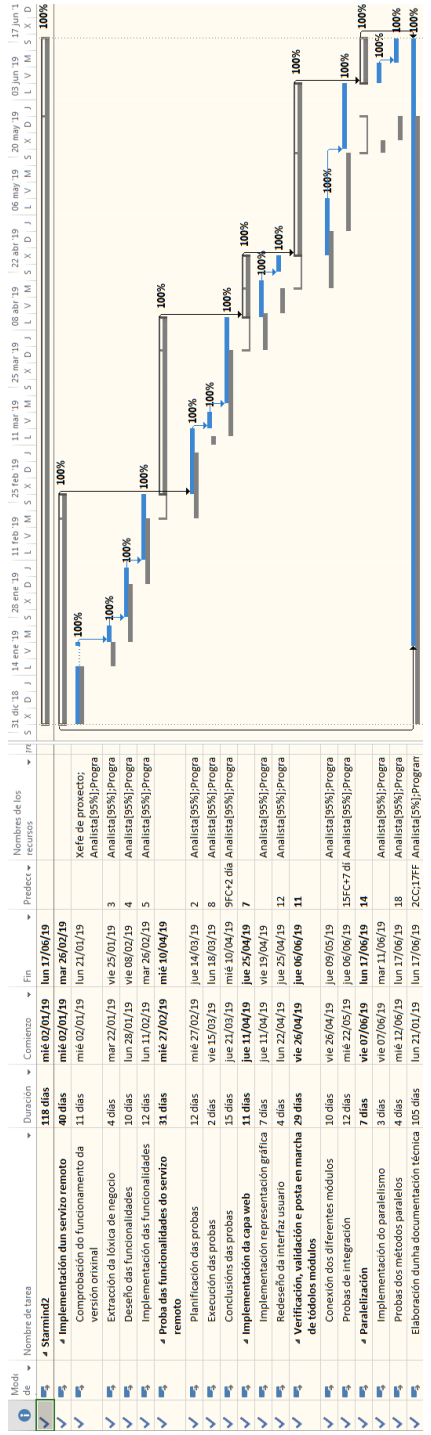


Figura 4.6: Seguimento ao 100%

4.3 Cálculo de custes

Pra o cálculo de custes hai que ter en conta que a plantilla está composta dun xefe de proxecto, que case nunca interven, un analista e máis un programador.

Un lugar oficial de onde quitar os prezos de man de obra, mostrados na táboa 4.1, dos tres integrantes do equipo foi en https://www.boe.es/eli/es/res/2009/03/18/(6), pero dado que aos membros do equipo nos parecía que debia ser comparado con respecto ao que cobraría unha empresa de software por levar a cabo Starmind2, decídese por convenio aplicarlle o prezo do BOE 2.5 veces.

O tema dos prezos, que nos parecen disparados hacia baixo, ten a súa loxica se se pensa detidamente; Realmente é o salario que recibe o traballador, pero non se ten en conta a parte que a empresa ten que pagar ao Estado, podendo ser normalmente algo máis do que recibe o traballador. Por esta razón poremos o prezo da man de obra 2.5 veces o salario indicado no BOE. Outro aspecto importante é que se houbese que empregar horas extras, que só se utilizasen como último recurso, pois aplicariáselles o mesmo prezo por hora que as ordinarias. A correspondencia creada polo equipo de Starmind2 entre os cargos do BOE e máis os cargos de Starmind2 é a mostrada na táboa 4.2. Facendo unhos cálculos elementais supoñendo que se traballa 8 horas ao día, que un ano ten aproximadamente 250 días laborables, sae o precio da hora de man de obra definido en 4.3. É importante recalcar que os proxectos nunca se planifican con horas extra, nin se fai neste caso, pois sería algo contraproducente e contrario a unha boa planificación, as horas extras sempre deben ser para imprevistos puntuais, alguén que está de baixa, florecemento de requisitos, unha determinada biblioteca que ía facer grande parte da carga de traballo e resulta que o funcionamento non é o esperado e non se atopou ningunha que faga o que se esperaba que fixera a anterior... Debe quedar moi claro, que nunca se debe planificar algo con horas extras.

As estatísticas do Microsoft Project dan o total de horas traballadas por cada un dos recursos existentes, co cal con operacións elementais podemos saber con bastante aproximación o custe real do proxecto, tendo xa unha cifra coa que se poden comparar outros proxectos. A seguinte táboa 4.2 mostra os recursos tal e como os manexa MS Project.

Segundo as estatísticas do sistema empregado para o control do proxecto, o custe total de Starmind2 é de segundo mostra o resume 4.7. Observando o custe dado polo Project, que é de 39.555,56 €, débese ter en consta que estrictamente só reflexa o gasto en persoal e nada máis. Por este motivo, débese ter en costa que o custe real, é sensiblemente maior, pois non estamos tendo en conta o prezo do uso dos equipos, xa sexa por aluguer ou en propiedade, o consumo eléctrico das máquinas computacionais, a iluminación da oficina, os folios que se gastaron, boligros, lápices, tinta de impresora... Nin siquiera temos en conta o prezo do único software non libre que empregamos, que é MS Project.

Categorías	Salarios mensual (14 pagas)	Salario anual
Analista e Analista de sistemas	1.569,25 €	21.969,50 €
Analista-programador/a e Deseñador/a de páxina Web	1.539,69 €	21.555,66 €
Programador/a senior, Xefe/a de operación e Programador/a en Internet	1.103,04 €	15.442,56 €

Táboa 4.1: Táboa de salarios según o BOE

Categoría BOE	Categoría Starmind2
Analista e Analista de sistemas	Xefe de proxecto
Analista-programador/a e Deseñador/a de páxina Web	Analista
Programador/a senior, Xefe/a de operación e Programador/a en Internet	Programador

Táboa 4.2: Táboa de equivalencia entre o BOE e Starmind2

Cargo Starmind	Precio/hora BOE	Precio/hora Starmind2
Xefe de proxecto	10,98 €	27,46 €
Analista	10,78 €	26,94 €
Programador	7,72 €	19,30 €

Táboa 4.3: Prezo da hora de man de obra do equipo Starmind2

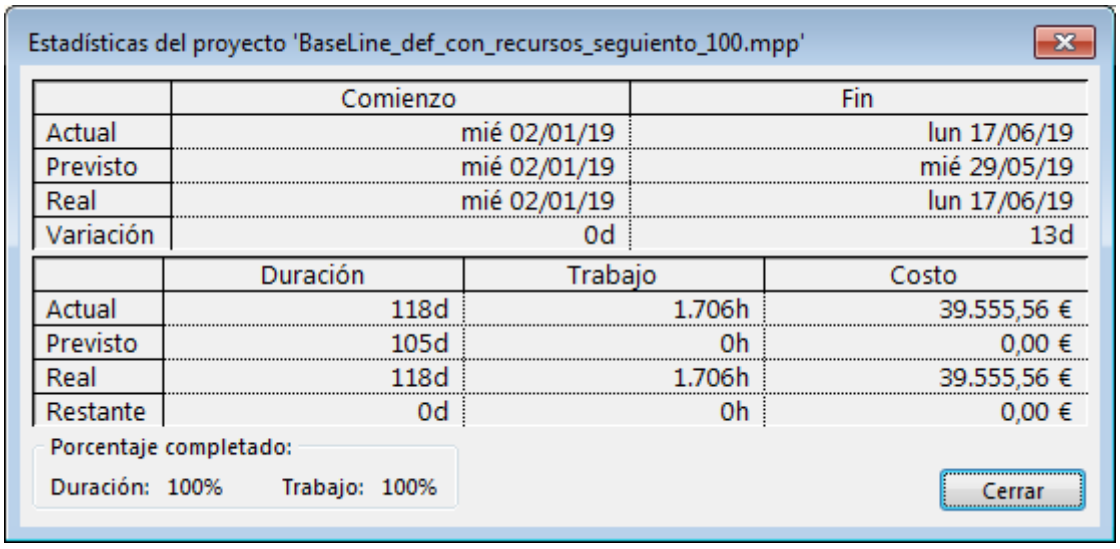


Figura 4.7: Recursos cos seus custes

Coas súas pequenas inexactitudes o autor pensa que é unha aproximación razoable o custe real do proxecto, tendo en conta que é moi difícil obter o custe real cunha exactitude total, evidentemente sen ter en conta a pérdida de precisión obtida por pasar as cantidades monetarias a IEEE-754, que é insignificante para algo monetario coma esto.

Tampoco están desglosados os gastos a maiores que causaron as incidencias nas tarefas *Conclusións das probas* e *Probas de integración*, posto que vendo o diagrama de seguimento ao 100% obsérvase que con unhos pequenos cálculos obtense o desfase existente entre o custe real e o previsto.

Finalmente convén recalcar que se deberían ter en conta os riscos sempre que e cando se fai unha planificación, pero dado que é un traballo de fin de grao, xunto coa inexperiencia en proxectos que teñan algunha relación coa investigación, non fai posible facer unha análise de riscos antes de estar plenamente metidos nas tarefas máis complicadas, con todo o que iso poida significar. Evidentemente o risco máis grande aconteceu na tarefa *Probas de integración*, a cal puxo en dúbida a continuidade de Starmind2.

Capítulo 5

Metodoloxía

UNHA metodoloxía de desenvolvemento de software refírese ao marco, entorno de traballo ou *Sprint* que se usa para estructurar, planificar e controlar o proceso de desenvolvemento en sistemas de información [17]. Segundo pasa o tempo foron desenvoltos unha gran cantidade de métodos diferenciándose entre sí polas súas fortalezas e debilidades. O marco de desenvolvemento para unha metodoloxía de desenvolvemento de software consiste en:

- Unha filosofía de desenvolvemento de aplicacións con enfoque no proceso de desenvolvemento de software.
- Ferramentas, modelos e métodos para axudar ao proceso de desenvolvemento do software.

