UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

Marco de Comparación de Algoritmos de Question Answering

Autor:
David Carracedo Montero

Titor:
Juan Carlos Vidal Aguiar

Cotitores:
Manuel Lama Penín
Efrén Rama Maneiro

Grao en Enxeñaría Informática

Xuño 2022

Traballo de Fin de Grao presentado na Escola Técnica Superior de Enxeñaría da Universidade de Santiago de Compostela para a obtención do Grao en Enxeñaría Informática

D. Juan Carlos Vidal Aguiar, Profesor do Departamento de Electrónica e Computación da Universidade de Santiago de Compostela, **D. Manuel Lama Penín**, Profesor do Departamento de Electrónica e Computación da Universidade de Santiago de Compostela, e **D. Efrén Rama Maneiro**, Investigador Predoutoral da Universidade de Santiago de Compostela,

INFORMAN:

Que a presente memoria, titulada *Marco de Comparación de Algoritmos de Question Answering*, presentada por **D. David Carracedo Montero** para superar os créditos correspondentes ao Traballo de Fin de Grao da titulación de Grao en Enxeñaría Informática, realizouse baixo a nosa titoría no Departamento de Electrónica e Computación da Universidade de Santiago de Compostela.

E para que así conste aos efectos oportunos, expiden o presente informe en Santiago de Compostela, a 2 de Xuño do 2022:

Titor,	Cotitor,	Cotitor,	Alumno,
VIDAL AGUIAR JUAN CARLOS - 32823890S Digitally signed by VIDAL AGUIAR JUAN CARLOS - 32823890S Date: 2022.06.14 18:11:06 +02'00'	Firmado digitalmente por LAMA PENIN MANUEL - 33288612E Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-33288612E, givenName=MANUEL, sn=LAMA PENIN, cn=LAMA PENIN MANUEL - 33288612E Fecha: 2022.06.14 23:31:11 +02'00'	RAMA MANEIRO EFREN - 53798548F Firmado digitalmente por RAMA MANEIRO EFREN - 53798548F Fecha: 2022.06.14 21:03:10 +02'00'	
Juan Carlos Vidal Aguiar	Manuel Lama Penín	Efrén Rama Maneiro	David Carracedo Montero

Resumo

Nos últimos anos, os grandes avances en tecnoloxías como o procesamento de linguaxe natural ou a recuperación de información, xunto co resurxir das técnicas de aprendizaxe profundo (DL, do inglés Deep Learning), están propiciando a aparición de novas e máis potentes técnicas no ámbito dos sistemas de pregunta-resposta (QA, do inglés Question-Answering). Non obstante, a complexidade inherente destas técnicas, normalmente dos algoritmos de DL ou de agrupacións de modelos de DL (ensembles), ten asociadas algunhas características que dificultan a comparación entre distintas aproximacións. Por un lado, o adestramento e proba destes modelos necesita dunha potente infraestructura. Ésta é a principal dificultade, é unha das razóns polas que habitualmente non se reproduce a experimentación indicada polos autores do modelo para un determinado conxunto de datos. Por outro lado, os procesos de adestramento e proba non sempre están correctamente configurados, como puidemos comprobar neste TFG, e utilizan distintos datos para adestrar e probar, ou directamente filtran parte dos datos.

Por todos estes motivos, neste Traballo Fin de Grao (TFG) desenvolveuse unha plataforma integral de probas para a comparación de algoritmos de QA co obxetivo de (i) minimizar os inconvenientes atopados á hora de comparar distintas aproximacións por medio dunha comparación xusta e, ademais, (ii) permitir a reutilización dos resultados de probas anteriores, facilitando así a incorporación de novos algoritmos ás comparativas cun mínimo esforzo.

Memoria tipo B – Índice xeral

Índice de figuras

Índice de cadros

Glosario

Este apartado recolle aqueles termos empregados nesta memoria que son relativos ao campo de estudo do traballo ou á implementación realizada.

API : interface de programación de aplicacións. No contexto deste traballo, fai referencia ao servidor REST do sistema.

Chatbot : asistente virtual que proporciona asistencia aos usuarios a través de mensaxes de texto [22].

Dataset : ficheiro con cuestións e datos necesarios para o adestramento ou avaliación dun algoritmo de Question Answering.

DL : acrónimo de Deep Learning.

Entorno : espazo illado e independente que contén todo o código dun algoritmo e as dependencias e ficheiros necesarios para a súa execución.

HTTP : acrónimo de Hypertext Transfer Protocol [26].

IR : acrónimo de recuperación de información.

NLP : acrónimo de procesamento de linguaxe natural.

QA : acrónimo de Question Answering.

REST : patrón de arquitectura de servizos sen estado no servidor, interface uniforme, sistema de capas e especificación HTTP [23].

Script : conxunto de instrucións contidas nun ficheiro e executables por un intérprete.

SCP : acrónimo de Secure Copy [25].

SSH : acrónimo de Secure Shell [24].

Capítulo 1

Introdución

Os sistema de pregunta-resposta (QA, do inglés Question Answering) é unha rama do campo do procesamento da linguaxe natural (NLP) e da recuperación de información (IR) que está orientada a responder, de maneira autónoma e sen intervención humana, a preguntas realizadas por persoas en linguaxe natural [1].

Este campo das ciencias da computación está moi extendido, por exemplo, dentro de múltiples implementacións de *chatbots* e de motores de búsqueda, e permiten que o usuario realice unha pregunta e o sistema responda, coa máxima exactitude posible, á súa dúbida. Para isto, empréganse unha serie de algoritmos que se escollen en función do eido no que se van empregar e que son adestrados a partir de grandes conxuntos de datos. O obxectivo é atopar aquel algoritmo que se comporte mellor na situación concreta na que se vai facer uso del, polo que é neste contexto onde aparece a necesidade de realizar comparativas de rendemento entre múltiples candidatos mediante unha ferramenta de probas que nos axude a escoller aquel algoritmo que teña un mellor desempeño.

A existencia dunha plataforma que nos permita establecer un marco comparativo entre múltiples algoritmos de forma sinxela, implicaría que imos poder realizar probas cun nivel de esforzo menor e con maior comodidade. Isto conségue-se tendo un sistema que deixe que o usuario seleccione unha serie de algoritmos, conxuntos de datos e parámetros de configuración e que o propio sistema sexa o responsable de xestionar o adestramento e probas necesarias para, finalmente, amosarnos os resultados dunha maneira ordenada, podendo así comparalos e analizar as métricas que máis interesen.

1.1. Motivación

Actualmente existe unha gran variedade de algoritmos distintos no campo dos sistemas de pregunta-resposta, cada un baseado nun tipo de arquitectura concreto ou habendo multitude de variacións dun mesmo modelo [18]. Por exemplo, existe un estudo que mide o rendemento dos algoritmos de QA sobre un dataset chamado

SQuAD 2.0, desenvolvido na universidade de Stanford [19]. Dentro deste estudo, onde se realiza unha clasificación dos algoritmos en función do resultado obtido para un dataset concreto, existen máis de 200 probas distintas sobre múltiples algoritmos, o que nos dá unha perspectiva da diversidade de opcións que existen na actualidade.

Esta situación non só implica que hai un grande abanico de opcións onde escoller, senón que dentro dos propios algoritmos, hai moitas diferencias baseadas, principalmente, no seu fundamento teórico ou na súa arquitectura interna. Desta maneira, temos moitas características diferentes, que implican que cada algoritmo se executa dun xeito concreto e que hai que realizar unha proba distinta para cada un deles, o que provoca ter que adicar un grande esforzo a adestrar, executar e realizar as adaptacións necesarias para cada caso.

Ademais, cando se realiza unha proba dun algoritmo, á parte do feito de ter que adestralo, normalmente hai que realizar varias execucións sobre un mesmo conxunto de datos, variando os seus parámetros de configuración, co obxectivo de axustar o rendemento e obter os mellores resultados posibles. Isto fai que as probas leven un esforzo temporal elevado, pois non só temos que executar a proba, senón que tamén temos que ir comprobando os resultados, un a un, e axustando os parámetros ata atopar o equilibrio buscado.

Outro feito a destacar á hora de executar un algoritmo é comparar o rendemento, sobre un dataset ou un conxunto de datasets, con respecto a outros algoritmos. Desta forma, podemos realizar comparativas que nos permiten escoller cal ten un mellor desempeño ou ver en que casos un se comporta mellor que outro. Para isto, precisamos, a maiores do axuste comentado anteriormente, realizar as mesmas probas para estes conxuntos de algoritmos, o que tamén fai que a comparativa sexa lenta, implique un esforzo elevado e, en moitos casos, haxa que ir probando cada algoritmo de forma individual para, finalmente, comparar os resultados.

O maior inconveniente da situación actual cos algoritmos de QA é que as probas realizadas, incluíndo os adestramentos necesarios, resultan moi caras tanto en tempo como en recursos físicos, pois as tarxetas gráficas empregadas son un recurso caro e cada nova proba implica, en moitos casos, non poder reutilizar o esforzo realizado en probas anteriores.

Na actualidade, hai ferramentas que axudan a certas fases da proba de algoritmos de QA, pero teñen o problema de que seguen requerindo certo esforzo e non integran todo nun mesmo espazo. Por exemplo, no caso de SQuAD [19], para obter o resultado temos que executar o algoritmo e despois avaliar as respostas a través dun script que se nos proporciona, obtendo así a puntuación pero tendo o inconveniente de ter que executalo para cada unha das probas que queiramos facer, implicando un consumo de recursos físicos ou temporais elevados. Outro caso é a plataforma UKP-SQuARE [21], que permite comparar o rendemento de algoritmos de QA sobre uns datos que nós podemos proporcionar, pero só temos a posibilidade de seleccionar entre un conxunto limitado de algoritmos. Ademais,

non existe a posibilidade de acceder a un servizo de adestramento en liña, o que implica que debemos subir os modelos previamente adestrados, e esta aproximación implica que nos debemos fiar de que dito adestramento se faga sobre os datos de adestramento e non sobre todo o conxunto de datos ou sobre os datos de avaliación, o que supoñería uns resultados pouco precisos e truncados.

A solución proposta neste traballo implica o desenvolvemento dunha plataforma que nos facilite a comparativa de algoritmos de QA. Búscase integrar nun mesmo espazo o adestramento, a execución e o procesado dos resultados de probas realizadas a conxuntos de algoritmos. Ademais, non só se busca centralizar o espazo de execución dos algoritmos, tamén se busca facer que o usuario que desexa realizar a proba teña que realizar o menor esforzo posible, e isto implica que tanto o adestramento como a execución sexan automatizados e non requiran intervención humana para poder obter os resultados. O obxectivo é poder configurar a proba cos datos necesarios e obter os resultados unha vez remate a execución. Tamén se pretende conseguir unha integración entre algoritmos e datos, onde as probas realizadas teñan unha igualdade de condicións e se executen baixo as mesmas circunstancias e datasets.

Esta plataforma implicaría un gran cambio na metodoloxía de comparativa de algoritmos actual, pasando dunha realización de probas de forma "manual" a un sistema integrado e unificado de probas que permite establecer un marco comparativo en igualdade de condicións para todos os algoritmos e usuarios que o empreguen.

1.2. Obxectivos xerais

Dentro deste proxecto, o obxectivo principal é a creación dunha plataforma que permita executar probas sobre algoritmos de QA e sobre múltiples datasets, obtendo os resultados das execucións realizadas e podendo comparar o rendemento dos algoritmos.

Co fin de alcanzar con éxito este obxectivo, definimos unha serie de subobxectivos que deben cumplirse ao remate do proxecto:

- **OB-01**: Implementar un sistema de subida e consulta de datasets.
- **OB-02**: Implementar un sistema de subida e consulta de algoritmos.
- **OB-03**: Implementar un sistema que permita consultar as probas executadas.
- **OB-04**: Implementar un sistema que permita executar probas con múltiples algoritmos e avaliar diversos datasets.

1.3. Estrutura da memoria

A estrutura desta memoria está composta por seis capítulos que recollen o traballo elaborado para este proxecto. Tamén se presentan tres apéndices que recollen figuras e cadros da memoria e os manuais necesarios para empregar e traballar sobre o sistema.

- **Capítulo 1**: realiza unha introdución ao contexto do proxecto, a motivación para a súa realización, os obxectivos buscados e os capítulos recollidos nesta memoria.
- **Capítulo 2**: recolle a xestión do proxecto que se aplicou, incluíndo a metodoloxía de traballo e xestión temporal, a xestión da configuración e a xestión dos riscos do proxecto.
- **Capítulo 3**: faise unha análise do proxecto desenvolvido, reflexando o alcance da solución proposta e os requisitos do proxecto.
- **Capítulo 4**: ilustra as decisións de deseño tomadas tanto na arquitectura do sistema como no deseño das interfaces de usuario, así como o funcionamento interno dos diferentes compoñentes do sistema.
- **Capítulo 5**: recolle o plan de probas realizado para o sistema.
- **Capítulo 6**: abrangue as conclusións obtidas tras realizar o proxecto e as posibles ampliacións ao traballo realizado.

Capítulo 2

Xestión do proxecto

Este capítulo da memoria trata aqueles aspectos relativos á enxeñaría do software e da xestión de proxectos. Faise unha análise, indicando os aspectos máis relevantes, da metodoloxía e planificación temporal do proxecto, a xestión da configuración e a xestión de riscos asociados ao proxecto.

2.1. Metodoloxía de desenvolvemento

O proxecto desenvolvido ten como característica a necesidade de probar e adaptar os elementos creados co fin de ir adoptando un producto final que cumpra os obxectivos establecidos. Deste xeito, temos que ter en conta os seguintes criterios á hora de escoller a metodoloxía axeitada:

- Os obxectivos do proxecto son claros pero hai requisitos novos que poden aparecer ou pode ter que modificarse algún dos planificados.
- A retroalimentación por parte dos titores é chave á hora de desenvolver este proxecto, cumprindo as necesidades polas que se leva a cabo o mesmo.
- As tecnoloxías necesarias para desenvolver o sistema non están definidas de forma clara, polo que se precisa realizar tarefas de investigación e probas que permitan escoller unha alternativa viable. Isto implica que pode ser necesario ter que modificar partes implementadas ou determinados aspectos do sistema.

Analizando as problemáticas anteriores, decídese escoller unha metodoloxía áxil orientada a incrementos. O proxecto divídese nunha serie de incrementos que permiten realizar un desenvolvemento iterativo, onde se van introducindo os distintos compoñentes e funcionalidades do sistema, e que permite refinar aqueles aspectos que se poden atopar máis difusos e que presentan máis incertidumbre.

A duración dun incremento comprende un esforzo de entre unha e dúas semanas, podendo nalgún caso ter unha duración maior por factores externos ao

proxecto, como a dispoñibilidade á hora de programar reunións, ou pola importancia/dificultade das tarefas realizadas durante o incremento. Cada incremento comeza cunha reunión cos titores, onde se realiza unha revisión do traballo realizado no incremento anterior e planifícase traballo asociado aos obxectivos propostos para o próximo incremento. Deste xeito, cada incremento é dinámico e permite adaptar o traballo en función das necesidades de cada fase do proxecto.

O traballo realizado durante cada incremento está priorizado en función da importancia dos obxectivos asociados ao sistema, polo que se comeza investigando e desenvolvendo as bases do sistema e, progresivamente, vaise engadindo a funcionalidade necesaria ata rematar o proxecto.

2.2. Xestión temporal

O proxecto ten unha duración total de 16,5 semanas, cun esforzo semanal de 25 horas. Isto suma un traballo total de 412,5 horas. O esforzo distribúese entre seis incrementos, nos que se desenvolve o proxecto, e tarefas de xestión do proxecto, como a organización e controis.

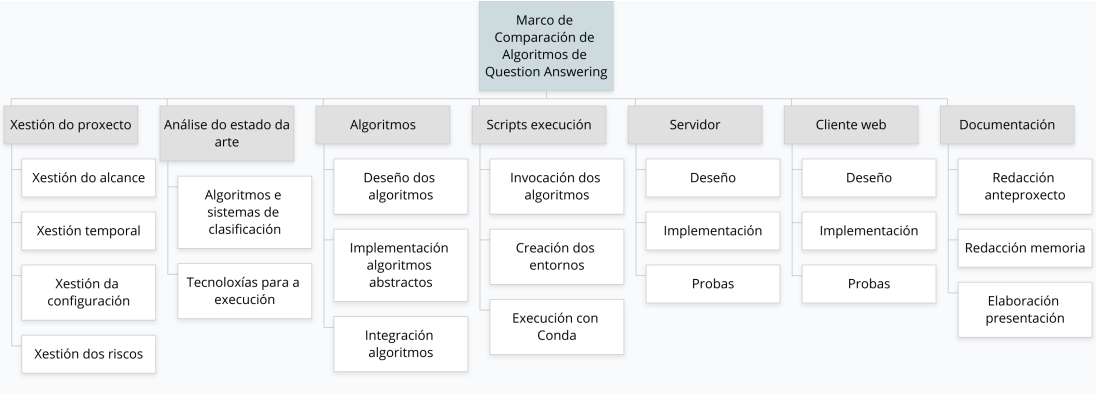


Figura 2.1: Estrutura de descomposición do traballo do proxecto.

Na Figura 2.1 móstrase a estrutura de descomposición do traballo. Á execepción da xestión do proxecto, que engloba todo o transcurso do mesmo, o resto de paquetes correspóndense cos incrementos realizados no proxecto. Os paquetes de traballo son os seguintes:

- **Xestión do proxecto** [1 semana]: agrupa as tarefas de xestión do proxecto e control do mesmo.
- **Análise do estado da arte** [4 semanas]: agrupa as tarefas relacionadas coa investigación dos algoritmos empregados en QA na actualidade, os sistemas de clasificación de algoritmos e as tecnoloxías dispoñibles para realizar o sistema de execución destos algoritmos.

- **Algoritmos** [3 semanas]: agrupa as tarefas de deseño da estrutura dos algoritmos e a implementación e adaptación dos algoritmos integrados na plataforma.
- **Scripts execución** [3 semanas]: agrupa as tarefas de deseño e implementación dos scripts que permiten executar os algoritmos de forma automatizada.
- **Servidor** [2 semanas]: agrupa as tarefas de desenvolvemento do servidor do sistema.
- **Cliente web** [1 semana]: agrupa as tarefas de desenvolvemento do cliente web da plataforma.
- **Documentación** [2,5 semanas]: agrupa as tarefas de documentación asociadas ao proxecto.

2.3. Xestión da configuración

Esta sección recolle o plan de xestión da configuración que se levou a cabo durante a realización deste proxecto. O obxectivo deste plan de xestión é asegurar a integridade e a calidade daqueles elementos necesarios para o correcto cumprimento dos obxectivos do proxecto.

2.3.1. Liña base

Entendemos por liña base aqueles elementos do proxecto que van ser sometidos ao control establecido pola xestión da configuración. Os elementos son os seguintes:

- Código fonte do servidor ou API REST.
- Código fonte da aplicación web cliente.
- Código fonte dos algoritmos probados.
- Documentación do proxecto.

2.3.2. Xestión da configuración

Os elementos que conforman a liña base do proxecto deben asegurar tanto a integridade dos mesmos como a xestión de versións e a posibilidade de acceso compartido por varios usuarios, de modo que tanto alumno como titores poidan acceder en caso necesario. Desta maneira, empréganse as seguintes plataformas para a xestión dos distintos elementos:

- **OneDrive**: neste servicio online, proporcionado na nube da USC, gardaranse todos os arquivos relacionados coa documentación elaborada como axuda a esta memoria e os diagramas e recursos gráficos realizados [15].
- **GitHub**: neste servicio establecerase o repositorio Git que aloxará o código fonte desenvolvido durante o proxecto [9].
- **Overleaf**: neste editor online de documentos LaTex realizarase tanto a redacción da memoria do proxecto como a revisión e almacenamento da mesma [14].

2.4. Xestión de riscos

Como parte do desenvolvemento do proxecto, é necesario identificar os riscos que poidan aparecer e facer que o proxecto non se poida levar a cabo dentro dos prazos establecidos ou que parte dos obxectivos poidan verse afectados. Nesta sección recóllense os aspectos relativos á identificación e xestión dos riscos do proxecto.

2.4.1. Metodoloxía de análise de riscos

Para poder tratar os riscos, é preciso definir un sistema de avaliación cualitativa dos mesmos para poder priorizar as accións necesarias sobre aqueles que teñan unha maior influencia no devir do proxecto.

Para saber que riscos tratar, establecemos a probabilidade de aparición de cada un (Cadro 2.1) e o impacto que tería a súa aparición no proxecto (Cadro 2.2). O resultado da exposición do noso proxecto a un risco concreto é a combinación de ambas métricas (Cadro 2.3). Para este proxecto, tomaremos medidas contra aqueles riscos que supoñan unha exposición *media* ou *alta*, pois serán aqueles que poidan influir na correcta finalización do mesmo.

Probabilidade de aparición	
Alta	A aparición do risco vaise producir, unha ou máis veces, durante o desenvolvemento do proxecto.
Media	A aparición do risco é posible que apareza durante o desenvolvemento do proxecto.
Baixa	Excepcionalmente pode aparecer o risco durante o desenvolvemento do proxecto.

Cadro 2.1: Probabilidade de aparición dun risco.

Impacto do risco	
Alto	Os obxectivos do proxecto non se poden cumprir e non se pode rematar o traballo.
Medio	O risco implica retrasos que poden facer que a entrega final se atrase de convocatoria.
Baixo	O risco podese solventar sen afectar á entrega final nin aos obxectivos.

Cadro 2.2: Impacto dun risco sobre o proxecto.

Exposición		**Probabilidade**		
		Alta	Media	Baixa
Impacto	Alto	Alta	Alta	Media
	Medio	Alta	Media	Baixa
	Baixo	Media	Baixa	Baixa

Cadro 2.3: Cálculo da exposición a un risco.

2.4.2. Identificación de riscos

Unha vez definida a maneira na que avaliar os riscos pola súa probabilidade de aparición e impacto no proxecto, procedemos a identificar os riscos que se poden materializar para este proxecto. A continuación, móstrase a lista dos riscos identificados:

- **RISC-01**: Planificación non se axusta á realidade.
- **RISC-02**: Certas funcionalidades teñen que ser rediseñadas.
- **RISC-03**: Non é posible realizar unha implementación que se adapte aos obxectivos.
- **RISC-04**: Perda dun elemento da liña base.
- **RISC-05**: Fallo do entorno de desenvolvemento.
- **RISC-06**: O sistema non resulta útil.
- **RISC-07**: Algunha funcionalidade non é compatible.

No apéndice A.1 amósase a estimación da probabilidade, impacto e exposición para cada risco identificado.

2.4.3. Planificación dos riscos

Identificados os riscos, e seguindo o criterio de tratamento definido no apartado da metodoloxía de análise de riscos, debemos seleccionar aqueles riscos con exposición media ou superior e propoñer os indicadores (que nos permiten identificar a súa aparición) e as accións de tratamento dos mesmos.

Os riscos a tratar son os seguintes: **RISC-01**, **RISC-03**, **RISC-04**, **RISC-07**.

Tratamento dos riscos

Para cada un dos riscos anteriores, establecemos as medidas de control e accións que levamos a cabo para reducir a súa exposición.

RISC-01 Planificación non se axusta á realidade.	
Indicador	Unha tarefa ten un retraso temporal superior ao 20 % da planificación prevista.
Acción prevención	Establecer marxes na planificación inicial e revisar o progreso respecto ao tempo previsto.
Acción corrección	Replanificar tarefas posteriores á retrasada.

RISC-03 Non é posible realizar unha implementación que se adapte aos obxectivos.	
Indicador	O inicio do desenvolvemento ten un retraso superior a un mes do que estaba previsto.
Acción prevención	Realizar unha investigación das tecnoloxías dispoñibles e das capacidades de cada unha así como das necesidades do proxecto.

RISC-04 Perda dun elemento da liña base.	
Indicador	Aparecen erros no sistema debido a elementos que non se atopan.
Acción prevención	Asegurarse de realizar gardados de versión en intervalos regulares.

RISC-07 Algunha funcionalidade non é compatible.	
Indicador	Aparecen erros de execución no sistema.
Acción prevención	Asegurarse de que os módulos e librerías empregados son compatibles.
Acción corrección	Refactorizar aqueles compoñentes afectados e priorizar o funcionamento dos módulos que xa estaban operativos.

2.4.4. Riscos materializados

A finalización do proxecto foi posible dentro dos prazos establecidos na planificación orixinal. Non obstante, materializáronse algúns dos riscos identificados e houbo que aplicar as medidas oportunas co fin de que o proxecto non se vira comprometido.

Durante a fase inicial do desenvolvemento, parte na que se realizou a implementación do sistema de execución dos algoritmos, o feito de non ter experiencia coas ferramentas empregadas e a aparición dalgún erro nas execucións provocou que o risco **RISC-01** se materializara. Houbo que executar a acción de corrección prevista e replanificar o resto de tarefas. Isto provocou un pequeno retraso pero, debido a que había marxe de tempo, non implicou retrasos nas entregas previstas.

O risco **RISC-07** tamén apareceu nun par de ocasións de cara ao final do proxecto, cando se incluíron determinados aspectos relacionados coa execución de algoritmos a través de *SSH* [24]. Nun principio a execución realizábase invocando os algoritmos na mesma máquina, en local. Ao cambiar o modelo de execución a unha máquina remota, houbo que introducir unha librería que permitira traballar con *SSH* [24] de forma programática, e esto fixo que parte dos scripts de execución deixasen de funcionar. O resultado foi a necesidade de aplicar a acción de corrección e axustar determinadas variables globais empregadas nos scripts para que fosen compatibles coa execución remota.

Capítulo 3

Análise

Este capítulo da memoria define o sistema que se vai implementar, recollendo o alcance do proxecto e a especificación dos requisitos que compoñen o sistema.

3.1. Alcance do proxecto

Dentro deste apartado descríbese o alcance da proposta deste traballo, os criterios para a súa finalización, as restricións impostas e os aspectos que quedan fóra do contexto do proxecto.

3.1.1. Definición do alcance

A proposta desenvolvida neste traballo consiste no desenvolvemento dunha plataforma que permita executar e comparar algoritmos de QA dunha forma sinxela e centralizada.

Proponse un sistema, cunha arquitectura *cliente-servidor*, que lle permita aos usuarios subir unha serie de algoritmos e datasets, e realizar probas sen ter que preocuparse pola xestión de dependencias de cada algoritmo, o adestramento necesario ou as execucións necesarias para obter os resultados. Ademais o sistema permitirá consultar todas as probas executadas anteriormente, así como unha información de cada proba onde se amosen os resultados obtidos para cada combinación de algoritmo e dataset seleccionado e un cómputo global dos resultados, que se obterá realizando unha serie de tests estadísticos sobre os valores obtidos das múltiples execucións dentro dunha proba.

A plataforma desenvolvida permitirálle aos usuarios subir os seus algoritmos, comprimidos nun ficheiro, que conterá o código necesario para a execución e a información necesaria para configuralo. Tamén se poderán consultar os algoritmos dispoñibles no sistema.

Os usuarios tamén poderán subir os datasets que desexen probar á plataforma. Así mesmo, poderanse consultar todos os datasets que se atopan no sistema e escollelos para ser empregados en calquera das execucións.

No relativo á execución, o usuario indicará os algoritmos cos parámetros necesarios e os datasets a avaliar e a plataforma realizará as accións necesarias para que a proba se execute correctamente. Isto inclúe o adestramento daqueles algoritmos que non se atopen adestrados e a avaliación dos diversos algoritmos seleccionados sobre cada un dos datasets que se escolleron para a proba.

3.1.2. Criterios de aceptación do sistema

Para poder definir a finalización existosa do proxecto, definimos unha serie de criterios que nos permiten asegurar que os obxectivos do mesmo se cumpren.

Estes criterios estarán identificados mediante a nomenclatura *CA-[núm. correlativo]*. Empregaremos esta nomenclatura para facer referencia a eles máis adiante, cando realicemos as probas do sistema e comprobemos que se cumpren todos os obxectivos.