No caso de Starmind emprégase a metodoloxía *Scrum*, a cal ten as seguintes características:

- É un método para traballar en equipo a partir de iteracións ou Sprints. Así pois, Scrum é unha metodoloxía áxil, polo que o seu obxectivo será controlar e planificar proxectos cun gran volume de cambios de última hora, onde o grado de incerteza é moi elevado.
- O habitual é facer a planificación por semanas. Ao final de cada *Sprint* ou iteración, revísase o traballo validado da semana anterior. En función do seu estado, priorízanse e planifícanse as actividades nas que se invertirán os recursos no seguinte *Sprint*.
- A metodoloxía Scrum céntrase en axustar os resultados e responder as esixencias, reais e exactas do cliente. Desta forma vaise revisando cada entregable, posto que os requisitos van variando a corto plazo. O tempo mínimo para un Sprint é dunha semana e como máximo é de catro semanas.

Entre as principais características da metodoloxía Scrum, destaca que é un desenvolvemento incremental en lugar da clásica planificación dun proxecto completo. Os equipos

de traballo caracterízanse por ser auto organizativos, centrándose na calidade do producto final.

O desenvolvemento dun producto ten un ciclo de vida na metodoloxía *Scrum*, sendo as fases nos que se divide un proceso Scrum as seguintes:

- Que e quen? O producto que se quere obter unha vez finalizado o Sprint, e os roles de equipo coas súas tarefas asignadas
- Onde e cando? Especifica o plazo e o contido do Sprint
- Por que e como? As distintas ferramentas para aplicar nesta metodoloxía áxil

Cada Sprint ou iteración ten unha serie de eventos ou etapas como se mostran na seguinte lista.

1. Reunión para a planificación do Sprint, na que se decide a duración del, o obxectivo a alcanzar e o entregable do mesmo.
2. Traballo de desenvolvemento durante o Sprint no que nos aseguramos que se cumplen os obxectivos, que non se producen cambios que alteren o seu obxectivo, e que se mantén un contacto constante co cliente ou dono do producto.
3. Revisión do Sprint na que se mantén unha reunión co cliente na que se revisa o producto do Sprint. Defínense os aspectos a cambiar, se fose necesario, de maior importancia para planificalos no seguinte Sprint.
4. Retrospectiva do proxecto na que o equipo de desenvolvemento ten a oportunidade de mellorar o seu proceso de traballo e aplicar os cambios nos seguintes Sprints.

A metodoloxía *Scrum* ten unhos roles e responsabilidades principais. Os roles asignados aos seus procesos de desenvolvemento son os seguintes.

- Project Owner é responsable de que o proxecto se estea desenrolando acorde coa estratexia de negocio. Escribe historias de usuario, priorízaas e colócaas no Product Backlog.
- Master Scrum ou Facilitador é responsable de eliminar os obstáculos que impiden que o equipo cumpla co seu obxectivo.
- Development team Member son os engargados de crear o producto para que poida estar listo cos requerimentos necesarios. Debe ser un equipo multidisciplinar de non máis de dez persoas, aínda que existen excepcións a esta norma.

El *Product Backlog* ou *lista de producto* é un listado de tódalas tarefas que se pretenden facer durante o desenvolvemento dun proxecto. Algunos *product backlog* poden asociarse con proxectos de varios anos. Tódalas tarefas deben listarse aquí, para que estean visibles para todo o equipo podendo manter unha visión panorámica de todo o que se espera realizar.

5.1 Metodología empregada coas súas correspondentes iteracións

Utilízase a metodoloxía *Scrum* explicada no parágrafo anterior. Os roles existentes no proxecto Starmind2 serán os seguintes: **Project Owner**, serán os titores, pois eles coñecen moi ben a aplicación de partida que deu lugar a este traballo, además de estar moi formados e postos ao día en temas de estudio de obxectos celestes, tamén xogarán o rol de **Master Scrum**, pois son os encargados de solucionar os obstáculos que dificultan a consecución do obxectivo final, mesmo tomando decisións ou mesmo eliminado algunha funcionalidade. Finalmente o autor deste TFG será **Development Team Member**, quen as veces fai unha portentaxe moi pequena de Master Scrum, pero só en casos moi puntuais. As diferentes iteracións son citadas na seguinte lista.

1. Implementación dun servizo remoto
2. Probas das funcionalidades do servizo remoto
3. Implementación da capa web
4. Verificación, validación e posta en marcha de tódolos módulos
5. Paralelización

Cada unha das seguintes iteracións será detallada nos seguintes parágrafos. Algunha delas dividiranse en varias, pois algunhas delas, son moi laboriosas como para cumplirse nunha unidade, algo así como cando en programación estructurada existe un problema grande a forma de resolvelo e a descomposición en problemas máis pequenos, que sexan capaz de darlle unha solución acertada ao problema de partida. Tamén é importante indicar que en cada un destes Sprints elabórase documentación técnica de utilidade para todo o equipo de desenvolvemento.

Implentación dun servizo remoto. Esta iteración é o punto de partida, dalgunha forma hai que extraer a lóxica de negocio da aplicación orixinal. A súa vez está composta de varias subiteracións, pois é algo bastante extenso e laborioso. O primeiro paso será conseguir que funcione, dalgunha forma o Development Team Member tivo que recurrir ao Master Scrum. Unha vez posta a funcionar xa se poido extraer a lóxica de negocio con algunha dificultade pois estaba orixinalmente todo xunto, non había separación en capas, e o coñecemento do autor en análise de espectros é moi limitado. Nas distintas reunións para a planificación deste Sprint quedou ben claro que era a primeira pedra pra construir Starmind2, defínese unha API a partir da cal o modelo poida exportar as súas funcionalidades a outras capas remotas. Unha vez definida unha API, e vendo que representaba ben as funcionalidades do servizo, pasouse a súa implementación, que non foi tan trivial como estaba previsto polo tema do paso de variables entre as linguaxes Python e C++. *Swig* facilita moito o paso de variables entre estas dúas linguaxes permitindo pasar parámetros que non son elementais.

Proba das funcionalidades do servizo remoto Finalizado o Sprint anterior, había que pasar as probas, inicialmente pensouse que podería ir dentro da iteración anterior, pero dada a característica crítica destas probas para a continuidade decidiuse que iría nunha iteración aparte. Houbo que planificalas moi ben, pois os seus resultados debían ser moi similares aos do software orixinal. Foi moi laborioso por diversos motivos, entre eles o feito de que os valores pra debuxar as gráficas na versión orixinal eran sintéticos, é dicir, eran creados a partir dos reais procesándoos para podelos por unha gráfica, pois naquela época non sobraban compoñentes para pintar gráficas como pasa hoxe en día; Xunto con que pequenas diferenzas de valores daban lugar a dúbidas sobre se se estaba facendo mal ou non. A execución das probas en sí mesma foi practicamente sen ningún esforzo, pois estaban moi ben planificadas, agora ben, a parte da súa interpretación foi máis tediosa do esperado debido a pequenas diferencias que despistaban ao Development Team Member e coas cales non estaban plenamente de acordo Project Owner e máis Master Scrum.

Implementación da capa web Esta iteración podería chegar a ser independente das outras, pois realmente non precisa de ningunha das demáis, pois podía ser construída e probada cunha capa ficticia feita exclusivamente para iso. Pero o feito de só haber un Development Team Member no equipo de traballo motivou que fose posta como unha tarefa que dependerá do fin de *Proba das funcionalidades de servizo remoto*. Estaba moi claro que había que pintar as gráficas dunha forma similar ou mellorada respecto a orixinal, mantendo unha interface similar a orixinal de aspecto agradable e funcional. O tema de non reformar moito a interface é para que os astrónomos e astrofísicos afeitos a usar a aplicación orixinal non teñan un período de adaptación demasiado grande, e que poidan pasar da orixinal a Starmind2 sen practicamente notar diferenza, salvo nas melloras.

Verificación, validación e posta en marcha de tódolos módulos Aquí ven a parte máis complicada de todo o desenvolvemento, xúntanse o servizo remoto, a capa de acceso ao servizo remoto coa parte web. Esta iteración dada a súa complexidade e diferenzas entre os membros do equipo está composta de varias subiteracións. En principio as integracións non soen ser complicadas, neste caso a conexión entre eles foi fluída, pero as probas de integración foron outro tema. O que complicou un pouco máis do habitual esta iteración foia a aparición dunhas diferenzas de valores en varios resultados, sendo finalmente arranxadas. Analizando detidamente o código apareceron as causas que propiciaban tales diferenzas. Unha vez que foron solucionadas pasouse a seguinte iteración.

Paralelización Esta iteración foi deixada para o final sobre todo polo tema do tempo de entrega, e incluso se barallou non implementala, pois o software podería funcionar sen esta característica, tal e como o leva facendo ata o de agora, e podería ser entregada nunha próxima iteración. Os membros do equipo tiñan moi claro que mentres non estiveran completadas as demáis iteracións non se podía empezar con esta, pois o feito de paralelizar algo soe dar pro-

blemas relacionados coa concorrencia. Con todo isto en conta, finalmente, o equipo optou por intentar engadir esa mellora, que en sí costou o seu esforzo, pero tamén é verdade que vendo os resultados, tivo unha compensación moi boa dando un rendemento bastante significativo.

Un dos aspectos moi bos entre outros desta metodoloxía é que sempre queda a porta aberta a posibles melloras e optimizacións en posteriores iteracións. Aquí non existe a última iteración, non habendo nada nesta metodoloxía que indique dalguna maneira cal é a derradeira. Así Starmind2 nunca estará exento de posibles melloras ou novas funcionalidades.