O sistema poderá considerarse como rematado na medida en que os criterios descritos a continuación se atopen totalmente satisfeitos:

- **CA-01**: O sistema permite que o usuario realice as funcións propias da plataforma sen producirse erros de execución.
- **CA-02**: O sistema permite tanto a subida de datasets como a consulta por parte do usuario.
- **CA-03**: O sistema permite subir e consultar algoritmos por parte do usuario.
- **CA-04**: O sistema permite consultar as probas realizadas en calquera momento pasado ou actual, incluíndo así tanto as probas en proceso de execución como aquelas que xa remataron e obtiveron os seus resultados.
- **CA-05**: O sistema permite executar probas sobre múltiples algoritmos e datasets ao mesmo tempo.
- **CA-06**: O sistema permite adestrar os algoritmos e avaliálos sobre os datasets.

3.1.3. Restricións do proxecto

Neste apartado expóñense as restricións que aplican a este proxecto e que deben ser cumpridas:

- O proxecto debe rematarse e entregarse antes do día 16 de Xuño do 2022.

- O esforzo asociado ao proxecto debe ser dun mínimo de 400 horas de traballo e non superar as 425 horas.
- A extensión máxima da memoria asociada ao traballo será de 50 páxinas (sen contar o índice, a bibliografía, os apéndices e os manuais necesarios).

3.1.4. Exclusións do proxecto

A continuación, defínense aqueles aspectos que quedan fóra do contexto do traballo realizado:

- A plataforma desenvolvida permitirá probar unicamente algoritmos de *Question Answering*. En ningún caso se dará soporte á execución de calquera tipo de algoritmo de intelixencia artificial, procesamento de linguaxe natural ou recuperación de información.
- A plataforma permitirá executar os algoritmos sobre datasets cun formato igual ao empregado nas probas do proxecto SQuAD 2.0, da universidade de Stanford [19]. Non se dará soporte para calquera outro formato de ficheiro para os datasets.

3.2. Especificación de requisitos

Nesta sección imos definir os requisitos do sistema proposto. Os requisitos permítennos determinar aqules aspectos que son necesarios para que o sistema funcione, a funcionalidade desexada e as restricións sobre o seu comportamento.

3.2.1. Requisitos de información

Empezamos definindo aqueles elementos de información que o sistema debe almacenar co propósito de dar soporte á funcionalidade do mesmo. Os requisitos identificados son os seguintes:

- **RI-01** - **Algoritmo**: representa a información e configuración dun algoritmo subido á plataforma.
- **RI-02** - **Dataset**: representa a información dun dataset subido á plataforma.
- **RI-03** - **Proba**: representa, como unha unidade, un conxunto de tarefas e resultados asociados á execución e avaliación duns algoritmos sobre uns datasets.
- **RI-04** - **Tarefa**: representa a información asociada a unha tarefa a ser executada polo sistema, ben sexa un adestramento dun algoritmo ou unha avaliación dun algoritmo sobre un dataset.

- **RI-05 - Entorno**: representa a información dun entorno de execución creado, a partir duns parámetros de adestramento específicos, para un algoritmo concreto.

No Apéndice A.2 amósanse unha serie de táboas cos requisitos anteriormente descritos e os datos específicos asociados a cada un deles.

3.2.2. Requisitos funcionais

Procedemos a definir a funcionalidade necesaria que debe ter o sistema co propósito de cumprir os obxectivos que dan validez ao mesmo. Os requisitos funcionais identificados indícanse a continuación e a súa descrición realízase a través dos casos de uso que se expoñen na sección 3.2.5:

- **RF-01**: O usuario debe poder subir un dataset á plataforma.
- **RF-02**: O usuario debe poder consultar a lista de datasets subidos á plataforma.
- **RF-03**: O usuario debe poder consultar a información dun dataset concreto.
- **RF-04**: O usuario debe poder subir un algoritmo á plataforma.
- **RF-05**: O usuario debe poder consultar a lista de algoritmos subidos á plataforma.
- **RF-06**: O usuario debe poder consultar a información dun algoritmo concreto.
- **RF-07**: O usuario debe poder consultar as probas en execución.
- **RF-08**: O usuario debe poder consultar o historial de probas executadas.
- **RF-09**: O usuario debe poder consultar unha proba, amosando tanto o estado da mesma como os resultados das execucións e un test estadístico, que permita comparar rendementos, no caso de ter rematado.
- **RF-10**: O usuario debe poder crear unha proba cun conxunto de algoritmos e datasets da plataforma.

3.2.3. Requisitos non funcionais

Os requisitos funcionais definen aquelas condicións que se deben dar no sistema para que este sexa válido. Non introducen ningunha funcionalidade específica pero si restricións sobre a totalidade ou parte do sistema.

RNF-01 Formato de subida dun algoritmo.	
Descrición	**O sistema deberá** permitir a subida de algoritmos nun ficheiro comprimido de tipo *ZIP* que conteña o ficheiro de configuración e o código fonte e ficheiros necesarios para a súa execución.

RNF-02 Formato de subida dun dataset.	
Descrición	**O sistema deberá** permitir a subida de datasets nun ficheiro de tipo *JSON*.

RNF-03 Compatibilidade de algoritmos.	
Descrición	**O sistema deberá** permitir a execución de algoritmos escritos na linguaxe de programación *Python* [2], tanto na versión 2 como 3 da mesma.

RNF-04 Compatibilidade de dependencias.	
Descrición	**O sistema deberá** permitir executar algoritmos con independencia das librerías de *Python* [2] das que estes dependan, sempre e cando ditas librerías sexan compatibles con *Conda* [11].

RNF-05 Execución remota.	
Descrición	**O sistema deberá** permitir realizar a execución dos adestramentos e avaliación en calquera máquina *Linux* accesible por *SSH* [24].

RNF-06 Seguridade de credenciais.	
Descrición	**O sistema deberá** asegurar que as credenciais empregadas para o *SSH* [24] se tratan de forma segura e non se almacena ningún contrasinal en texto plano.

3.2.4. Interacción con outros sistemas

Para cumprir parte da funcionalidade proposta (en concreto a relativa á comparativa de resultados dos algoritmos dunha proba), o sistema interactúa cun sistema externo que nos proporciona a funcionalidade correspondente ao cálculo de tests estadísticos.

Para isto, emprégase a plataforma ***STAC*** do *CiTIUS* [20] que, a través dunha API *HTTP*, permítenos enviar os resultados obtidos polos algoritmos sobre os múltiples datasets e recuperar as métricas de diversos tests estadísticos.

3.2.5. Casos de uso

Os casos de uso son unha representación dos requisitos funcionais na que se detalla, dunha forma máis precisa, aqueles pasos e condicións que se poden dar durante a execución dunha funcionalidade concreta.

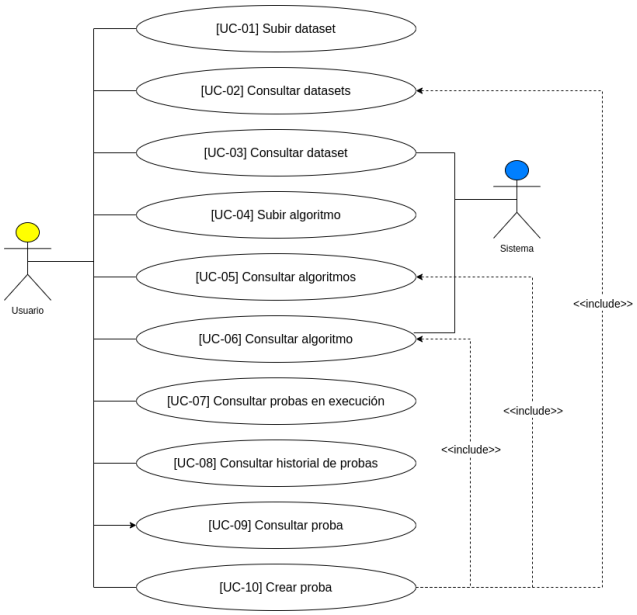


Figura 3.1: Diagrama de casos de uso do sistema

Actores

Para os casos de uso representados nas seguintes seccións, imos empregar dous actores nas interaccións: **Usuario** e **Sistema**.

O actor ***Usuario*** representa ao individuo que accede á plataforma e realiza operacións a través da interface web proporcionada. Faremos referencia a este actor mediante o identificador **ACT-1**.

O actor ***Sistema*** representa o conxunto de compoñentes que forman o sistema da plataforma e que proporcionan a funcionalidade da mesma. Faremos referencia a este actor mediante o identificador **ACT-2**.

Diagrama de casos de uso

Co fin de representar os casos de uso, os actores e as relacións entre estes elementos, móstrase na figura 3.1 o diagrama de casos de uso do sistema.

Descrición dos casos de uso

Os casos de uso definidos no diagrama da Figura 3.1 son os seguintes:

- **UC-01**: Subir un dataset.
- **UC-02**: Consultar datasets.
- **UC-03**: Consultar dataset.
- **UC-04**: Subir algoritmo.

- **UC-05**: Consultar algoritmos.
- **UC-06**: Consultar algoritmo.
- **UC-07**: Consultar probas en execución.
- **UC-08**: Consultar historial de probas.
- **UC-09**: Consultar proba.
- **UC-10**: Crear proba.

Debido ás restricións de espazo desta memoria, descríbense no Apéndice A.3 cada un dos casos de uso, indicando as condicións e excepcións que se poden dar, xunto coa secuencia de pasos que se deben executar para a correcta finalización dos casos.

Capítulo 4

Deseño

Este capítulo trata as decisións de deseño adoptadas á hora de desenvolver este proxecto. Nas seccións deste capítulo farase unha explicación, o máis detallada posible, de cómo funciona o sistema, os seus compoñentes e a interacción entre eles.

4.1. Equipamento hardware e software

O desenvolvemento do sistema realizase nun ordenador con sistema operativo **Linux Debian 11**. Este dispositivo conta cun procesador *Intel Core i7-1185G7*, 16 GiB de RAM e unha tarxeta gráfica Nvidia 1650 Max-Q e 4 GiB de memoria. O entorno de execución empregado é un servidor *Linux* aloxado no CiTIUS, que conta cun proceasdor *Intel Xeon Gold 5220*, 190 GiB de RAM e dúas tarxetas gráficas Nvidia Tesla V100S de 32 GiB de memoria cada unha.

A programación realízase empregando o IDE **PyCharm** [12], para o código de Python, e o editor **Visual Studio Code** [13] para a programación do cliente e do servidor web.

A api do sistema realízase empregando **TypeScript** [6], coa axuda de librerías como *Express* [27] para a creación da api HTTP, *Yup* [28] para a validación das peticións e *Mongodb* [29] para a conexión co driver da base de datos.

Para o cliente web, emprégase a librería **React** [7] xunto coas librerías *Validator* [30], para a comprobación de formularios, *Recharts* [31] para as gráficas dos resultados e *Tailwind* [32] para o estilo das páxinas.

4.1.1. Ferramentas e tecnoloxías empregadas

Para o correcto desenvolvemento do proxecto, incluíndo tanto a parte software como a documentación, empréganse as seguintes ferramentas:

- **Linux Debian 11**: o desenvolvemento do proxecto é realizado nun sistema operativo *Linux* coa distribución *Debian 11*.

- **Windows 10**: as tarefas de documentación e xeración de gráficos necesarios para as mesmas son realizados nun sistema operativo *Windows 10*.
- **Python**: o desenvolvemento da parte relativa á execución dos algoritmos de QA realízase con esta linguaxe de programación posto que os algoritmos cos que se proban están implementados en dita linguaxe [2].
- **MongoDB**: base de datos NoSQL empregada para almacenar a información necesaria polo sistema [3].
- **Node.js**: entorno de execución de JavaScript que permite tanto a execución da API REST do sistema como do cliente web [4].
- **Typescript**: linguaxe de programación coa que se desenvolve a API REST do sistema. Escóllese polas ventaxas que ofrece sobre a linguaxe JavaScript [5], como son o tipado de datos estático e unha mellor depuración do código [6].
- **React**: librería empregada para o desenvolvemento do cliente web do sistema [7].
- **Git**: ferramenta empregada para o control de versións do proxecto software [8].
- **GitHub**: ferramenta empregada para aloxar o repositorio Git [9].
- **Bash**: intérprete de ordes no que se implementan os módulos de execución dos algoritmos e entornos de execión [10].
- **Conda**: ferramenta empregada para a xestión e creación de entornos de execución dinámicos para a linguaxe Python. Emprégase para poder executar os algoritmos no sistema [11].
- **PyCharm**: entorno de desenvolvemento integrado empregado para a programación na linguaxe Python [12].
- **Visual Studio Code**: editor de textos empregado para a programación da API REST e do cliente web, así como dos scripts en Bash [13].
- **Overleaf**: editor de LaTex na nube no que se realiza a documentación do proxecto [14].
- **Draw.io**: editor de diagramas empregado para realizar as figuras e gráficos presentes na documentación do proxecto [16].
- **Adobe XD**: programa de deseño empregado para a maquetación dos deseños da interface de usuario [17].

4.2. Deseño do sistema

Neste apartado recóllese a estrutura do sistema e as decisións de deseño tomadas. Farase unha análise xeral dos compoñentes do sistema e despois realizarase unha análise ascendente de cada unha das partes que o conforman.

4.2.1. Deseño da arquitectura do sistema

O obxectivo da arquitectura proposta é deseñar un sistema que permita unha execución de tarefas nun servidor remoto e que ao mesmo tempo sexa escalable e flexible a introducir novos cambios e melloras. Para isto, deseñase un sistema onde se diferencian tres partes fundamentais e independentes: o servidor ou API REST, o cliente web e os workers de execución.

A separación destas tres partes do sistema permiten adaptar cada unha delas ás necesidades que se teñan. Así, podemos modificar o cliente web sen afectar á funcionalidade do servidor ou escalar os recursos físicos asociados a cada parte, podendo asignar os workers a máquinas máis potentes para a execución das tarefas e ter unha máquina máis modesta para o cliente web, por exemplo. Ademais, ao ter separado servidor e cliente, se nalgún momento se desexa realizar un programa con interface por comandos que interactúe co sistema ou integrar un sistema externo, poderase facer sen ter que modificar ou adaptar o servidor. O sistema está preparado para soportar futuras novas ferramentas e funcionalidades sen deixar de ser compatible co estado actual.