5.2 Análise do sistema

5.2.1 Definición dos actores do sistema

Para o análise do sistema utilízase fundamentalmente UML. Un paso básico é identificar os actores, que son fundamentalmente entidades que sen formar parte do sistema interactúan con el. Basicamente existen dous tipos de actores: **Administrador** e **Astrofísico**. A asociación existente entre un actor e un caso de uso indica que este actor pósee a capacidade de interactuar co sistema da maneira indicada polo caso de uso [18] [19]

O **administrador** é quen pode administrar usuarios. Este tipo de actor só poderá administrar usuarios como tal, se necesita usar o sistema para o que foi deseñado, precisa crear un usuario de tipo astrofísico. O **astrofísico** é un usuario logueado (necesita autentificarse para acceder ao sistema) que pode utilizar Starmind2 coas funcionalidades propias para o que foi creado. Probablemente nas próximas iteracións de Starmind2 o administrador poida facer máis tarefas que as actuais, pero simplemente non houbo tempo para pensar que máis precisaría facer un administrador. Por eso quedámonos simplemente coas básicas: Administrar usuarios, algo totalmente necesario tendo en conta a natureza da aplicación.

5.2.2 Casos de uso

Os casos de uso describen en forma de lista de accións e de interacións o comportamento do sistema, estudiado desde o punto de vista dos actores. Definen os límites do sistema e as súas relacións co entorno. Entre un actor e o sistema, os casos de uso describen as accións e interacións vinculadas cun obxectivo funcional do actor. Os casos de uso detallan os requisitos funcionais do sistema relativos a algunhos obxectivos dalgún autor [18].

Especificación dos casos de uso

Aquí definiranse os casos de uso que describen as funcionalidades básicas, non todos nin moito menos, pois sería unha lista enorme, ademáis de que non é necesario especificalos todos, pois as similitudes entre algunhos casos de uso son triviais.

Autenticar un usuario	
Actores	Un usuario administrador ou experto en astrofísica.
Descripción	Inicia sesión un usuario do sistema
Precondicións	O usuario debe existir previamente no sistema
Postcondicións	Se o usuario é experto en astrofísica o sistema mostra a pantalla inicial do sistema listo para o seu uso; Se o usuario é administrador lévao a pantalla específica de administración

Táboa 5.1: Logueo dun usuario

Cambiar contrasinal de calquera usuario	
Actores	Un usuario administrador
Descripción	Cambia o contrasinal de calquera outro usuario
Precondicións	O usuario debe estar previamente autentificado
Postcondicións	O usuario ao que un administrador lle acaba de cambiar o contrasinal teno recién cambiado. A próxima vez que se identifique terá que empregar o novo contrasinal

Táboa 5.2: Cambio de contrasinal de calquera usuario

Crear usuario	
Actores	Un usuario administrador
Descripción	Crea un usuario de calquera tipo sexa astrofísico ou administrador
Precondicións	O usuario debe estar previamente autentificado
Postcondicións	Un novo usuario experto en astrofísica ou administrador foi creado.

Táboa 5.3: Crea un usuario

Pechar sesión	
Actores	Un usuario experto en astrofísica ou administrador
Descripción	Pecha sesión o usuario
Precondicións	O usuario debe ter unha sesión iniciada
Postcondicións	O sistema volve a pantalla de inicio, se quere volver a entrar no sistema ten que identificarse

Táboa 5.4: Peche de sesión dun usuario

Cargar un ficheiro de espectro	
Actores	Un usuario experto en astrofísica
Descripción	Carga un ficheiro de espectro
Precondicións	O usuario debe estar previamente autentificado
Postcondicións	Carga os datos iniciais dese espectro amosando a gráfica normalizada entre 0 e 1

Táboa 5.5: Carga dun ficheiro de espectro

Visualizar catálogo	
Actores	Un usuario experto en astrofísica
Descripción	Amosa un catálogo
Precondicións	O usuario ten un espectro cargado no sistema
Postcondicións	O sistema amosa un espectro catálogo xunto co espectro inicial que estaba cargado antes de invocar este caso de uso

Táboa 5.6: Visualización dun catálogo

Eliminar catálogo	
Actores	Un usuario experto en astrofísica
Descripción	Deixa de amosar un catálogo
Precondicións	O usuario ten un espectro e un catálogo cargados no sistema
Postcondicións	O sistema deixa de amosar o catálogo

Táboa 5.7: Eliminación dun catálogo

Visualizar liñas de Hidróxeno	
Actores	Un usuario experto en astrofísica
Descripción	Amosa as liñas de Hidróxeno
Precondicións	O usuario ten un espectro cargado no sistema e non está mostrada a función distribución sigma 5.22
Postcondicións	O sistema amosa as liñas de Hidróxeno se non estaban mostradas antes deste caso de uso, e deixaas de amosar no caso de que xa estiveran mostradas antes de invocarse este caso de uso

Táboa 5.8: Visualización das liñas de Hidróxeno

Visualizar liñas de Helio	
Actores	Un usuario experto en astrofísica
Descripción	Amosa as liñas de Helio
Precondicións	O usuario ten un espectro cargado no sistema e non está amosada a función distribución sigma 5.22
Postcondicións	O sistema amosa as liñas de Helio se non estaban mostradas antes deste caso de uso, e deixaas de amosar no caso de que xa estiveran mostradas antes de invocarse este caso de uso

Táboa 5.9: Visualización das liñas de Helio

Visualizar liñas de metalicidade	
Actores	Un usuario experto en astrofísica
Descripción	Amosa as liñas de metalicidade
Precondicións	O usuario ten un espectro cargado no sistema e non está mostrada a función distribución sigma 5.22
Postcondicións	O sistema amosa as liñas de metalicidade se non estaban mostradas antes deste caso de uso, e deixaas de amosar no caso de que xa estiveran mostradas antes de invocarse este caso de uso

Táboa 5.10: Visualización das liñas de metalicidade

Visualizar bandas principais	
Actores	Un usuario experto en astrofísica
Descripción	Amosa as bandas principais
Precondicións	O usuario ten un espectro cargado no sistema e non está mostrada a función distribución sigma 5.22
Postcondicións	O sistema amosa as bandas principais se non estaban mostradas antes deste caso de uso, e deixaas de amosar no caso de que xa estiveran mostradas antes de invocarse este caso de uso

Táboa 5.11: Visualización das bandas principais

Visualizar bandas secundarias	
Actores	Un usuario experto en astrofísica
Descripción	Amosa as bandas secundarias
Precondicións	O usuario ten un espectro cargado no sistema e non está mostrada a función distribución sigma 5.22
Postcondicións	O sistema amosa as bandas secundarias se non estaban mostradas antes deste caso de uso, e deixaas de amosar no caso de que xa estiveran mostradas antes de invocarse este caso de uso

Táboa 5.12: Visualización das bandas secundarias

Eliminar liñas de H, He, Metalicidade, Bandas principais ou secundarias	
Actores	Un usuario experto en astrofísica
Descripción	Elimina as liñas de Hidróxeno, Helio, Metalicidade, Bandas principais e secundarias
Precondicións	O usuario ten un espectro cargado no sistema e non está mostrada a función distribución sigma 5.22
Postcondicións	O sistema elimina as liñas de Hidróxeno, Helio, Metalicidade, Bandas principais e secundarias

Táboa 5.13: Eliminación das liñas de Hidróxeno, Helio, Metalicidade, Bandas primarias e secundarias

Aplicar zoom	
Actores	Un usuario experto en astrofísica
Descripción	Fai zoom entre as lonxitudes de onda inicial e final indicadas por tecrado ou ben co ratón
Precondicións	O usuario ten un espectro cargado no sistema
Postcondicións	O sistema amosa o espectro, e outros parámetros se xa estivesen visualizados antes de invocar este caso de uso, entre as lonxitudes de onda indicadas excepto que se saian de rango, ou que a diferenza entre elas sexa menor que catro veces a resolución. Neste dous últimos casos non se cambiará o gráfico quedando tal e como estaba

Táboa 5.14: Aplicación do zoom

Salir do zoom	
Actores	Un usuario experto en astrofísica
Descripción	Sae do zoom e amosa o espectro completo
Precondicións	O usuario sexa de tipo astrófisico ten un espectro cargado no sistema
Postcondicións	O sistema amosa o gráfico completo, e outros parámetros que xa estivesen seleccionados antes de invocar este caso de uso, do espectro que ten cargado

Táboa 5.15: Finalización do zoom

Desprazamento do gráfico	
Actores	Un usuario experto en astrofísica
Descripción	Despraza o gráfico hacia a dereita ou esquerda mantendo constante a diferenza das lonxitudes de onda inicial e final
Precondicións	O usuario ten un espectro cargado no sistema e aplicoulle previamente o zoom
Postcondicións	O sistema amosa un novo gráfico, e outros parámetros que xa estivesen seleccionados antes de invocar este caso de uso, desprazado a dereita ou a esquerda segundo indicase o usuario. Se o gráfico desplazado pasa dos límites inferior e superior do espectro séguese amosando o mesmo gráfico que antes de invocar este caso de uso

Táboa 5.16: Desprazamento do gráfico

Os seguintes cinco seguintes casos de uso explican as función estatísticas. É importante indicar que as funcións estatísticas modifican as marxes do gráfico, pois dada a súa natureza non tratan certos valores do inicio nin do final. No caso de que estea activado o zoom e o gráfico mostrado esté suficientemente alonxado dos límites do espectro, non se actualizarán as marxes.

Visualizar media	
Actores	Un usuario experto en astrofísica
Descripción	Amosa ou deixa de amosar respectivamente a función de media
Precondicións	O usuario ten un espectro cargado no sistema e non está mostrada a función distribución sigma 5.22
Postcondicións	O sistema amosa a función de media, se non estaba mostrada antes deste caso de uso, e deixaa de amosar no caso de que xa estivera mostrada antes de producirse este caso de uso. Actualiza as marxes do gráfico en caso de que non se aplicase o zoom e máis non estivese mostrada ningunha destas funcións antes de invocar este caso de uso.

Táboa 5.17: Visualización da media

Visualizar sigma	
Actores	Un usuario experto en astrofísica
Descripción	Amosa ou deixa de amosar respectivamente a función de sigma
Precondicións	O usuario ten un espectro cargado no sistema e non está mostrada a función distribución sigma 5.22
Postcondicións	O sistema amosa a función de sigma, se non estaba mostrada antes deste caso de uso, e deixaa de amosar no caso de que xa estivera mostrada antes de producirse este caso de uso. Actualiza as marxes do gráfico en caso de que non se aplicase o zoom e máis non estivese mostrada ningunha destas funcións antes de invocar este caso de uso.

Táboa 5.18: Visualización da sigma

Visualizar derivada	
Actores	Un usuario experto en astrofísica
Descripción	Amosa ou deixa de amosar respectivamente a función de derivada
Precondicións	O usuario ten un espectro cargado no sistema e non está mostrada a función distribución sigma 5.22
Postcondicións	O sistema amosa a función de derivada, se non estaba mostrada antes deste caso de uso, e deixaa de amosar no caso de que xa estivera mostrada antes de producirse este caso de uso. Actualiza as marxes do gráfico en caso de que non se aplicase o zoom e máis non estivese mostrada ningunha destas funcións antes de invocar este caso de uso.

Táboa 5.19: Visualización da derivada

Visualizar continuo	
Actores	Un usuario experto en astrofísica
Descripción	Amosa ou deixa de amosar respectivamente a función de valor continuo
Precondicións	O usuario ten un espectro cargado no sistema e non está mostrada a función distribución sigma 5.22
Postcondicións	O sistema amosa a función de valor continuo, se non estaba mostrada antes deste caso de uso, e deixaa de amosar no caso de que xa estivera mostrada antes de producirse este caso de uso. Actualiza as marxes do gráfico en caso de que non se aplicase o zoom e máis non estivese mostrada ningunha destas funcións antes de invocar este caso de uso.

Táboa 5.20: Visualización do valor continuo

Eliminar media, sigma, derivada e continuo	
Actores	Un usuario experto en astrofísica
Descripción	Deixa de amosar as funcións estatísticas media, sigma, derivada e continuo
Precondicións	O usuario ten un espectro cargado no sistema e non está mostrada a función distribución sigma 5.22
Postcondicións	O sistema deixa de amosar as función estatísticas media, sigma, derivada e continuo, actualizando as marxes do gráfico no caso de que non estivese activo o zoom e estivese previamente activada algunha destas funcións estatísticas

Táboa 5.21: Eliminación das funcións estatísticas media, sigma, derivada e continuo

Visualizar distribución sigma	
Actores	Un usuario experto en astrofísica
Descripción	Amosa a sigma ordenada de menor a maior
Precondicións	O usuario sexa ten un espectro cargado no sistema
Postcondicións	O sistema amosa as sigmas ordeadas de menor a maior en caso de que se estea mostrando un espectro, en caso de que se estivese mostrando a distribución sigma volve a mostrar o gráfico espectral que había antes invocar este caso de uso

Táboa 5.22: Visualización da distribución sigma

Analizar espectro	
Actores	Un usuario experto en astrofísica
Descripción	Analiza as bandas principais, secundarias, liñas de Hidróxeno-Helio, liñas de metalicidade e a enerxía dun espectro estelar segundo as opcións que escolle un usuario.
Precondicións	O usuario ten un espectro cargado no sistema e escolle polo menos unha das cinco opcións: Bandas principais, bandas secundarias, liñas metálicas, liñas de Hidróxeno-Helio e enerxía.
Postcondicións	O sistema amosa a análise segundo as opcións escollidas

Táboa 5.23: Análise dun espectro

Analizar espectro coa nova versión	
Actores	Un usuario experto en astrofísica
Descripción	Amosa diversos parémetros dun espectro.
Precondicións	O usuario ten un espectro cargado no sistema
Postcondicións	O sistema amosa os resultados da análise mostrando diversos parámetros de bandas e liñas

Táboa 5.24: Análise nova versión dun espectro

Clasificar no sistema M-K con distintos tipos de redes neuronais	
Actores	Un usuario experto en astrofísica
Descripción	Amosa a clasificación segundo o sistema Morgan-Keenan empregando un tipo de redes neuronais indicado polo usuario
Precondicións	O usuario ten un espectro cargado no sistema
Postcondicións	O sistema amosa a clasificación Morgan-Keenan do espectro do espectro cargado previamente

Táboa 5.25: Clasificación con redes neuronais dun espectro no sistema M-K

Clasificar según luminosidade con redes neuronais	
Actores	Un usuario experto en astrofísica
Descripción	Amosa a clasificación según a luminosidade empregando redes neuronais
Precondicións	O usuario ten un espectro cargado no sistema
Postcondicións	O sistema amosa a clasificación lumínica do espectro cargado previamente

Táboa 5.26: Clasificación con redes neuronais da luminosidade dun espectro

Clasificar espectro nova versión (Global T)	
Actores	Un usuario experto en astrofísica
Descripción	Amosa a clasificación segundo o sistema Global T
Precondicións	O usuario ten un espectro cargado no sistema
Postcondicións	O sistema amosa os resultados da clasificación segundo o sistema Global T do espectro cargado previamente

Táboa 5.27: Clasificación dun espectro nova versión (Global T)

Analizar múltiples espectros nova versión	
Actores	Un usuario experto en astrofísica
Descripción	Amosa parámetros de bandas, liñas e tasas de un ou máis espectros
Precondicións	O usuario está logueado no sistema, debe proveerlle ao sistema un ficheiro de clasificación válido e un ou máis ficheiros de espectros
Postcondicións	O sistema amosa os parámetros de bandas, liñas e tasas dos espectros indicados polo usuario

Táboa 5.28: Análise de liñas, bandas e tasas de múltiples espectros

Clasificar múltiples espectros nova versión (Global T)	
Actores	Un usuario experto en astrofísica
Descripción	Amosa a clasificación segundo o sistema Global T de un ou máis espectros
Precondicións	O usuario está logueado e debe proveerlle ao sistema un ou máis ficheiros de espectros
Postcondicións	O sistema amosa os resultados da clasificación segundo o sistema Global T dos espectros indicados polo usuario

Táboa 5.29: Clasificación de múltiples espectros nova version (Global T)

Clasificar múltiples espectros no sistema M-K con distintos tipos de redes neuronais	
Actores	Un usuario experto en astrofísica
Descripción	Amosa a clasificación segundo o sistema Morgan-Keenan empregando un tipo de redes neuronais indicado polo usuario
Precondicións	O usuario está logueado e debe proveerlle ao sistema un ou máis ficheiros de espectros
Postcondicións	O sistema amosa a clasificación Morgan-Keenan dos espectros indicados polo usuario

Táboa 5.30: Clasificación múltiple con redes neuronais dun espectro no sistema M-K

Clasificar múltiples espectros según luminosidade con redes neuronais	
Actores	Un usuario experto en astrofísica
Descripción	Amosa a clasificación de un ou máis espectros segundo a luminosidade empregando redes neuronais
Precondicións	O usuario está logueado e debe proveerlle ao sistema un ou máis ficheiros de espectros
Postcondicións	O sistema amosa a clasificación lumínica dos espectros indicados polo usuario

Táboa 5.31: Clasificación múltiple con redes neuronais da luminosidade dun espectro

Practicamennte van explicados tódolos casos de uso, aínda que algunhos deles teñen en conta o estado anterior do sistema, está suficientemente indicado nas postcondicións que o estado final tamén dependerá do estado anterior, tal é como son os casos de uso Visualizar lineas H 5.8, He, 5.9, Metalicidade 5.9, bandas principais 5.11 e bandas secundarias 5.12. Tamén se da a mesma situación nas funcións estatísticas de Visualizar Media 5.17, Sigma 5.18, Derivada 5.19 e Valor Continuo 5.20.

Nos casos de uso nos que se prové algún ficheiro excepto cargar Espectro 5.5 e cargar Catálogo 5.6 para que se cumplan as precondicións é importante aclarar, por se houbese algunha dúbida, que de haber algún espectro cargado no sistema, non modifica para nada o seu estado, simplemente se limitan a mostrar os resultados do caso de uso en cuestión sen alterar nada máis.

5.2.3 Diagramas UML

As clases empregadas son *Handler*, *TEstrella* e *TLinea*. *Handler* foi implementada a propósito para Starmind2 fundamentalmente debido a que na versión orixinal non había separación entre capas, todo facía de todo, e por outra banda era necesario a creación dunha nova clase que fixera de controlador, aparte de permitir exportar os seus métodos de forma remota. A seguinte imaxe mostra un diagrama de UML moi simplicado, que permite intuír como é realmente o diagrama de clases e mesmo como interactúan entre sí.