Arquitectura do sistema

O sistema deséñase seguindo unha arquitectura orientada a servizos. O diagrama de sistema pódese ver na Figura 4.1. Os compoñentes do sistema están acotados dentro da sección delimitada pola liña continua exterior. A plataforma STAC [20] é unha dependencia externa ao sistema, da que só se emprega a súa funcionalidade. O sistema divídese en tres partes principais: o cliente web, a API REST e os elementos (wrokers) de execución. A través das frechas, indícase a dependencia entre compoñentes internos, sendo o compoñente apuntado pola frecha o elemento do que se depende e do que se fai uso da súa funcionalidade.

A peza principal da arquitectura é un servidor ou API REST, formado por un conxunto de servizos que ofrecen a funcionalidade do sistema. O cliente fai uso desta funcionalidade chamando a través dunha comunicación HTTP aos métodos destes servizos. Finalmente, os encargados de executar as tarefas de adestramento e avaliación dos algoritmos son uns compoñentes chamados elementos ou "workers", que son dous scripts que crean os entornos, xestionan as dependencias e realizan as invocacións dos algoritmos.

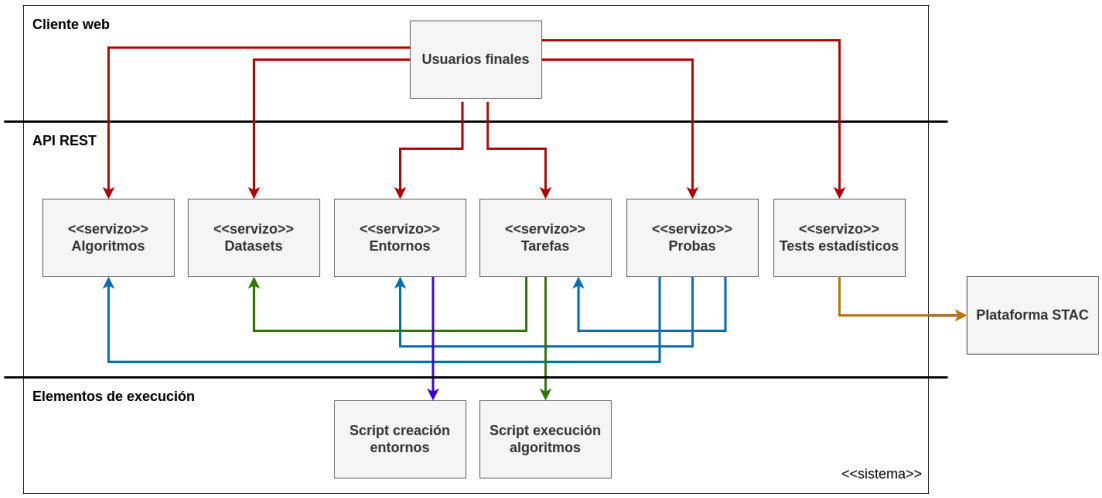


Figura 4.1: Diagrama do sistema.

Patrón de comunicación entre compoñentes

A comunicación entre o cliente web e o servidor realízase a través do protocolo HTTP. Todas as chamadas están baseadas nos verbos HTTP [26] e, como o servidor non almacena información dos clientes, é preciso que cada petición conteña toda a información necesaria para ser procesada correctamente polo servidor. Isto é o que se coñece como interface ou API REST.

Existe outro tipo de comunicación, que é a que se dá entre o servidor e os workers de execución. Como os workers de execución poden estar noutra máquina física distinta ao servidor, a invocación dos mesmos realízase a través do protocolo SSH [24], que permite a conexión segura de forma remota.

Patróns de deseño empregados

Como xa se comentou anteriormente, o patrón de deseño do sistema é unha **arquitectura orientada a servizos**. No que respecta aos compoñentes deste sistema, aplícanse unha serie de patróns de deseño que nos permiten resolver determinados problemas relativos á interación ou ás interfaces.

No cliente web emprégase unha clase que unifica a comunicación co servidor. Para isto, impleméntase o patrón creacional **instancia única** (singleton), que nos permite compartir a mesma instancia da clase e ter un acceso global a ela.

No servidor faise uso do patrón estrutural **ponte** (bridge), co que podemos desacoplar a implementación dos repositorios de acceso á base de datos da súa definición nas interfaces, que establecen o contrato coa funcionalidade requerida.

Emprégase o patrón **proxy** no acceso aos métodos da base de datos, realizando a invocación dende os servizos e proporcionando un intermediario para os controladores. Tamén se emprega este patrón no acceso aos métodos do servidor

dende o cliente, onde as chamadas, no lugar de realizarse de forma directa dende os compoñentes, realízanse a través da clase responsable da comunicación.

Finalmente, o deseño do servidor realízase mediante un deseño guiado polo dominio, onde se implementan os módulos do sistema baseados nos compoñentes e funcionalidades básicas. Cada módulo do servidor ten unha estrutura de capas, cun núcleo principal que modela o dominio. Para interactuar con este modelo, empréganse os servizos, que xestionan a lóxica e os accesos aos repositorios. Na capa máis externa atópanse as implementacións dos repositorios e os controladores. Con este deseño, mantemos o dominio da aplicación como base da funcionalidade e o resto de elementos simplemente permiten interactuar con él sen crear dependencias innecesarias.

4.2.2. Deseño da estrutura dos algoritmos

Implementación dos algoritmos

O primeiro aspecto a ter en conta ao deseñar o sistema é a problemática de executar algoritmos con deseños e arquitecturas totalmente distintas. Co obxectivo de estandarizar a estrutura dos algoritmos, deséñase unha clase abstracta que os algoritmos deben implementar [33]. Desta forma, todos os algoritmos teñen tres métodos comúns e a invocación a cada un dos métodos, por parte do sistema, é posible. Estes métodos son os seguintes: *setup_default_args*, *train* e *evaluate*. A continuación, podemos ver a clase abstracta:

```
class AbstractAlgorithm:
    __metaclass__ = ABCMeta

    def __init__(self, args, progress_callback):
        default_args = self.setup_default_args()
        self.args = merge_args(default_args, args)
        self.progress_callback = progress_callback

    @abstractmethod
    def setup_default_args(self):
        pass

    @abstractmethod
    def train(self):
        pass

    @abstractmethod
    def evaluate(self, evaluation_file_path):
        pass
```

O primeiro método, *setup_default_args*, permite que un algoritmo devolva un dicionario cos seus parámetros e valores por defecto. Estes parámetros pódense sobreescribir cos valores que aporte o usuario á hora de executar o algoritmo. Isto realízase no constructor da clase.

O método *train* é invocado á hora de adestrar un algoritmo, e a implementación interna queda da man do propio algoritmo para que realice as accións necesarias para cada caso.

O método *evaluate* é invocado cando se realiza unha avaliación. Ten un parámetro que corresponde ao ficheiro do dataset que se está a avaliar. O algoritmo debe implementar a lóxica necesaria para obter os resultados e devolver un dicionario coas respostas.

Adicionalmente, o algoritmo recibe unha función no constructor que permite que o algoritmo a invoque para notificar do seu progreso, enviando o paso actual e o total de pasos que vai realizar. Isto permítelle ao sistema controlar o progreso da execución do algoritmo en caso de que sexa necesario.

No caso deste proxecto, para poder probar a execución dalgúns algoritmos de QA, realízase a adaptación de dous algoritmos: **Electra** [34, 35] e **Albert** [36]. Para isto, pártese da implementación realizada nos repositorios dos citados algoritmos. Para cada un deles, procédese a adaptar o código dispoñible aos métodos do algoritmo abstracto definido. Isto conséguese movendo o código de inicialización ao constructor e o código de adestramento e avaliación aos respectivos métodos. Tamén se crea un terceiro algoritmo, empregado para as probas, que simplemente implementa os métodos e simula os resultados, o que permite unha execución máis rápida.

Configuración dos algoritmos

O segundo aspecto a ter en conta é a necesidade dunha estrutura común para os algoritmos subidos. Para poder invocalos, estes algoritmos teñen que ser módulos de Python, podendo así importalos estando nunha carpeta concreta. Para isto, cada algoritmo, ademais de implementar a clase abstracta, debe estar contido nun ficheiro comprimido onde, ademais do código que implementa dita clase, tamén haxa un ficheiro *__init__.py*, que define un módulo de Python, e un ficheiro de configuración chamado *config.json*. Tamén poderá haber outros ficheiros, carpetas e elementos necesarios dos que o algoritmo faga uso. Podemos ver na Figura 4.2 a estrutura de subida dun algoritmo.

O ficheiro de configuración define a información básica dun algoritmo (como o nome, nome do ficheiro que implementa o algoritmo e nome da clase que o implementa), información sobre o entorno de execución (versión de Python necesaria, librerías das que depende e comandos necesarios para a súa instalación) e unha lista de posibles parámetros que pode recibir o algoritmo. Con esta información, o sistema é capaz de crear os entornos de execución coas dependencias necesarias así como de suministrar aqueles parámetros necesarios para cada caso.

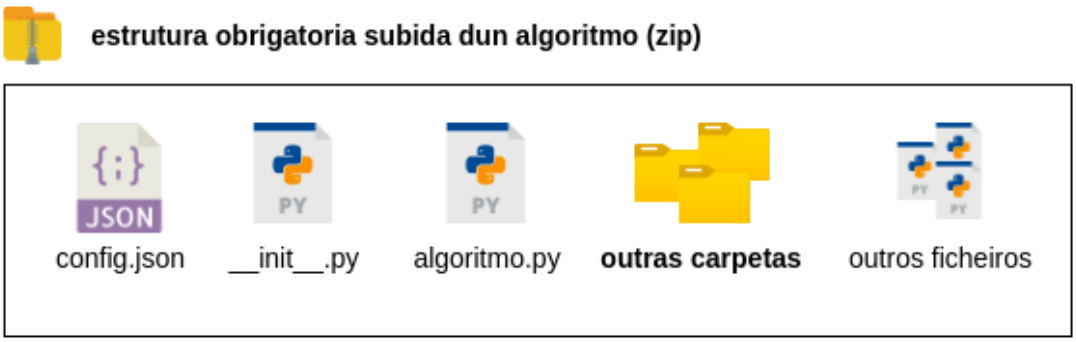


Figura 4.2: Estrutura do ficheiro *ZIP* dun algoritmo.

O ficheiro de configuración terá un formato *JSON* coas seguintes entradas:

- **algorithm_name**: Nome do algoritmo
- **algorithm_class_file**: Nome do ficheiro onde se atopa a clase que implementa o algoritmo
- **algorithm_class_name**: Nome da clase que implementa o algoritmo
- **environment**: Un *JSON* cos campos *python_version*, que poderá ser calquera versión compatible con Conda [11], *dependencies*, que será unha lista de *JSONs* con entradas *lib* (nome da librería) e *version* (versión necesaria ou *null* para a última), e *extra_python_commands*, que será unha lista de strings que definan comandos necesarios que haxa que executar cando se executa o algoritmo (por exemplo: "spacy download en_core_web_lg").
- **args**: Lista de parámetros que admite o algoritmo. Cada entrada terá a seguinte información:
  - **name**: Nome do parámetro (por exemplo: "–learning_rate").
  - **required**: *true* se é obrigatorio para a execución do algoritmo ou *false* se non. O seu valor a *true* anula calquera valor dos campos *for_train* e *for_evaluation*.
  - **for_train**: *true* se se emprega para o adestramento ou *false* se non.
  - **for_evaluation**: *true* se se emprega para a avaliación ou *false* se non.
  - **type**: Tipo de dato do parámetro (string, float ou int)
  - **default**: Valor por defecto ou *null* se non se quere incluir.
  - **choices**: Lista de valores posibles ou *null* se non se quere incluir.
  - **help**: Información para o parámetro ou *null* se non se quere incluir.
  - **is_path**: *true* se é un path a un ficheiro ou directorio, *false* noutro caso.

4.2.3. Deseño dos workers de execución

Unha vez definida a estrutura común que van ter todos os algoritmos que se executan na plataforma, procedemos a definir a forma na que estes algoritmos se van executar.

En primeiro lugar, hai que ter en conta se o algoritmo se adestrou algunha vez no sistema ou non. Para isto lévase un rexistro dos algoritmos executados e dos seus parámetros de adestramento, de modo que se se produce algún cambio, podemos crear un entorno novo, realizar un novo adestramento e manter as versións dos adestramentos previas, evitando así ter que adestrar cada vez que se realiza unha execución. Cando se crea un **entorno**, que é un espazo illado onde cada algoritmo ten todo o necesario para poder executarse, o encargado de crear a estrutura de carpetas e ficheiros necesarios é un script escrito en *bash* [10] (*prepare_environment.sh*). Esta estrutura inicial está formada por un directorio co nome do identificador do entorno e no seu interior terá unha carpeta chamada *algorithm* co código do algoritmo (todo o contido do ficheiro comprimido subido á plataforma) e unha serie de scripts escritos en *Python* [2], que permiten realizar a invocación ao propio algoritmo.

En segundo lugar, precisamos executar o algoritmo e hai que ter en conta a versión de *Python* empregada e as dependencias asociadas ao mesmo. Para isto, empregamos a ferramenta *Conda* [11], que nos permite crear espazos illados de execución de *Python*, onde se van usar as versións e dependencias indicadas só para ese espazo. Desta maneira conseguimos o illamento e interoperabilidade buscados á hora de executar calquera tipo de algoritmo na plataforma.

O encargado de xestionar *Conda* e invocar ao propio algoritmo é outro script escrito en *bash* (*execute_conda_environment.sh*). As funcións deste script son as seguintes:

- **Paso 1**: Crear un entorno *Conda* coa versión de *Python axeitada*. Esta información sácase do ficheiro de configuración.
- **Paso 2**: Activar o entorno *Conda* para que, á hora de executar calquera orde *Python*, se empreguen as dependencias do entorno actual.
- **Paso 3**: Instalar as dependencias (librerías) indicadas no ficheiro de configuración. A xestión destas dependencias tamén a realiza *Conda*.
- **Paso 4**: Executar os comandos adicionais necesarios definidos no ficheiro de configuración.
- **Paso 5**: Invocar o algoritmo a través do script de invocación escrito en *Python*.