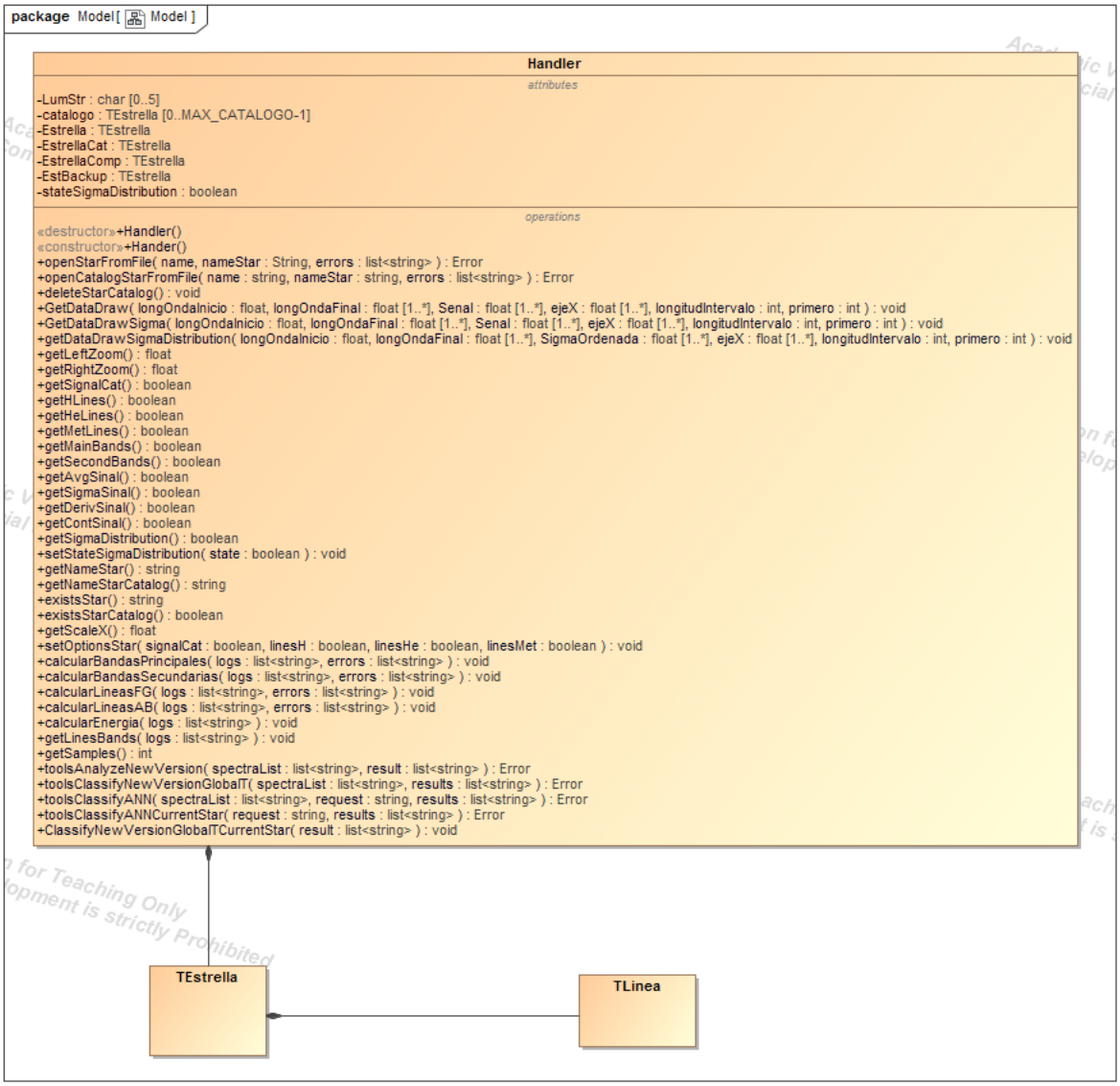


Figura 5.1: Diagrama UML moi simplificado

Fundamentalmente existen tres clases, podería interpretarse que existe algunha máis, pero como as clases que manexan a toda a lóxica de negocio *Handler*, *TEstrella* e *TLinea* están implementadas en C++, linguaxe na que se dispon de estructuras de datos como *struct* algo parecido a un rexistro de Pascal, non se puxeron máis clases no diagrama UML. Estas estructuras de datos *struct* permiten almacenar datos estructurados, sendo moi útiles na parte de C++ de Starmind. Se fose en Java serían necesarias novas clases, pois non existen nesa linguaxe [18].

É importante recalcar que só se citan os atributos é métodos máis significativos, se listásemos todos sería algo inasumible para isto, además de que está todo no código fonte; Esto só é unha axuda para comprender como está desenrolado o modelo. Nas clases *TEstrella* e *TLinea* non se indican métodos nin atributos porque o autor deste traballo entende que non aportan nada para a compresión da lóxica de negocio utilizada, son clases que se modificaron algo para Starmind2 para desfacerlles o seu gran acoplamento que tiñan coa interface gráfica e con C++ Builder. Sen desfacer este grande acoplamento sería imposible construir Starmind2, do mesmo xeito *Handler* foi diseñado e implementado para eliminar calquera acoplamento existente coa interface de usuario, de forma que incluso podería usarse no futuro por unha capa vista que non fose web.

Con este deseño cada clase está máis cohesionada, reducindo o acoplamento entre elas ao mínimo. De non ser por esto non sería posible a paralelización de diferentes métodos que son moi pesados computacionalmente.

En canto aos métodos paralelizados da clase *Handler* son: *toolsAnalyzeNewVersion, toolsClassifyNewVersionGlobalT* e *toolsClassifyANN*. Poden parecer poucos pero son os máis pesados computacionalmente, así como do resto das funcionalidades, pódese decir que funcionan sen espera por parte do usuario, non se pode afirmar o mesmo destes tres en cuestión. A parelelización coa tecnoloxía aberta opemMP consegue en gran medida baixar significativamente o tempo de espera do usuario, además de permitir facer un mellor aproveitamento do hardware, o cal é un dos obxectivos principais deste TFG. Por outra parte aínda que só se utiliza unha tecnoloxía de paralelización, o feito de empregar capas, deixa aberta a posibilidade de empregar outras técnicas máis adiante, tanto de baixo nivel como de máis alto nivel, como pode ser *Spark*.

5.3 Diferentes tipos de probas efectuadas

As probas son unha parte fundamental en calquera desenvolvemento tanto en software coma noutros desenvolvementos. Dentro dos diferentes proxectos software, a magnitude e a complexidade existentes en Starmind2, fan moi necesario un gran número de probas previamente planificadas, sendo moi variadas e bastante estrictas. Pra calquera tipo de proba dinámica antes da súa execución haberá que ter o valor esperado, e unha vez executada haberá que compara-los resultados. Se son significativamente diferentes atoparase un ou máis erros, que haberá que analizar ata atopar a súa causa, para posteriormente subsanalos. Starmind2 está en tres linguaxes diferentes, o cal da unha idea do rigurosos que se debe ser coas probas. O sistema de referencia inicial pra as probas é Starmind orixinal, os resultados dados deben ser tomados coma unha referencia para o novo sistema a probar.

As probas empregadas para obter este producto son todas de tipo dinámico, é dicir, execútase código en todas elas. Existen outro tipo de probas, que son as estáticas, pero neste caso non se empregan ao considerar que non son moi útiles sobre todo porque a maioría do código ven doutra aplicación, ademáis tendo en conta que non se implementa desce cero. Outro aspecto importante polo que non se empregan as probas estáticas e porque un dos obxectivos delas é detectar posibles erros antes de que se produzan, aquí como a mesma persoa fai de analista e de programador, xa procura ter en conta isto, polo tanto parece que non está xustificado o seu uso.

O tema dos valores en punto flotante IEEE754 de 32 e 64 bit respectivamente dou moitos problemas, en parte son debidos a que hai tipos de datos que ocupan máis bits dos esperados, e por outra parte operacións elementais coma a división dan resultados bastante diferentes, segundo empregamos un compilador Borlanc C++ Compiler ou un da familia gcc. Ao cambiar entre as diferentes versións de Borland C++ Compiler non hai diferenzas significativas entre resultados, pero entre estes últimos e gcc hai unhas diferenzas moi grandes, é o peor é que son dificilmente explicables. Polo tanto unha parte significativa das probas irán encamiñadas a descartar estes fallos, pois non son asumibles nin nas etapas intermedias do producto. Este tipo de erros, se non se corrixen, darán lugar a que non funcione o sistema experto clasificador.

Ao estar formado por capas tan diferentes, en varias linguaxes, en concreto tres, precisa de varias e diferentes probas para comprobar que todo funciona bastante ben. Son necesarias para probar cada módulo por independente e logo probar o sistema en conxunto facendo as probas de integración.

5.3.1 Probas de unidade

En canto as probas de unidade probarase de forma aillada o módulo *Handler*, que o fai de controlador interactuando coa lóxica de negocio formada polos módulos *TEstrella* e *TLinea*.

Aplícanse as distintas técnicas:

1. **Partición de equivalencia**.
2. **Análisis de valores límite/frontera**
3. **Táboas de decisión**
4. **Transición de estados**

Cada técnica aplicarase segundo xurdan as necesidades, é dicir algunhas son axeitadas para unhos casos mentres que outras son axeitadas pra outros casos.

Partición de equivalencia

Os seus obxectivos son dividir a entrada dun módulo en clases de datos nos que se poida derivar casos de proba, definir casos de proba de descubran casos de erros reducindo así o número de casos de proba a desenvolver.

	Partición 1	Partición 2	Partición 3	Partición 4	Partición 5
Entradas	Fichero valeiro	Fichero de texto con só caracteres non numéricos	Ficheiro de texto con carácteres non numéricos e números	Ficheiro binario	Ficheiro correcto
Saídas	Sistema non cambia	Sistema non cambia	Sistema non cambia	Sistema non cambia	Sistema carga espectro e mostra o seu gráfico

Táboa 5.32: Particións de proba

A única partición válida é a *Partición 5* como se ve na táboa 5.32. As outras catro particións son útiles porque proban a confiabilidade do módulo *Handler*, se algunha desas particións deixase o sistema nun estado inconsistente habería que arranxalo. Unha vez deseñadas e executadas comprobouse que o resultado esperado é totalmente compatible co obtido.

Análises de valores límite/frontera

Técnica de deseño de casos de proba que complementa a anterior. As diferenzas entre ambos son:

1. Partición de equivalenza: Selecciona un elemento representativo da clase de equivalenza
2. Análise dos valores límite: Selecciona un ou máis elementos de maneira que os límites de cada clase de equivalenza son obxecto dunha proba.