Cabe destacar que é a este script de execución ao que lle pasamos como parámetros todos os valores necesarios para a invocación do algoritmo, e é o

propio script o encargado de pasarllos á hora de invocalo. Tamén lle indicamos se queremos que realice un adestramento ou unha avaliación do algoritmo.

O acceso aos ficheiros comprimidos dos algoritmos e aos datasets realízase indicando as rutas destes, polo que deben atoparse na mesma máquina que os scripts e ser accesibles polos mesmos.

O feito de que cada execución destes scripts teña toda a información necesaria para executarse e funcionar de forma independente como un proceso na máquina de execución, é o motivo polo que se lle chama a estes scripts os **workers de execución**, pois serán procesos invocados e independentes, coa capacidade de realizar a tarefa asociada.

A estrutura final do entorno do algoritmo pódese observar na Figura 4.3. Dentro da estrutura mencionada, podemos apreciar unha serie de directorios e ficheiros que se detallan a continuación:

- **algorithm/**: este directorio contén o código do algoritmo.
- **env/** este directorio contén as dependencias e recursos necesarios para a execución do entorno *Conda*. É xestionado pola propia ferramenta de forma independente.
- **abstract_algorithm.py**: Contén a clase abstracta que implementa o algoritmo.
- **evaluate_v2.py**: Contén os métodos de avaliación dos resultados para *SQuAD 2.0*. É invocado por *main.py* durante as avaliacións para poder obter as métricas de rendemento.
- **generate_requirements.py**: Script que xera os ficheiros coas dependencias necesarias para crear o entorno *Conda*.
- **__init__.py**: Ficheiro para poder crear o módulo *Python* e importar o algoritmo.
- **utils.py**: Contén utilidades de axuda empregadas por *main.py*.
- **main.py**: Contén a lóxica necesaria para procesar os parámetros do algoritmo, importar a clase do algoritmo e invocar o método *train* ou *evaluate* en función da tarefa asignada. Tamén contén a lóxica correspondente a notificar a finalización da execución ao servidor. Isto realízase facendo unha chamada HTTP cando remata e enviando os resultados como datos da chamada. Este ficheiro é o encargado de toda a execución dos algoritmos e é invocado polo script *execute_conda_environment.sh*.
- **extra_python_commands.txt**: Ficheiro xerado por *generate_requirements.py* que contén a lista de comandos adicionais que debe executar o script *execute_conda_environment.sh*.

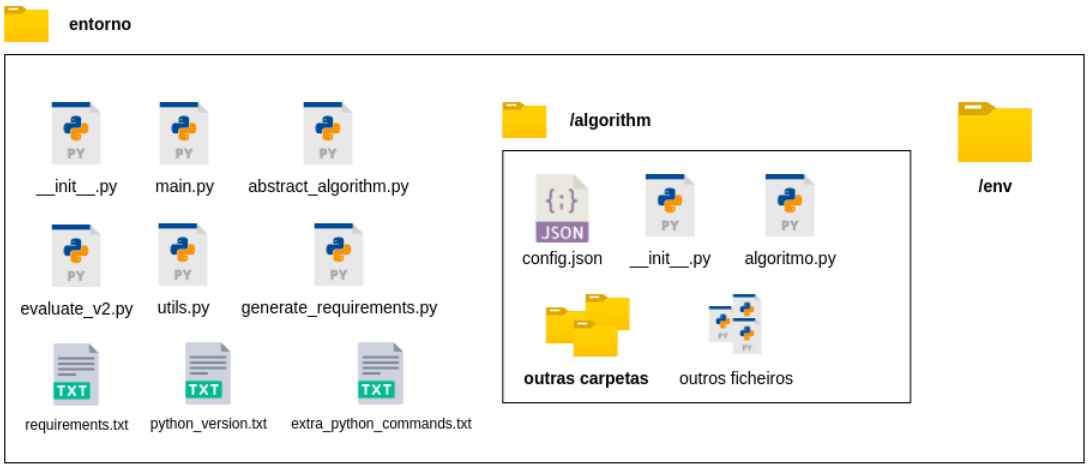


Figura 4.3: Estrutura dun entorno de execución.

- **python_version.txt**: Ficheiro xerado por *generate_requirements.py* que contén a versión de *Python* coa que se debe crear o entorno *Conda*.

- **requirements.txt**: Ficheiro xerado por *generate_requirements.py* que contén a lista de dependencias que se deben instalar no entorno *Conda* mediante o script *execute_conda_environment.sh*.

Finalmente, móstrase na Figura 4.4 o proceso seguido durante a execución dun algoritmo.

4.2.4. Deseño da arquitectura do servidor

O servidor do sistema é o encargado de proporcinar a funcionalidade ao cliente web a través dun servidor HTTP. Permite a creación e xestión das probas, algoritmos e datasets do sistema.

O servidor está implementado en *Typescript* [6] e para a parte da API HTTP emprégase a librería *Express* [27]. Esta librería permítenos declarar as rutas necesarias así como xestionar as peticións e o procesado das mesmas. Para a validación das peticións emprégase a librería *Yup* [28], que nos permite definir a estrutura necesaria para os datos requeridos en cada unha das rutas.

O servidor emprega unha base de datos non relacional. En concreto, escolleuse *MongoDB* [3] debido a que nos permite gardar os datos como documentos, ademais de darnos a flexibilidade de non ter que tratar cun esquema predefinido e poder iterar rapidamente. Tamén se adapta de maneira natural ao datos cos que imos tratar, pois a aplicación emprega obxectos JavaScript [5], moi similares aos documentos, e os algoritmos teñen unha serie de parámetros e información que tamén se proporcionan como *JSONs*.

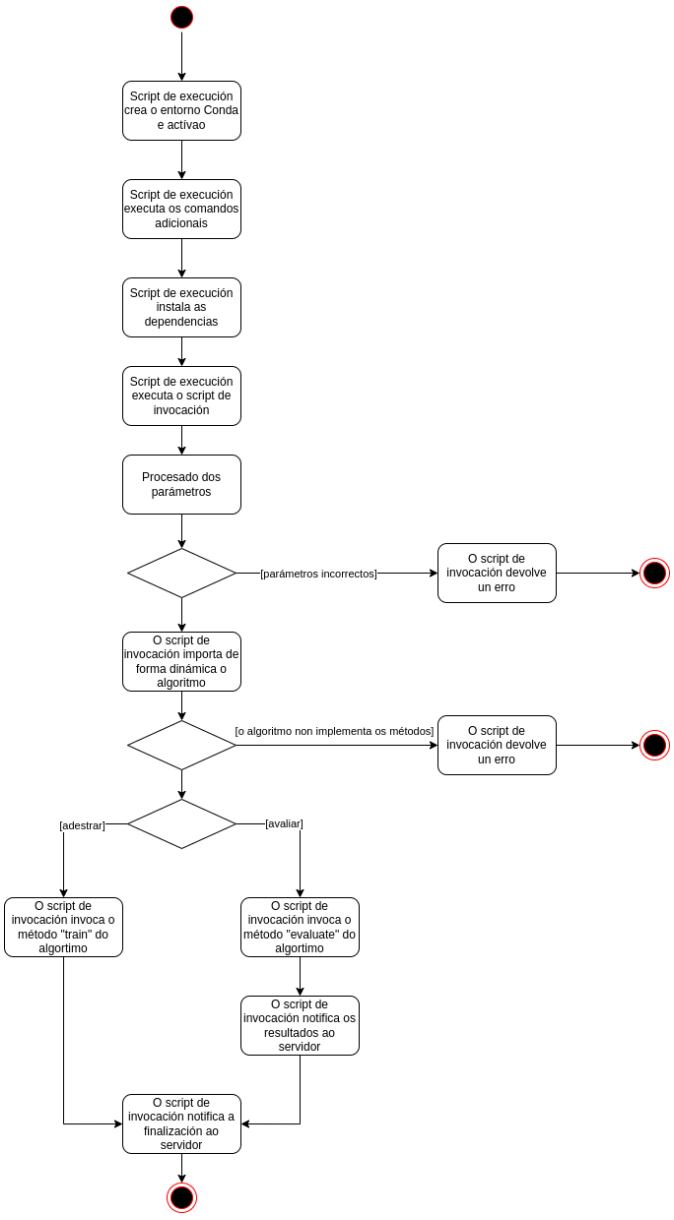


Figura 4.4: Diagrama de actividade da execución dun algoritmo.

Arquitectura

A arquitectura escollida para o servidor é unha arquitectura orientada a servizos. A característica principal deste tipo de arquitecturas é a modularidade e independecia de cada un dos servizos, de forma que cada un se encarga dunha funcionalidade concreta e ben definida, e o seu funcionamento non está fortemente ligado ao resto de servizos.

O servidor consta de seis servizos, que son os seguintes:

- **Servizo de algoritmos**: este servizo é o responsable de proporcionar a funcionalidade relacionada coa creación e consulta dos algoritmos do sistema.
- **Servizo de datasets**: este servizo é o responsable de proporcionar a funcionalidade relacionada coa creación e consulta dos datasets do sistema.
- **Servizo de entornos**: este servizo é o responsable de xestionar os entornos de execución dos algoritmos. Isto inclúe as tarefas de comprobar se un algoritmo está adestrado ou non para determinados parámetros e de crear os entornos cando sexa preciso.
- **Servizo de tarefas**: este servizo é o responsable de xestionar a invocación das tarefas de adestramento e avaliación, así como da xestión da finalización e notificación dos resultados das mesmas.
- **Servizo de probas**: este é o servizo principal do sistema e engloba a funcionalidade asociada á creación e consulta das probas.
- **Servizo de tests estadísticos**: este servizo é o responsable de calcular os tests estadísticos das probas.

Dentro de cada servizo, dispoñemos dunha arquitectura limpa onde se estrutura a funcionalidade da seguinte maneira: un controlador recibe e xestiona as peticións HTTP, o controlador invoca os métodos do servizo, que xestiona a lóxica correspondente a cada caso de uso do sistema, e este servizo fai uso ao seu mesmo tempo dun repositorio, que xestiona o acceso e a persistencia dos datos na base de datos. Ademais, destacar que os repositorios están definidos como interfaces que son implementadas por clases concretas, o que non só nos permite definir mellor a funcionalidade necesaria, senón que tamén nos permite implementar ditas interfaces como mocks á hora de realizar as probas dos servizos. Cada servizo xestiona as súas propias rutas, e estas defínense como se indica no Cadro A.1 do Apéndice A.4.

A funcionalidade global do sistema depende da coordinación dos servizos para realizar as tarefas necesarias. Isto provoca que aparezan dependencias entre os servizos, que se indican a continuación:

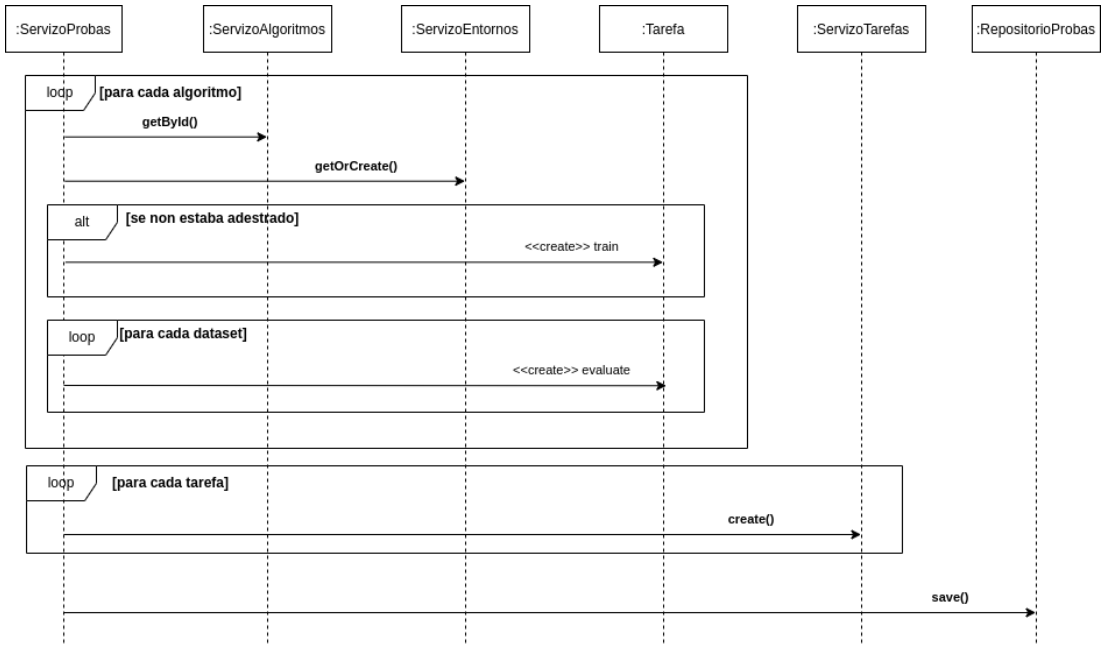


Figura 4.5: Diagrama de secuencia da creación dunha proba.

- Os servizos de **algoritmos**, **datasets**, **entornos** e **tests estadísticos** non teñen dependencias por ser unidades fundamentais do sistema.
- O servizo de **tarefas** depende do servizo de datasets (para recuperar información) e do repositorio das probas (para gardar os resultados das execucións). A dependencia sobre o as probas faise sobre o repositorio e non sobre o servizo para evitar dependencias circulares.
- O servizo de **probas** depende do servizo de algoritmos (para recuperar información), do servizo de entornos (para crear os espazos de execución) e do servizo de tarefas (para invocar a execución dos adestramentos e avaliacións).

Finalmente, ilústrase na Figura 4.5 o diagrama de secuencia da creación dunha proba, mostrando as distintas interaccións entre os servizos na execución do caso de uso. Só se mostran as chamádas a métodos, sen os retornos, para simplificar o diagrama.