Nesta caso fanse as probas consistentes nos casos fronteira entre as clases de equivalenza non válidas. Non ten sentido probar casos fronteira entre casos válidos pois xa che fixeron moitas probas con ficheiros con espectros válidos.

As probas concretas a realizar neste caso serán:

1. Ficheiro correcto menos a última liña que terá outros carácteres non numéricos
2. Fichero con só a primeira liña correcta e o resto incorrectas
3. Ficheiro coa primeira incorrecta sendo tódalas demáis correctas, es dicir, conteñen valores numéricos

Os resultados destas tres probas mostran que o estado do sistema non cambia e non queda en situación inestable; Se hai un espectro cargado queda tal e como está antes de facer as probas, se non había ningún cargado o sistema permanece na mesma situación que antes de calquera destas probas.

Táboas de decisión

Lista tódalas condicións de entradas que poden suceder e tódalas accións que poden surxir delas. Asegúrase de que todas las combinacións das condicións que pode suceder son probadas. Axudan na elección sistemática de casos de proba efectivos e que poden ser beneficiosas a hora de encontrar defectos ou ambigüidades nas especificación. Neste proxecto de momento non se lle ve a utilidade a este tipo de probas, pero igual nun futuro próximo son moi útiles.

Probas de integración

Unha vez que os diferentes módulos dos que se compón o sistema foron probados de forma independente nas probas de unidade pásase as de integración, nas que se comproban que os distintos módulos funcionan de forma conxunta existindo unha correcta comunicación entre eles.

5.3.2 Transicións de estado

Como este software ten moitos estados son unha ferramenta bastante útil. Non se detallan tódolos estados na táboa 5.33 porque serían un montón, cada un deles s_3, s_4, s_5 e s_6 está formado por varios subestados moi relacionados. Por unha razón similar tampouco se detallan tódalas entradas posibles na táboa 5.34. O mesmo pasa coas transicións de estado, non se mostran tódolas posibles porque non tería moita utilidade detallar o funcionamento do sistema como se fose unha máquina de estados finitos. As diferentes transicións entre os estados simplificados veñen dadas pola seguinte táboa 5.35

Estados do handler	
s_0	Estado inicial do handler
s_1	Handler con espectro cargado correctamente
s_2	Handler con zoom activado
s_3	Handler con liñas de H, He ou metalicidade ou bandas primarias ou secundarias amosadas sen zoom
s_4	Handler con liñas de H, He, metalicidade ou bandas primarias ou secundarias amosadas e zoom aplicado
s_5	Handler con algunha función estatística (Avg, Sigma, Der ou Cont) amosadas sen zoom
s_6	Handler con algunha función estatística (Avg, Sigma, Der ou Cont) amosadas e zoom aplicado
s_7	Handler con sigma distribución mostrado
s_8	Handler con sigma distribución mostrado e zoom aplicado
s_9	Handler con algún resultado de calquera tipo de análise ou clasificación

Táboa 5.33: Diferentes estados do handler

Estradas do handler	
i_0	Un espectro válido
i_1	Un espectro inválido
i_2	Fai zoom
i_3	Sale do zoom
i_4	Amosar ou ocultar as liñas de H, He, metalicidade ou bandas primarias ou secundarias
i_5	Eliminar as liñas de H, He, metalicidade e bandas primarias e secundarias
i_6	Amosar ou ocultar as funcións estatísticas Avg, Sigma, Der ou Cont
i_7	Eliminar as funcións estatísticas (Avg, Sigma, Der e Cont)
i_8	Amosar ou ocultar sigma distribution
i_9	Amosar a análise do espectro cargado
i_{10}	Amosar a análise nova versión do espectro cargado
i_{11}	Amosar a clasificación por redes neuronais segundo o sistema M-K do espectro cargado
i_{12}	Amosar a clasificación por luminosidade do espectro cargado
i_{13}	Amosar a clasificación nova versión (Global T) do espectro cargado
i_{14}	Amosar o análise nova versión dun ou máis espectros
i_{15}	Amosar a clasificación por redes neuronais segundo o sistema M-K dun os máis espectros
i_{16}	Amosar a clasificación por luminosidade dun ou máis espectros
i_{17}	Amosar a clasificación nova versión (Global T) dun os máis espectros

Táboa 5.34: Diferentes entradas do handler

Estados	Transición																	
	Entrada																	
	i_0	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}	i_{17}
s_0	s_1	s_0	s_0	s_0	s_0	s_0	s_0	s_0	s_0	s_0	s_0	s_0	s_0	s_0	s_0	s_0	s_0	s_0
s_1	s_1	s_1	s_2	s_1	s_3	s_1	s_5	s_1	s_7	s_9	s_9	s_9	s_9	s_9	s_9	s_9	s_9	s_9
s_2	s_0	s_2	s_2	s_1	s_3	s_1	s_5	s_1	s_7	s_9	s_9	s_9	s_9	s_9	s_9	s_9	s_9	s_9

Táboa 5.35: Transición de estados resumida

Capítulo 6

Resultados

Neste capítulo farase fincapé nos obxectivos cumpridos e nalgunhas melloras obtidas respecto a versión orixinal de Starmind. Por unha parte mostrarase todo o que se cumpliu, e o que non se cumpriu por diversos motivos, además das melloras obtidas, que non deixan de ser sigmificativas.

6.1 Obxectivos cumpridos

Se nos cinguimos estrictamente aos obxectivos do anteproxecto temos os seguintes:

1. Implementación dunha aplicación web que permita o manexo e análise da información espectral procedente de campañas astronómicas. Por unha parte, permitirá a visualización de espectros así como tamén de información que aporte coñecemento adicional sobre a observación, como as posicións no ceo

2. Por outra banda, permitirá a realización de cruces de datos con outros catálogos así como a súa clasificación mediante unha aplicación implementada no grupo de investigación dos directores de proxecto (Starmind); Sendo preciso implementar algún servizo remoto entre a aplicación web e o clasificador, este último, é o que implementa Starmind

3. Implementación dun servizo remoto entre a aplicación web e o clasificador

Con Starmind2 xa en funcionamento pódese observar directamente o que se mellorou con respecto a versión orixinal.

6.2 Melloras obtidas

En canto as melloras obtidas, son varias e significativas:

1. Eliminada a dependencia da ferramenta propietaria Borland C++ Builder

2. Uso de software basado en capas
3. Agora é unha aplicación web multiusuario que non precisa de ningún tipo de instalación onde a vaian utilizar os expertos en astrofísica. O único que precisan é dun navegador web e dunha conta de usuario.
4. A mantenza é adsequible. Antes desta versión era practicamente inviable
5. Permite a escalabilidade segundo se dispoña dun maior número de núcleos ou procesadores
6. Aumenta en gran medida o rendemento de tres funcionalidades moi pesadas
7. Aumenta o grao de aproveitamento do hardware, pois empréganse instruccións de paralelización, ademáis de que o núcleo da aplicación foi compilado para 64 bits
8. Optimiza o uso de memoria
9. É máis robusto que a versión orixinal

Como se observa as diferentes melloras obtidas a partir do **punto 2** son derivadas dese mesmo punto. En canto as interpretacións que fai o autor do traballo da lista anterior, destácanse os seguintes puntos.

En canto ao **punto 1** a vantaxe principal de non depender dos productos de Borland (actualmente en mans de empresa *Embarcadero Tecnologies*), sen estar en contra do software propietario, é que segundo van evolucionando eliminan ou cambian compoñentes que se usan, o cal pode ser un grave problema ao hora de manter actualizado o software, tamén é un inconvinte ter que comprar a ferramenta de cada vez que se queiran modificar ou engadir funcionalidades. Se Borland elimina un ou máis compoñentes utilizados por Starmind orixinal e non se susbtitúen por algunhos similares, existe un grave problema coa actualización e mantenza da versión inicial. Outro problema do Borland Builder é que dificulta a separación do código en capas, con tódolos inconvintes que esto conleva segundo se irán vendo nos próximos puntos. Todos estes defectos nomeados neste parágrafo foron eliminados coa mellora do **punto 1**.

En canto ao **punto 2** as grandes vantaxes de usar o código separado en capas son máis que evidentes. Favorecen a detección e corrección de erros, a fácil mantenza do código ao engadir, modificar ou mesmo eliminar funcionalidades. Se unha aplicación da envergadura de Starmind2 non está feita en capas, ou é practicamente imposible de que chegue a funcionar correctamente, e no insólito caso de que chegase a funcionar correctamente, sería practicamente imposible facerlle calquera tipo de mantenza.

En canto ao **punto 3** ao ser unha aplicación web, que é unha tendenza na actualidade, o seu uso non está restrinxido as máquinas onde foi instalada, senón que se pode acceder desde

calquera dispositivo que dispoña dun navegador web actualizado, incluso sería posible utilizala desde un dispositivo móbil, ben facéndoa completamente *responsive* ou ben construíndo unha aplicación móbil para alguhna das plataformas existentes. Unha vantaxe de todo esto é que sin querer foi construída unha API que permitiría a utilización sen maior problema por terceiros, sexan aplicacións *Android*, *IOS*, *Gtk*..., co cal queda unha porta aberta a futuras funcionalidades que non foron pensadas nin ao principio, nin practicamente ao final deste proxecto. Ao ser unha aplicación web só necesita ser executada por parte de algún servizo que estará nalgún servidor físico ou virtual.

En canto ao **punto 4** ao empregar principios e patróns de deseño estase facilitando directamente e indirectamente a mantenza do aplicación. Este punto está moi relacionado co **punto 2**.

En canto ao **punto 5** é evidente que se algo non se pode paralelizar, o seu rendemento é independente do número de núcleos ou procesadores, pois só poderá ser executado dun deles. É coñecido que fai unhos anos a frecuencia dos procesadores foi deixando de subir paulatinamente ata quedar practicamente estancada, e foi cando se popularizou o uso de procesadores multicores por parte do gran público. A pesar de haber o hardware multicore, a maioría das aplicacións non estaban preparadas para os sistemas de memoria compartida, así que moitas aplicacións tiveron que ser refeitas. O paralelismo a nivel de compilador é un filón que está aínda sen explotar, se xa estivese explotado non habería practicamente que refacer moitas das aplicacións, senón que con só dispoñer do código fonte, un compilador de última xeración sería capaz de percatarse do que se pode paralelizar e máis o que non. De haber un compilador deste estilo tamén baixaría a cantidade de carga de traballo que dou Starmind2. A versión orixinal de Starmind estaba codificada para utilizar só un procesador ou núcleo, algo que naquela época era habitual polo hardware existente, pois o feito de dispoñer dun procesador de máis dun núcleo, ou mesmo dunha máquina con máis dun procesador, era algo prohibitivo económicamente para a maioría de organizacións de astrofísicos que usasen a aplicación. Además hai que ter en conta que cada usuario de Starmind orixinal precisaría dun equipo destas prestacións para el só, tal é como estaba Starmind orixinal. O feito de que se poidan usar tódolos procesadores ou núcleos, que estean dispoñibles, dalle unha grande posibilidade de escalar ao poder aproveitar máis recursos. Loxicamente o sistema non pode escalar indefinidamente, ao empregar memoria compartida, pero dista moito da versión inicial dun so núcleo.

En canto ao **punto 6** é obvio que está moi relacionado co anterior. Ao permitir que métodos ou funcionalidades poidan usar máis dun core, pois estáselle dando escalabilidade ao producto de forma indirecta, o cal tamén repercute en que a comunidade de expertos en astrofísica que usen Starmind2, observe que cando manda a procesar un conxunto grande de espectros, o tempo de resposta é bastante menor.

En canto ao **punto 7** é importante recalcar que ao compilar a aplicación con un compi-

lador de 64 bits, pódese dispoñer dun espazo de direccionamento moi grande, posibilitando aproveitar máis de 4 GiB de memoria. Coa versión orixinal non se podían aproveitar máis alá da citada cantidade de memoria, co cal sería un desperdicio tendo en conta a cantidade de memoria dun computador persoal actual. Además tendo en conta que a aplicación execútase nun servidor, de non ser por esta mellora estariase desaproveitando unha gran cantidade de recursos da máquina de cómputo.

En canto ao **punto 8**, Starmind2, usa a memoria dunha forma máis optimizada que na versión orixinal. Na clase *TEstrella* había varios atributos con valores de tipo *float* e *unsigned short* sendo arrays de 15000 elementos cada un deles. Na nova versión contrólanse o número de mostras do espectro, e segundo eso resérvase memoria exactamente co número de mostras que conteña cada espectro a procesar, además de controlar que non pase dun certo número de mostras. Un espectro real non soe conter máis de 4000 mostras, co cal as vantaxes da optimización son bastante evidentes. Un efecto real disto é que o servidor, independentemente das características que teña, poderá atender simultáneamente un maior número de usuarios, que se non se levase a cabo esta optimización.

Por último con respecto ao **punto 9** convén recalcar que na versión orixinal practicamente non había control de erros, estaba pensada para ser utilizada por usuarios ben intencionados, pero en Starmind2, ao tratarse entre outras partes, dun servizo remoto, pois hai que controlar as condicións de erro dende o principio ata o final, porque un erro de execución neste caso implicaba deixar a tódolos usuarios da aplicación sen servizo. Por tanto faise un control de erros bastante estricto, o cal non quere dicir que o servizo sexa invulnerable, senón que simplemente intentouse facelo o máis robusto posible, tengo en mente a seguridade desde o principio.

Capítulo 7

Conclusións e traballos futuros

7.1 Conclusións

Na etapa final está implementada unha aplicación web que permite a utilización simultánea por parte dos usuarios. Ten unhas funcionalidades similares a Starmind orixinal. Incluso algunhas delas foron optimizadas con diversas tecnoloxías de forma que teñen un tempo de resposta significativamente menor.

Nos comezos de Starmind2 todo parecía moito máis complicado, mesmo as veces determinados obxectivos parecian difícilmente inalcanzables para o autor, pois non tiña experiencia nas técnicas e tecnoloxías aquí empregadas.

O interese na consecución dun determinado fin cunhos medios axeitados da un alto grao de garantía de que se chegará a obter un producto final razoablemente satisfactorio. Neste caso os medios axeitados foron o tempo adicado, o esforzo empregado e o traballo feito por tódolos membros, xefe de proxecto, analista e programador.

Outro aspecto a destacar é que nunca se sabe a complexidade real de algo, mentres un non está involucrado nun proxecto, sendo necesario tomar varias decisións en solitario que afecten a todo o proxecto garantindo o cumprimento dos requisitos iniciais. Ao usar reenxeñería vanse descubrindo funcionalidades e posibles problemas, os cales aparecen máis veces das previstas.

Foi necesario superar algunhas incidencias e problemas durante o seu trancurso, os comentados nos capítulos anteriores non foron máis que os máis tediosos e salientables.

A capacidade de actualización dun software ven dado pola calidade empregada en tódalas fases das que se compón un proxecto, sendo directamente proporcional coa calidade empregada nas diferentes fases.

Os procesos de reenxeñaría poden ser moito máis complicados que os de enxeñería directa. Starmind2 é unha proba máis disto, obtendo directamente moitas das funcionalidades e características empregando o software orixinal e mesmo executándoo varias veces.

Despóis de participar en Starmind2 o autor decatouse do importante que é o entendemento entre os membros do equipo, a pesar de que teñan diferentes puntos de vista. Tamén despóis disto é máis fácil comprender certos motivos polos que a proxectos máis grandes disparouselles o seu custe, e outros foron cancelados cando xa tiñan un grande custe xenerado.

Unha aplicaciòn de tipo científico coma esta, precisa da estreita colaboración de expertos na materia. É unha diferenza bastante grande con respecto ao resto das aplicacións.

A linguaxe C++ a día de hoxe é indudablemente moito menos utilizada que Java. Dado que o producto final ten o núcleo implementado en C++ estándar, pódese afirmar con rotundidade que C++ non é algo anticuado nin deixa de ser apto para construir aplicacións en moitos ámbitos, simplemente que a súa curva de aprendizaxe é máis grande que Java.

A paralelización do código debe ser algo que forme parte do deseño dende o principio ata o final. En Starmind2 levouse a cabo unha reenxeñería, foi un pouco máis complicado, pero con todo, existen diferentes técnicas que se adaptan a múltiples situacións.

En relación ao resultado, foi bastante satisfactorio segundo os membros do equipo. Esto non quere dicir, nin moito menos que fose impecable, nin que non haxa nada que mellorar, a perfección non existe da mesma forma que ∞ non é un número real. Tendo en conta tódalas incidencias e contratempos que sucederon durante todo o desenvolvemento, mirando a lista dos obxectivos iniciais antes de estar metido de cheo, e vendo os obxetivos cumplidos ao final, o autor síntese bastante satisfeito de que tanto esforzo chegase a dar como fruto un producto final aceptable cunha certa calidade.

O coñecemento adquirido durante as etapas de desenvolvemento de Starmind2 foi significativo e de diversos ámbitos. O feito de participar dalgunha forma nun proxecto coma este, fai que un se percate de que non só importan os coñecementos técnicos e as habilidades coas que se manexen, senón que tamén ten moita importancia saber como comunicarse co resto dos membros do equipo. Sen unha comunicación fluída entre un equipo é moi difícil obter un producto final que cumpla os obxectivos para os construído.

7.2 Continuación e próximas melloras

Sen caer no bucle infinito do perfeccionamento, xa comentado neste capítulo, sempre haberá modificacións ou engadido de novas funcionalidades útiles, ou as necesarias debido a aparición de novos requisitos. Unha orde posible e razoada de posibles novas funcionalidades, pode ser a seguinte lista, puntualizando que son independentes entre sí:

1. A integración do sistema cun servizo de directorio como pode ser *LDAP* ou *Active Directory*. Soe ser algo moi útil para administrar os expertos en astrofísica que utilicen a aplicación.

2. Un xestor de colas con varios niveles de prioridade, manexando correctamente a concorrencia e os criterios de igualdade. Esto último será moi útil en canto a aplicación teña unha carga significativa de traballo. Ata chegar a este punto funcionará perfectamente sen este sistema de prioridades entre unhos usuarios e outros.

3. Que sexa un deseño web totalmente *responsive*. A aplicación é plenamenente operativa desde un dispositivo de pantalla pequena, sempre e cando teña potencia suficiente para manexar as gráficas. Pero o seu uso desde un dispositivo deste estilo non é algo cómodo tendo que desplazarse en horizontal para usar o menú. *responsive*.

4. Desde o punto de vista científico, podía pensarse en amplia-la funcionalidade a obxectos do universo como poden ser quásares, estrelas binarias.... Dada a complexidade e a variedade de obxectos astronómicos existentes non é trivial.

7.3 Relación coas competencias da titulación

Para ver a relación coas competencias xerais do grao e coas específicas das mencións, mostraranse extraídas directamente da memoria do grao, citando textualmente as que se lles fai referencia de forma moi evidente:

7.3.1 Conpetenzas xerais

- "CB2 - Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio."

- "4 - Capacidad para definir, evaluar y seleccionar plataformas hardware y software para el desarrollo y la ejecución de sistemas, servicios y aplicaciones informáticas, de acuerdo con los conocimientos adquiridos según la tecnología específica del itinerario cursado."