Execución de tarefas

O servidor é o encargado de invocador os *workers de execución*, que nos permiten crear os entornos e executar os algoritmos. A invocación a estes scripts realízase cunha chamada á API nativa do sistema operativo.

No caso da creación dos entornos, a execución é síncrona, pois é unha operación rápida e pódese asumir o coste temporal.

Para as execucións dos algoritmos, como poden implicar tempos de execución moi longos, créase un proceso que executa ese script en segundo plano, de forma que a súa execución non sexa bloqueante e permita ao servidor seguir estando dispoñible para recibir outras peticións.

A prioridade das execucións impleméntase mediante unha cola *FIFO* [37], onde se van executando as tarefas en orde de chegada. A base de datos garda unha lista de tarefas e o seu estado, e cando unha tarefa remata, escollese a seguinte e invócase. Cada tarefa ten os datos necesarios para ser executada: o entorno onde ten que ser executada, a acción (adestrar ou avaliar), o dataset (en caso de ser unha avaliación), e os parámetros que se van aplicar ao algoritmo en cuestión.

Tests estadísticos

Os tests estadísticos permiten comparar o rendemento dun conxunto de algoritmos sobre unha serie de datasets. O servidor pode procesar o resultado dun conxunto de probas e calcular o test estadístico máis axeitado a cada caso. Este cálculo realízase chamando á plataforma **STAC** [20] a través dunha chamada HTTP.

O test estadístico aplicado escóllese en función dos seguintes criterios:

- Se o número de algoritmos é 2 e o número de datasets é par, aplícase o test **Wilcoxon**.
- Se o número de algoritmos é 2 e o número de datasets é impar, aplícase o test **Mann Whitney U**.
- Se o número de algoritmos é maior que 4 e o número de datasets é igual ou maior que o doble de algoritmos, aplícase o test **Friedman**.
- Se o número de algoritmos é maior que 2 e menor que 5 ou o número de datasets é menor que o doble de algoritmos, aplícase o test **Aligned Ranks**.

Estes criterios baséanse nas recomendacións indicadas na propia plataforma STAC [20].

Execución en remoto

Como a execución dos algoritmos se realiza nunha máquina diferente daquela na que se atopa o servidor, é preciso invocar os *workers de execución* de forma remota. Para isto, emprégase o protocolo SSH, que nos permite o acceso remoto a unha máquina e a execución dos scripts necesarios [24].

Debemos ter en conta que os scripts que executan os algoritmos deben ter acceso tanto aos datasets como ao código dos algoritmos. Isto implica que cada vez que se crea un novo dataset ou algoritmo, debemos enviar ese ficheiro ao

servidor de execución. Para isto empregamos o protocolo SCP, co cal podemos enviar ficheiros a unha máquina remota de forma segura [25].

Seguridade

A execución en remoto e o uso de SSH e SCP implican que hai que xestionar unha serie de credenciais que nos permitan acceder á máquina remota.

Para garantir a seguridade e confidencialidade das credenciais, decídese optar polo uso dun par de chaves de tipo chave pública-privada. Isto implica que o servidor remoto debe coñecer a chave pública que empreguemos, pero deste xeito conseguimos que non faga falta enviar ningún tipo de credencial a través da rede nin almacenar as credenciar en texto plano, simplemente debemos gardar o ficheiro coa chave privada.

4.2.5. Deseño da arquitectura do cliente web

O cliente web consiste nun proxecto programado coa librería *React* [7]. Esta librería permite crear unha aplicación web con rutas dinámicas, renderizado do lado do cliente, reducindo así a carga de traballo do servidor, e un desenvolvemento rápido grazas ao sistema de compoñentes que ten incorporado, podendo así definir tanto lóxica, estrutura e estilos reutilizables en toda a aplicación.

A comunicación co servidor realízase a través do protocolo HTTP e chamadas a través de *fetch*, a API nativa do navegador para este tivo de comunicacións.

Para os estilos CSS emprégase a librería *Tailwind* [32], pois evita ter que reescribir moito código. Destacar tamén que se emprega *Validator* [30] para as validacións dos parámetros dos algoritmos, *React Json View* [38] para mostrar a vista previa dos datasets e *Recharts* [31] para as gráficas dos resultados.

Para crear as rutas no cliente, emprégase a librería *React Router DOM* [39]. Isto permítenos ter rutas dinámicas e non temos a necesidade de recargar a páxina cada vez que navegamos, polo que a carga é máis rápida. As rutas do cliente web son as amosadas no Cadro A.2 do Apéndice A.4.

Na Figura 4.6 móstrase a estrutura básica do cliente web. A aplicación engloba a xestión das rutas dentro do *router*. Cada ruta correspóndese cunha páxina ou compoñente cuxa función é representar os datos e agrupar compoñentes funcionais. Os compoñentes funcionais xuntan tanto a representación visual da aplicación como a xestión do estado e a funcionalidade, resultado da interación co usuario. Os compoñentes poden facer uso dos hooks, que xestionan un estado común e reutilizable. Estes hooks invocan os méotodos dos servizos e utilidades da aplicación. Os servizos son os encargados da itneración cos sistemas externos, como o servidor do sistema. Con este deseño conseguimos unha aplicación web estruturada en módulos que agrupan funcionalidades concretas e construída a través de elementos comúns e reutilizables.

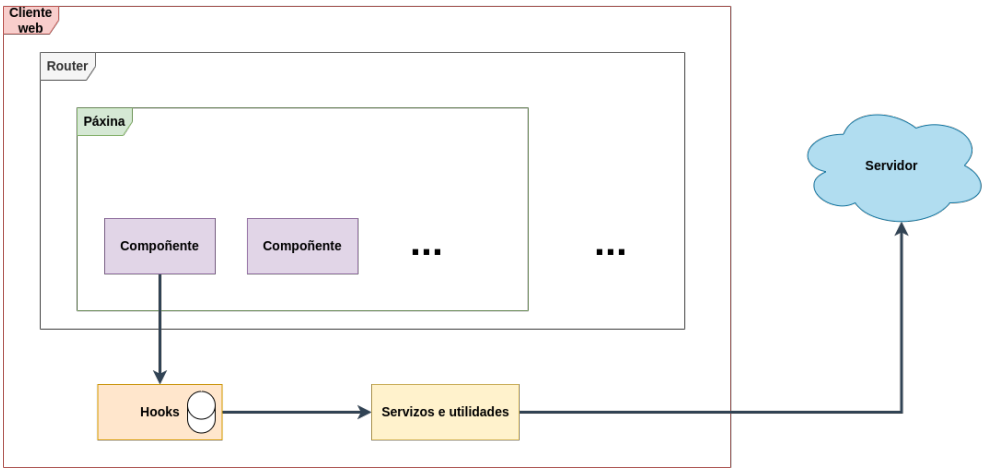


Figura 4.6: Estrutura do cliente web.

4.3. Deseño das interfaces de usuario

O cliente web deséñase para consumir a funcionalidade do sistema e garantir un nivel de usabilidade que permita aos usurios ter unha boa exeperiencia de uso. Ademais, búscase ofrecer un deseño que permita analizar e comprender a información amosada da forma máis sinxela e clara posible.

Neste apartado móstrase unha análise do deseño e decisións tomadas en cada unha das vistas do cliente web.

4.3.1. Vistas

Todas as pantallas, a execepción da pantalla principal, contan cun menú lateral que permite navegar entre todas as pantallas da plataforma. Todas elas comparten unha cabeceira común.

Pantalla principal

A pantalla principal é a que se mostra cando se accede á plataforma. Nesta pantalla podemos ver unha sección principal que actúa como menú para acceder ao resto de pantallas. Debaixo desta sección, aparecen dúas táboas: unha cunha lista de probas en execución e outra cunha lista de probas finalizadas.

Pantalla de datasets

A pantalla de datasets mostra na zona superior unha táboa coa lista de datasets rexistrados na plataforma. Nesta táboa aparecen datos como a data de subida, o nome do dataset, o nome do ficheiro e o tamaño do ficheiro. Premendo encima de cada unha das filas, asociada cada unha a un dataset, podemos acceder á pantalla de consulta do propio dataset.

Debaixo desta táboa temos unha zona onde podemos premer para seleccionar un ficheiro, ou simplemente soltalo encima. Isto permite subir un dataset e asignarlle un nome a través dun campo de texto. Tamén se dispón dun botón para confirmar a subida e outro para cancelar a operación. No caso de que se produza un erro, aparecerá unha mensaxe de cor vermello cun texto explicativo sobre o que ocorreu.

Pantalla de consulta dun dataset

A pantalla de consulta dun dataset está formada por tres seccións. A primeira sección mostra a información asociada ao dataset. A segunda sección mostra a temática das cuestións do dataset como unha lista de temas. Finalmente, a terceira sección mostra unha vista previa do contido do ficheiro, a modo de resumo.

Pantalla de algoritmos

A pantalla de algoritmos está formada por unha táboa coa lista de algoritmos rexistrados no sistema, amosando información como a data de subida e nome do algoritmo. Premendo encima de cada unha das filas podemos acceder á pantalla do algoritmo seleccionado.

Debaixo da táboa de algoritmos, temos unha zona que, igual que no caso dos datasets, permite seleccionar un ficheiro comprimido e subir un algoritmo. Ao seleccionar o ficheiro, móstrase un resumo dos datos do mesmo, con información do ficheiro de configuración contido nel, e permítese confirmar a subida premendo nun botón situado inmediatamente debaixo. Tamén aparece un botón para cancelar a operación e unha mensaxe de erro en cor vermello en caso de que a operación falle.

Pantalla de consulta dun algoritmo

A pantalla de consulta dun algoritmo mostra toda a información contida no ficheiro de configuración dun algoritmo. Esta pantalla permite consultar o nome, data de subida, dependencias e parámetros do algoritmo. Os parámetros móstranse nunha táboa onde cada fila representa un parámetro e cada columna os atributos posibles xunto co seu valor.

Pantalla do historial de execución

A pantalla do historial de execución de probas mostra unha táboa coas probas rematadas. Para cada proba, móstrase a data, o nome e o número total de algoritmos e datasets. Premendo sobre unha fila, podemos acceder á pantalla da proba asociada.

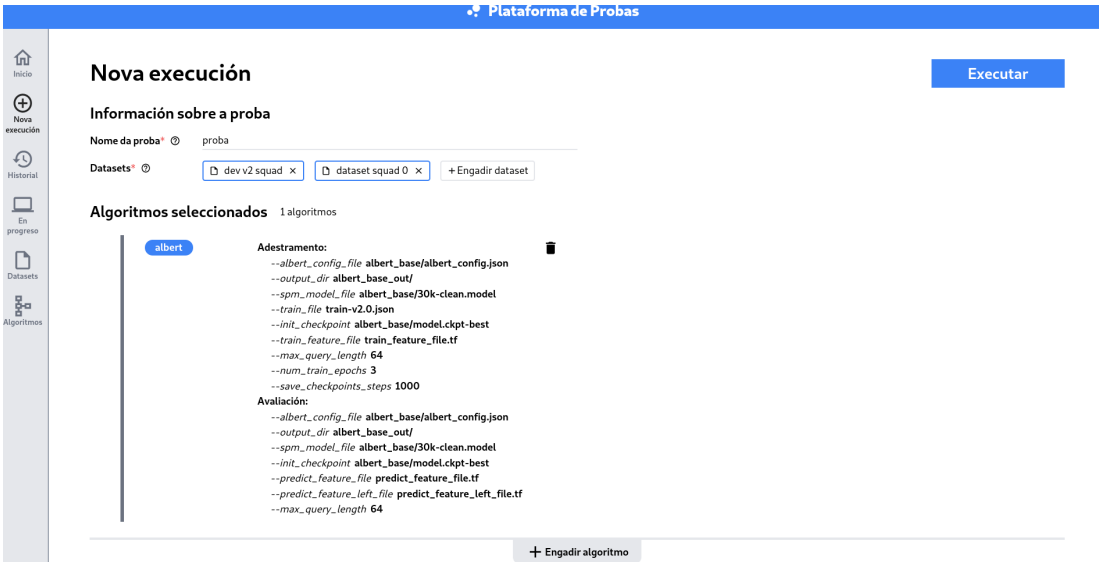


Figura 4.7: Pantalla de creación dunha proba.

Pantalla de probas en execución

A pantalla de probas en execución mostra unha táboa coas probas que aínda non remataron. Para cada proba, móstrase a data, o nome, o número total de algoritmos e datasets e a porcentaxe de progreso. Premendo sobre unha fila, podemos acceder á pantalla da proba asociada.

Pantalla de creación dunha proba

A pantalla de creación dunha proba permite introducir o nome da mesma a través dun campo de texto.

Debaixo deste campo, podemos seleccionar unha serie de datasets premendo nun botón que abre unha lista con todos os datasets dispoñibles. Ao premer nun, engádese á proba e móstrase como seleccionado.

Na zona inferior, dispoñemos dun botón que abre un modal onde podemos seleccionar e configurar os algoritmos que desexamos probar. Este modal está formado por tres vistas: a primeira permite seleccionar un algoritmo e as outras dúas permiten asignar os valores desexados aos parámetros de adestramento e avaliación do mesmo. Para estes parámetros, móstrase un campo de texto no caso de ser un valor de libre elección, ou un desplegable no caso de admitir só valores concretos. No caso de haber algún erro de validación de datos dos parámetros, móstrase un erro en cor vermella. Ao rematar a configuración do algoritmo, péchase o modal e móstrase unha lista cos algoritmos seleccionados e os parámetros que se lle asignaron a cada un.

Podemos ver unha representación desta pantalla na Figura 4.7 e do modal de configuración dun algoritmo na Figura 4.8.

Figura 4.8: Modal de configuración dun algoritmo para unha proba.

Pantalla de consulta dunha proba

A pantalla de consulta dunha proba ten dúas variantes: unha no caso de non ter rematado aínda e outra no caso de ter rematado.

Se a proba se atopa en execución (aínda non rematou), móstrase o nome, a lista de datasets e algoritmos e a lista de tarefas asociadas á mesma. Nesta lista de tarefas, indícase o estado (rematada, erro ou pendente) e a acción que se executou (adestramento dun algoritmo ou avaliación dun algoritmo sobre un dataset).

Se a proba rematou, móstrase o nome da proba, a lista de datasets e unha zona na parte inferior cos resultados das probas. Nesta zona inferior, aparecen, na parte lateral esquerda, a lista de algoritmos executados. Podemos seleccionar estes algoritmos para mostrar os resultados de cada un deles.

Ao seleccionalos, aparece na parte dereita unha gráfica que permite agrupar os resultados por dataset ou por algoritmo. Esta gráfica mostra, mediante un histograma, os resultados de cada un dos algoritmos sobre os datasets, indicándose o valor *F1* obtido en cada execución.

Debaixo desta gráfica, atopamos dúas táboas. A primeira mostra os resultados *F1* dos algoritmos sobre cada un dos datasets e a segunda mostra os valores *EM* ou exactos das avaliacións.

Finalmente, na zona inferior aparece o test estadístico executado sobre os algoritmos seleccionados. Móstrase o nome do test (pois execútase o test máis axeitado en función do número de algoritmos e datasets) e permítese seleccionar

Figura 4.9: Gráfica de resultados dunha proba.

Test estadísticos - *Friedman Aligned Ranks - Holm*

Nivel de significancia 0.05

Ranking:
Null hypothesis (H_0): As medias dos resultados de dous ou máis algoritmos son as mesmas

Post-hoc:
Null hypothesis (H_0): A media dos resultados de cada par de grupos é a mesma

Friedman Aligned Ranks test (nivel de significancia de 0.05)

Statístico	p-value	Resultado
2.60241	0.27220	H_0 é aceptada

Ranking

Algoritmo	Ranking
mi-algoritmo-1	3
mi-algoritmo-2	5
mi-algoritmo-3	7

Comparación Friedman Aligned Ranks test (nivel de significancia de 0.05)

Comparación	Statístico	p-value axustado	Resultado
mi-algoritmo-2 vs mi-algoritmo-3	1.78885	0.22091	H_0 é aceptada
mi-algoritmo-2 vs mi-algoritmo-1	0.89443	0.74219	H_0 é aceptada
mi-algoritmo-3 vs mi-algoritmo-1	0.89443	0.74219	H_0 é aceptada

Figura 4.10: Resultados dos tests estadísticos dunha proba.

o nivel de significancia (que axusta o grao de aceptación da hipótese aplicada no test). Do resultado do test, aparece en pantalla a explicación das hipóteses dos resultados e tres táboas co resultado obtido, o ranking dos algoritmos e a comparación por pares de algoritmos.

Podemos ver unha representación da gráfica de resultados na Figura 4.9 e dos tests estadísticos na Figura 4.10.

Capítulo 5

Probas

Esta sección recolle as probas realizadas, comprobando o sistema implementado e a súa conformidade respecto ao proxecto e obxectivos establecidos.

5.1. Usabilidade

5.1.1. Usabilidade mediante os heurísticos de Nielsen

O deseño das interfaces de usuario fíxose seguindo o cumprimento dos heurísticos de Nielsen [40], garantindo a usabilidade do sistema mediante accións que permitan que o usuario teña toda a información posible da forma máis clara posible.

A plataforma ofrece unha **visibilidade do estado do sistema** a través de elementos que permiten manter ao usuario informado, como son o caso dos indicadores de carga cando se executan operacións que poden consumir un tempo elevado (é o caso da subida dos ficheiros dos algoritmos ou datasets). A **relación entre a plataforma e o mundo real** consíguese empregando unha linguaxe familiar para o usuario, non empregando termos complexos na información amosada e aportado aclaracións en casos específicos. O usuario ten **liberdade e control** sobre as súas accións e pode rectificar a través dos botóns de cancelación das operacións da plataforma. A **consistencia** consíguese empregando unha estrutura común en toda a plataforma e tendo unha mesma estética en todos os elementos que realizan accións semellantes, como os botóns de execución de accións ou os de cancelar operacións. O usuario sempre dispón da información necesaria para favorecer o **recoñecemento** e estar ao tanto do estado das súas accións, como se pode ver na lista de algoritmos configurados para unha proba, onde o usuario non precisa recordar cales configurou ou con qué parámetros. A **estética e o deseño son minimalistas** e aseguran que se mostra a información xusta e necesaria, evitando elementos irrelevantes. As mensaxes de erro axudan ao usuario a **recoñecer** a situación producida e recuperarse deles, indicando accións a tomar e

a causa dos mesmos, como é o caso das mensaxes de erro nos datos introducidos para a configuración dun algoritmo, que lle indican ao usuario qué valores debe asignar e cales son obrigatorios. Tamén se evitan posibles erros derivados das accións do usuario mediante inhabilitacións de botóns antes de que a acción asociada se atope dispoñible, como no caso de comezar unha proba antes de ter escollidos os conxuntos de datos e os algoritmos. Finalmente, o usuario conta con indicadores de **axuda** en determinados elementos, como os campos de texto ou os elementos de configuración do modal de selección dun algoritmo, indicando cal é a finalidade de cada campo e proporcionando información do mesmo.

5.1.2. Usabilidade a través de probas e cuestionarios con usuarios

Observación dos usuarios

Durante a implementación do proxecto, mantéñense diversas revisións cos titores, dende o punto de vista dos usuarios, para recoller comentarios e información que permitise mellorar o deseño e usabilidade do sistema.

As probas de usabilidade realizadas consistiron na observación do usuario final empregando o sistema e analizando aqueles aspectos onde se atopaban máis problemas ou dificultades. Unha vez o usuario analizaba o sistema, realizábase un resumo dos puntos máis importantes onde se debía mellorar e aplicábanse accións de corrección sobre o deseño e a implementación da plataforma. Entre os aspectos mellorados destaca a pantalla de consulta dunha proba, que nun principio contiña demasiada información e estaba pouco estruturada. Tras a revisión cos usuarios, apréciase que non son capaces de atopar os datos que buscaban e non entendían a información que estaban vendo na pantalla, así que se decide mellorar a súa usabilidade incluíndo unha serie de táboas e gráficas que dan máis liberdade ao usuario e que lle permiten consultar a información dun xeito máis sinxelo.

Cuestionario SUS

Outra proba realizada cos usuarios finais foi a avaliación de escalas de usabilidade (SUS, do inglés System Usability Scale) a través dun cuestionario realizado sobre 5 usuarios, entre os que se atopan 2 profesores (os titores do traballo), 2 alumnos do Grao e 1 usuario sen relación coa informática. Escóllense estes tres perfís de usuario para ver cómo se comportan sobre un sistema onde poden non coñecer a funcionalidade ou finalidade do mesmo, apreciando así o grao de usabilidade dun xeito rápido e eficaz.

Os usuarios deben indicar o seu grao de conformidade cunha serie de enunciados para calcular o nivel de usabilidade. Estes enunciados están predefinidos e as respostas son anónimas. As posibles respostas que os usuarios poden indicar son as seguintes:

- *1. Totalmente en desacordo.*
- *2. En desacordo.*
- *3. Neutro.*
- *4. De acordo.*
- *5. Totalmente de acordo.*

A continuación, descríbense os enunciados que se lle amosan aos usuarios enquisados e as respostas destes (respostas e número de veces que se repite cada unha indicado entre parénteses):

- **Enunciado 1**: *Creo que me gustaría utilizar este sistema con frecuencia.*
  - ***Respostas***: Totalmente de acordo (3), De acordo (1), Neutro (1).
- **Enunciado 2**: *Atopei o sistema innecesariamente complexo.*
  - ***Respostas***: Totalmente en desacordo (4), En desacordo (1).
- **Enunciado 3**: *Pensei que o sistema era fácil de usar.*
  - ***Respostas***: Totalmente de acordo (5).
- **Enunciado 4**: *Creo que necesitaría o apoio dun técnico para poder utilizar este sistema.*
  - ***Respostas***: Totalmente en desacordo (5).
- **Enunciado 5**: *Atopei que as diversas funcións do sistema estaban ben integradas.*
  - ***Respostas***: Totalmente de acordo (5).
- **Enunciado 6**: *Pensei que había demasiada inconsistencia neste sistema.*
  - ***Respostas***: Totalmente en desacordo (5).
- **Enunciado 7**: *Imaxino que a maioría da xente aprendería a utilizar este sistema moi rápidamente.*
  - ***Respostas***: Totalmente de acordo (3), De acordo (2).
- **Enunciado 8**: *Atopei o sistema moi complicado de usar.*
  - ***Respostas***: Totalmente en desacordo (4), En desacordo (1).
- **Enunciado 9**: *Sentínme moi seguro usando o sistema.*

    - ***Respostas***: Totalmente de acordo (5).

- **Enunciado 10**: *Necesitaba aprender moitas cousas antes de empezar a utilizar este sistema.*

    - ***Respostas***: Totalmente en desacordo (3), En desacordo (1), De acordo (1).

Para calcular o resultado final, súmanse as respostas (valor da numeración das posibles respostas, rango 1-5) dos enunciados impares e réstaselle 5. Súmanse as respostas dos enunciados pares e réstaselle o resultado a 25. Finalmente, súmanse ambos valores e multiplícase por 2,5. Esta puntuación ten en conta o peso daqueles enunciados que son positivos e os que son negativos. Para ter en conta as respostas dos cinco usuarios, faise a media. O resultado final é o seguinte (indícase a suma das medias das respostas dos cinco usuarios):

- **Puntuación SUS** = [(4,4 + 5 + 4 + 4,6 + 5) - 5 + 25 - (1,2 + 1 + 1 + 1,2 + 1,8)] * 2,5 = **92**

A puntuación obtida é de 92 sobre 100, co cal podemos considerar que o sistema implementado ten unha usabilidade aceptable en base aos resultados do cuestionario realizado cos usuarios.

5.2. Verificación

As probas de verificación tratan de comprobar que o sistema funciona correctamente. Neste apartado recóllese o plan de probas executado e os resultados obtidos.

5.2.1. Plan de probas

A estratexia de probas levada a cabo divídese en dúas partes: probas unitarias dos servizos da API e probas de integración do sistema.

Probas unitarias

Para comprobar que os métodos dos servizos do servidor funcionan e executan correctamente os casos de uso definidos en cada un dos seus métodos, realízanse unha serie de probas de caixa blanca que demostran que o sistema funciona.

A proba limítase á funcionalidade propia de cada método dos servizos, mockeando con falsas implementacións outros métodos dos que se poida depender ou as dependencias do propio servizo (como outros servizos ou repositorios).

O obxectivo é probar todas as casuísticas posibles, incluíndo posibles camiños e datos que poden admitir os métodos, de modo que comprobemos que todos os

pasos necesarios se cumplen, os erros se tratan correctamente e a cobertura do código é a maior posible.

Os resultados das probas unitarias, xunto coa cobertura obtida, pódense ver na Figura 5.1. O resultado é un 100 % de probas superadas nos servizos e máis dun 90 % de cobertura dos métodos probados (cerca do 100 % se temos en conta só os servizos).

```
PASS tests/stadiscticTestService.unit.spec.ts (8.921 s)
PASS tests/environmentService.unit.spec.ts (9.843 s)
PASS tests/benchmarkService.unit.spec.ts (10.637 s)
PASS tests/datasetService.unit.spec.ts (10.903 s)
PASS tests/algorithmService.unit.spec.ts (11.094 s)
PASS tests/taskService.unit.spec.ts (11.587 s)
```

File	% Stmts	% Branch	% Funcs	% Lines
All files	93.97	86.25	78.65	94.06
src/algorithms	97.56	50	100	100
algorithm.ts	100	100	100	100
algorithmService.ts	96.87	50	100	100
src/benchmarks	98	85.71	100	97.91
benchmark.ts	100	100	100	100
benchmarkService.ts	97.61	85.71	100	97.5
src/datasets	100	83.33	100	100
dataset.ts	100	100	100	100
datasetService.ts	100	83.33	100	100
src/environments	100	100	100	100
environment.ts	100	100	100	100
environmentService.ts	100	100	100	100
src/shared/config	100	100	100	100
index.ts	100	100	100	100
src/shared/logger	92.85	66.66	100	92.3
index.ts	92.85	66.66	100	92.3
.../middlewares/errors	100	80	100	100
error.ts	100	80	100	100
src/shared/scp	80	100	0	75
index.ts	80	100	0	75
src/shared/ssh	57.14	100	0	50
index.ts	57.14	100	0	50
src/shared/utils	100	100	100	100
date.ts	100	100	100	100
src/stadisticTests	100	92.85	100	100
...sticTestService.ts	100	92.85	100	100
src/tasks	95.23	75	83.33	95.08
task.ts	100	100	100	100
taskService.ts	94.54	75	81.81	94.33

Figura 5.1: Resultados das probas unitarias.

Probas de integración

Mediante as probas de integración buscamos comprobar que os distintos compoñentes do sistema interactúan correctamente entre eles. Dada a magnitude do sistema desenvolvido e da complexidade para probar todas as casuísticas, decídese realizar unha serie de probas de integración sobre aquelas funcionalidades máis importantes do sistema.

PI-01 Creación dun algoritmo na plataforma.	
Descrición	Búscase comprobar que o sistema é capaz de crear un algoritmo. Compróbase que, a través dun ficheiro comprimido válido, se importa o algoritmo e se mostra a información correctamente.
Resultado esperado	O sistema crea o algoritmo e a información correspóndese cos datos do ficheiro subido.
Resultado obtido	A proba ten un resultado satisfactorio.

PI-02 Creación dun dataset na plataforma.	
Descrición	Búscase comprobar que o sistema é capaz de crear un dataset. Compróbase que, a través dun ficheiro válido, se importa o dataset e se mostra a información correctamente.
Resultado esperado	O sistema crea o dataset e a información correspóndese cos datos do ficheiro subido.
Resultado obtido	A proba ten un resultado satisfactorio.

PI-03 Creación dunha proba na plataforma.	
Descrición	Búscase comprobar que o sistema é capaz de crear unha proba. Introducindo un conxunto de algoritmos e datasets, compróbase que o sistema avalia cada par algoritmo-dataset e que os parámetros e resultados obtidos coinciden cos argumentos de entrada seleccionados.
Resultado esperado	O sistema executa cada un dos algoritmos con todos os datasets seleccionados e obtén un resultado para cada execución.
Resultado obtido	A proba ten un resultado satisfactorio.