- "8 - Conocimiento de las materias básicas y tecnologías, que capaciten para el aprendizaje y desarrollo de nuevos métodos y tecnologías, así como las que les doten de una gran versatilidad para adaptarse a nuevas situaciones."

En relación as competanzas xerais, o traballo está moi relacionado coa *CB2* porque se resolveron problemas dunha forma bastante satisfactoria coa axuda dos titores, en relación coa *4* houbo que seleccionar entre varias alternativas tanto de desenvolvemento coma de producción, evaluando as vantaxes e inconvintes de cada unha delas, como se indica no capítulo 3

e por último en relación coa *8* houbo que aprender varias tecnoloxías diferentes como C++, Swig, Ajax, CSS... para poder levar a cabo a consecución de tan variadas funcionalidades.

7.3.2 Competenzas específicas da mención Tecnoloxías da Información

- ”IT3 - Capacidad para emplear metodologías centradas en el usuario y la organización para el desarrollo, evaluación y gestión de aplicaciones y sistemas basados en tecnologías de la información que aseguren la accesibilidad, ergonomía y usabilidad de los sistemas.”
- ”IT7 - Capacidad para comprender, aplicar y gestionar la garantía y seguridad de los sistemas informáticos.”

En relación coas competenzas específicas de Tecnoloxías da Información, Starmind2 está bastante relacionado coa *IT3*, pois empréganse metodoloxías que faciliten o máximo posible dentro do razoable a facilidade de uso por parte dos usuarios e por último coa *IT7*, cando se ten en conta a seguridade desde os comezos do proxecto, pois xa se ten claro que os contrasinais non poden viaxar en claro pola rede, nin mesmo ser almacenados en claro nunha base de datos, e poñendo como un dos obxectivos, que a conexión entre o navegador web e o servidor será sempre de forma cifrada.

7.3.3 Competenzas específicas da mención Enxeñaría de Computadores

- ”CE3 - Capacidad de analizar y evaluar arquitecturas de computadores, incluyendo plataformas paralelas y distribuidas, así como desarrollar y optimizar software para las mismas.”
- ”CE5 - Capacidad de analizar, evaluar y seleccionar las plataformas hardware y software más adecuadas para el soporte de aplicaciones empotradas y de tiempo real.”
- ”CE7 - Capacidad para analizar, evaluar, seleccionar y configurar plataformas hardware para el desarrollo y ejecución de aplicaciones y servicios informáticos.”

En relación coas competenzas específicas de Enxeñería de Computadores Starmind2 presenta unha relación significativa coa *CE3* optimizando software e mesmo nalgunhos casos redeseñalo optimizado para plataformas de memoria compartida, coa *CE5* analízanse diferentes plataformas de hardware paralelo como poden ser sistemas multinúcleo, e por último coa *CE7* codificando coas diferentes directivas de openMP para obter un tempo de resposta significativamente menor nalgunhos métodos costosos computacionalmente.

O feito de que non se nomeen tódalas competenzas xerais e específicas de cada unha das dúas mencións cursadas, non quere dicir que non se tocasen durante todo ou algunha fase do desenvolvemento de Starmind2, simplemente que non están tan relacionadas dunha forma tan evidente como están as citadas nesta sección.

Apéndices

Apéndice A

Material adicional

A.1 Clasificación das diferentes arquitecturas paralelas: Taxonomía de Flynn

Está baseada nos fluxos de instrucións e datos presentes no sistema. Existen as seguintes categorías:

- SISD: Single Instruction Single Data. Exemplos: Uniprocesadores
- MISD: Multiple Instruction Single Data. Exemplos: Múltiples procesadores sobre un único fluxo de datos
- SIMD: Single Instruction Multiple Data. É un modelo de programación sinxelo caracterizado pola execución síncrona da mesma instrución sobre datos diferentes
- MIMD: Multiple Instruction Multiple Data. É un modelo de execución asíncrona de múltiples "segmentos" de código diferentes sobre datos diferentes

A.2 Algunhas definicións útiles

Green Computing *Green Computing tamén coñecido como Green IT ou traducido ao galego como Tecnoloxías Verdes refírese ao uso eficiente dos recursos computacionais minimizando o impacto ambiental, maximizando a súa viavilidade económica e asegurando deberes sociais. Non só identifica a principais tecnoloxías consumidoras de enerxía e productores de desperdicios ambientais senón que tamén ofrece o desenrolo de productos informáticos ecolóxicos e promove o reciclaxe computacional.*

ANSI *O Instituto Nacional Estadounidense de Estándares, máis coñecido como ANSI (pola súas siglas en inglés: American National Standards Institute), é unha organización sen fins de*

lucro que supervisa o desenrolo de estándares para productos, servizos, procesos e sistemas nos Estados Unidos.

IEEE *O Instituto de Enxeñaría Eléctrica e Electrónica (coñecido polas siglas IEEE Institute of Electrical and Electronics Engineers) é unha asociación mundial de enxeñeiros adicada a normalización e ao desenrolo en áreas técnicas.*

LDAP *LDAP son as siglas de Lightweight Directory Access Protocol (en galego Protocolo Lixeiro/Simplificado de Acceso a Directorios) que fan referencia a un protocolo a nivel de aplicación que permite o acceso a un servizo de directorio ordeado e distribuído para buscar diversa información nun entorno de rede.*

Active Directory *Active Directory (AD) ou Directorio Activo son os termos que utiliza Microsoft para referirse a súa implementación de servizo de directorio nunha rede distribuida de computadores. Utiliza distintos protocolos, principalmente LDAP, DNS, DHCP e Kerberos.*

Modelo vista controlador *Modelo-vista-controlador (MVC) é un patrón arquitectónico de software, que separa os datos e máis a lóxica de negocio dunha aplicación da súa representación e o módulo encargado de xestionar os eventos e as comunicacións. Para esto o modelo MVC propón a construcción de tres compoñentes distintos, que son o modelo, a vista e o controlador, é dicir, por un lado define compoñentes para a representación da información, e por outro lado para a interación co usuario. Este patrón arquitectónico de software básase na reutilización de código e a separación de conceptos, buscando facilitar a tarefa de desenvolvemento de aplicacións e o seu posterior mantenemento.*

Relación de Acrónimos

ERLANG/OTP *Erlang Open Telecom Platform.*

API *Application Programming Interface*

PFC *Proxecto fin de carreira*

BOE *Boletín Oficial del Estado*

LDAP *Lightweight Directory Access Protoco*

CUDA *Compute Unified Device Architecture*

Glosario

Bytecode Código independente da máquina que xeran compiladores de determinadas linguaxes (Java, Erlang,...) e que é executado polo correspondente intérprete.

Bibliografía

[1] M. T. Barriuso Pérez, *Física de las energías, física teórica, física nuclear Didáctica e historia de la física y de la química, divulgación de la física, enseñanza de la física (encuentros ibéricos), mujeres en la física Simposio especial Albert Fert, simposio multidisciplinar-óptica, termodinámica Astrofísica, física de plasmas, física de la materia blanda, física médica, información cuántica.* PubliCan, 2011.

[2] E. e. C. Santiago Burbano de Ercilla, *Física General*, 3rd ed. Tébar, 2003.

[3] Antares: Guía de astronomía y astrofísica para profesores de educación secundaria. [En liña]. Dispoñible en: http://ntic.educacion.es/w3/eos/MaterialesEducativos/mem/antares/observatorio/mods_uds/0guia.html

[4] D. Fernández Castrillón, "Diseño e implementación de una aplicación web para la gestión y clasificación de información procedente de observatorios astronómicos," 2010. [En liña]. Dispoñible en: http://kmelot.biblioteca.udc.es/search*gag~S28/Y?SEARCH=Starmind&searchscope=28

[5] P. A. M. Páez, "Análisis de espectros de alta resolución de estrellas frías en el infrarrojo cercano," 2011. [En liña]. Dispoñible en: https://eprints.ucm.es/13752/1/Paulo_Alberto_Miles_Paez.pdf

[6] R. v. d. Pas, *Using OpenMP– the next step : affinity, accelerators, tasking, and SIMD.* Cambridge, MA : The MIT Press, 2017.

[7] Swig. [En liña]. Dispoñible en: http://www.swig.org/

[8] F. J. Ceballos Sierra, *C/C++ : curso de programación.* Paracuellos del Jarama: Ra-ma, 2015.

[9] Cplusplus. [En liña]. Dispoñible en: http://www.cplusplus.com/

[10] A. P. Hinojosa Gutiérrez, *Python: paso a paso.* Paracuellos del Jarama: Ra-ma, 2016.

[11] T. G. Salvaggio, Alessandra, *JavaScript : Guía completa.* Barcelona : Marcombo, 2019.

[12] B. S. al.], *Borland C++ Builder 6 developer's guide.* Indianapolis, Indiana : Sams, 2003.

[13] Django: The web framework for perfectionists with deadlines. [En liña]. Dispoñible en: https://www.djangoproject.com/

[14] A. Mele, *Django by example : create your own line of successful web applications with Django.* Birmingham, UK : Packt Publishing, 2015.

[15] omp4j openmp for java 6/7/8. [En liña]. Dispoñible en: http://www.omp4j.org/home

[16] R. S. Pressman, *Ingeniería del software : un enfoque práctico.* México D. F. : McGraw-Hill, 2010.

[17] R. Laza Fidalgo, *Metodología y tecnología de la programación.* Prentice Hall, 2008.

[18] L. Debrauwer, *UML 2.5 : Iniciación, ejemplos y ejercicios corregidos.* Barcelona : Ediciones ENI, 2016.

[19] C. Jiménez de Parga, *UML : aplicaciones en Java y C++ / Carlos Jiménez de Parga ; revisión técnica: Manuel Arias Calleja.* Paracuellos de Jarama : RA-MA, 2015.