PI-05 Consulta dunha proba na plataforma.	
Descrición	Búscase comprobar que o sistema é capaz de mostrar os resultados dunha proba. Recuperando unha proba finalizada, compróbase que se pode consultar os resultados dos algoritmos e datasets e que, tanto as gráficas como os tests estadísticos, funcionan correctamente.
Resultado esperado	O sistema recupera a proba correctamente e permite consultar todos os resultados a través da gráfica, táboas e tests estadísticos.
Resultado obtido	A proba ten un resultado satisfactorio.

5.3. Validación

As probas de validación tratan de comprobar que o sistema construído satisfai os obxectivos establecidos no proxecto. Isto implica que o produto final ten a funcionalidade esperada.

5.3.1. Criterios de aceptación

Para comprobar se o sistema cumple cos requisitos establecidos imos analizar se os criterios de aceptación establecidos para o proxecto se cumplen.

CA-01 O sistema permite que o usuario realice as funcións propias da plataforma sen producirse erros de execución .	
Descrición	O usuario debe poder empregar calquera funcionalidade ofrecida polo sistema sen producirse ningún tipo de erro non esperado e resultado da propia execución.
Estado	**Superado**, o sistema non produce erros inesperados.

CA-02 O sistema permite tanto a subida de datasets como a consulta por parte do usuario .	
Descrición	O usuario debe poder subir un ficheiro cun dataset e consultar aqueles datasets que xa foron subidos á plataforma anteriormente.
Estado	**Superado**, o usuario pode consultar e subir datasets.

CA-03 O sistema permite subir e consultar algoritmos por parte do usuario.	
Descrición	O usuario debe poder subir un ficheiro cun algoritmo e consultar os algoritmos que xa foron subidos á plataforma anteriormente.
Estado	**Superado**, o usuario pode consultar e subir algoritmos.

CA-04 O sistema permite consultar as probas realizadas en calquera momento pasado ou actual, incluíndo tantos as probas en proceso de execución como aquelas que xa remataron e obtiveron os seus resultados.	
Descrición	O usuario debe pode consultar tanto o historial de execución de probas como aquelas probas que se atopan en execución no momento da consulta.
Estado	**Superado**, o usuario pode consultar as probas rematadas e non rematadas.

CA-05 O sistema permite executar probas sobre múltiples algoritmos e datasets ao mesmo tempo.	
Descrición	O usuario debe poder seleccionar un conxunto de datasets e algoritmos e crear unha proba con eses datos, móstrandose o resultado da execución ao finalizar dita proba.
Estado	**Superado**, o usuario pode realizar probas con varios algoritmos e datasets ao mesmo tempo.

CA-06 O sistema permite adestrar os algoritmos e avalialos sobre os datasets.	
Descrición	O sistema debe adestrar os algoritmos sen necesidade de intervención do usuario e realizar as avaliacións de todos os algoritmos dunha proba sobre todos os datasets desta.
Estado	**Superado**, o adestramento e avaliación realízanse de forma automatizada e independente.

Capítulo 6

Conclusións e posibles ampliacións

6.1. Conclusións

Este proxecto foi desenvolvido a partir da identificación dunha necesidade concreta: unha plataforma que permita a proba e comparación de algoritmos de Question Answering cun nivel de esforzo reducido. Establecéronse unha serie de obxectivos e requisitos para o sistema proposto, e implementouse unha plataforma que lle permite aos usuarios subir os seus propios datasets e algoritmos, crear probas e consultar o historial e resultados das mesmas. Isto implica que os criterios de aceptación do producto final foron cumpridos e o proxecto puido finalizar con éxito.

Este traballo permitiume aprender moito sobre unha gran variedade de tecnoloxías, ferramentas e o campo do QA (algoritmos, métricas, metodoloxías de traballo, etc.). A gran maioría de conceptos eran novos e non foran estudados durante o Grao, pero tamén se puideron aplicar os coñecementos técnicos e teóricos aprendidos durante estes catro anos de carreira.

Finalmente, destacar que o proxecto implicou un grande esforzo, e espérase que esta memoria recolla, da forma máis fiel posible, todo o traballo realizado e o sistema desenvolvido, de modo que a súa implementación sirva de utilidade ás persoas que traballan no campo do Question Answering.

6.2. Posibles ampliacións

A plataforma desenvolvida ten unha serie de aspectos que están abertos a posibles melloras.

O sistema de tarefas está implementado como unha cola *FIFO* [37], co cal a súa execución prodúcese na orde de chegada, podendo provocar bloqueos de tarefas curtas por outras máis longas. Unha posible mellora sería implementar

un sistema de colas con prioridade ou a execución paralela de múltiples tarefas ao mesmo tempo. Tamén se podería implementar a cola mediante algún sistema de colas, como *RabbitMQ* [41] ou *Kafka* [42].

Outra vía de mellora pode ser a posibilidade de ter varias máquinas para a execución dos algoritmos, posiblemente con distintas características técnicas, e poder escoller en que máquina se executan as probas seguindo criterios como a carga de traballo ou as necesidades do algoritmo. Isto podería implementarse seguindo algún tipo de criterio á hora de executar as probas ou permitindo que o usuario escolla o servidor á hora de crear a proba.

Outra mellora que se podería aplicar sería o feito de ter un sistema de rexistro de usuarios e poder rexistrar quen realiza cada proba e poder ter probas de carácter privado ou resultados públicos, accesibles por calquera persoa.

Finalmente, o último aspecto no que se poden aceptar melloras sería a posibilidade de poder empregar calquera tipo de dataset, sen depender do formato concreto dos datasets de SQuAD 2.0 [19].

Apéndice A

Figuras e táboas

A.1. Táboas de riscos do proxecto

RISC-01 Planificación non se axusta á realidade.	
Descrición	A planificación temporal non se adapta ao tempo real de desenvolvemento e prodúcense atrasos.
Probabilidade	Media
Impacto	Medio
Exposición	Media

RISC-02 Certas funcionalidades teñen que ser rediseñadas.	
Descrición	Ao rematar unha funcionalidade detéctase que non é o que se quería obter e hai que redeseñala e reimplementala.
Probabilidade	Baixo
Impacto	Medio
Exposición	Baixa

RISC-03 Non é posible realizar unha implementación que se adapte aos obxectivos.	
Descrición	Non se atopan ferramentas ou librerías que permitan realizar a implementación dalgún obxectivo fundamental do proxecto e non se pode rematar o traballo.
Probabilidade	Media
Impacto	Alto
Exposición	Alta

RISC-04 Perda dun elemento da liña base.	
Descrición	asdasdasdadasdasdasd.
Probabilidade	Baixa
Impacto	Alto
Exposición	Media

RISC-05 Fallo do entorno de desenvolvemento.	
Descrición	O ordenador no que se realiza o traballo falla e tense que sustituir.
Probabilidade	Baixa
Impacto	Medio
Exposición	Baixa

RISC-06 O sistema non resulta útil.	
Descrición	Ao rematar o traballo a súa utilidade non é a esperada e non se emprega.
Probabilidade	Baixa
Impacto	Baixo
Exposición	Baixa

RISC-07 Algunha funcionalidade non é compatible.	
Descrición	A implementación dunha parte do sistema fai que algún compoñente previo deixe de funcionar correctamente.
Probabilidade	Media
Impacto	Medio
Exposición	Media

A.2. Táboas de requisitos de información

RI-01 Algoritmo.	
Descrición	Representa a información e configuración dun algoritmo subido á plataforma.
Datos específicos	▪ Identificador ▪ Nome ▪ Nome do ficheiro *ZIP* ▪ Lista de parámetros posibles • Nome • Indicador se é obrigatorio • Indicador se se emprega para o adestramento • Indicador se se emprega para a avaliación • Tipo de dato • Valor por defecto • Comentario de axuda • Indicador se é un path ou é un valor normal ▪ Data de subida

RI-02 Dataset.	
Descrición	Representa a información dun dataset subido á plataforma.
Datos específicos	▪ Identificador ▪ Nome ▪ Nome do ficheiro ▪ Nome orixinal do ficheiro ▪ Tamaño do ficheiro ▪ Data de subida

RI-03 Proba.	
Descrición	Representa, como unha unidade, un conxunto de tarefas e resultados asociados á execución e avaliación duns algoritmos sobre uns datasets.
Datos específicos	▪ Identificador ▪ Nome ▪ Lista de datasets • Nome • Nome do ficheiro ▪ Lista de algoritmos • Identificador do algoritmo • Nome • Lista de parámetros de adestramento ◦ Nome ◦ Valor • Lista de parámetros de avaliación ◦ Nome ◦ Valor ▪ Lista de tarefas • Identificador • Indicador se está rematada • Indicador se rematou sen erros • Identificador algoritmo • Acción (avaliar ou adestrar) • Nome dataset • Traza erro (en caso de producirse) ▪ Lista de resultados • Nome dataset • Identificador algoritmo • Resultado (exact, f1, total respostas) ▪ Data de creación

RI-04 Tarefa.	
Descrición	Representa a información asociada a unha tarefa a ser executada polo sistema, ben sexa un adestramento dun algoritmo ou unha avaliación concreta dun algoritmo sobre un dataset.

Datos específicos	▪ Identificador
	▪ Identificador do entorno
	▪ Acción (avaliar ou adestrar)
	▪ Nome do ficheiro do dataset
	▪ Lista de parámetros
	  • Nome
	  • Valor
	▪ Indicador se está activa
	▪ Data de creación

RI-05 Entorno.	
Descrición	Representa a información dun entorno de execución creado, a partir duns parámetros de adestramento específicos, para un algoritmo concreto.

Datos específicos	▪ Identificador
	▪ Identificador do algoritmo
	▪ Nome do ficheiro do dataset
	▪ Lista de parámetros de adestramento
	  • Nome
	  • Valor
	▪ Data de creación

A.3. Táboas dos casos de uso

UC-01 Subir dataset.	
Descrición	O usuario sube un novo dataset á plataforma.
Precondición	-
Secuencia normal	▪ **Paso 1**: O actor **ACT-01** selecciona subir un novo dataset. ▪ **Paso 2**: O actor **ACT-01** sube un ficheiro *JSON* co dataset e introduce o nome do dataset. ▪ **Paso 3**: O sistema comproba que o nome non existe. ▪ **Paso 4**: O sistema sube o ficheiro ao servidor onde se van executar as probas. ▪ **Paso 5**: O sistema garda o dataset correctamente.
Postcondición	O dataset queda correctamente gardado no sistema.
Excepcións	▪ **Paso 3**: O nome xa existe, móstrase unha mensaxe de erro e remata o caso.

UC-02 Consultar datasets.	
Descrición	O usuario desexa consultar os datasets dispoñibles e o sistema recupera a lista de datasets rexistrados.
Precondición	-
Secuencia normal	▪ **Paso 1**: O actor **ACT-01** selecciona consultar todos os datasets. ▪ **Paso 2**: O sistema recupera a lista de datasets dispoñibles.
Postcondición	O sistema recupera todos os datasets rexistrados.
Excepcións	-

UC-03 Consultar dataset.	
Descrición	O usuario desexa consultar un dataset concreto rexistrado no sistema.
Precondición	O dataset existe no sistema.
Secuencia normal	▪ **Paso 1**: O actor **ACT-01** selecciona un dataset concreto. ▪ **Paso 2**: O sistema recupera o dataset. ▪ **Paso 3**: O sistema recupera o ficheiro do dataset. ▪ **Paso 4**: O sistema mostra a información do dataset xunto cun resumo do contido do mesmo.
Postcondición	-
Excepcións	-

UC-04 Subir algoritmo.	
Descrición	O usuario sube un novo algoritmo á plataforma.
Precondición	O ficheiro comprimido do algoritmo ten unha estrutura correcta.
Secuencia normal	▪ **Paso 1**: O actor **ACT-01** selecciona subir un novo algoritmo. ▪ **Paso 2**: O actor **ACT-01** sube un *ZIP* co algoritmo. ▪ **Paso 3**: O sistema recupera o contido do ficheiro de configuración do algoritmo. ▪ **Paso 4**: O sistema comproba que o nome do algoritmo non está en uso. ▪ **Paso 5**: O sistema sube o ficheiro comprimido ao servidor onde se van executar as probas. ▪ **Paso 6**: O sistema garda o algoritmo correctamente.
Postcondición	O algoritmo queda rexistrado correctamente no sistema.
Excepcións	▪ **Paso 4**: O nome xa existe, móstrase unha mensaxe de erro e remata o caso.

UC-05 Consultar algoritmos.	
Descrición	O usuario desexa consultar os algoritmos dispoñibles e o sistema recupera a lista de algoritmos rexistrados.
Precondición	-
Secuencia normal	▪ **Paso 1**: O actor **ACT-01** selecciona consultar todos os algoritmos. ▪ **Paso 2**: O sistema recupera a lista de algoritmos dispoñibles.
Postcondición	O sistema recupera todos os algoritmos rexistrados.
Excepcións	-

UC-06 Consultar algoritmo.	
Descrición	O usuario desexa consultar un algoritmo concreto rexistrado no sistema.
Precondición	O algoritmo existe no sistema.
Secuencia normal	▪ **Paso 1**: O actor **ACT-01** selecciona un algoritmo concreto. ▪ **Paso 2**: O sistema recupera o algoritmo. ▪ **Paso 3**: O sistema recupera o ficheiro de configuración. ▪ **Paso 4**: O sistema mostra a información do algoritmo xunto coa información contida na configuración do mesmo.
Postcondición	-
Excepcións	-

UC-07 Consultar probas en execución.	
Descrición	O usuario desexa consultar as probas que se atopan en estado de execución.
Precondición	-
Secuencia normal	▪ **Paso 1**: O actor **ACT-01** selecciona consultar as probas en execución. ▪ **Paso 2**: O sistema recupera aquelas probas que teñan algunha tarefa sen rematar. ▪ **Paso 3**: O sistema mostra a lista de probas xunto cun indicador da porcentaxe de progreso para cada unha.
Postcondición	-
Excepcións	-

UC-08 Consultar historial de probas.	
Descrición	O usuario desexa consultar as probas que se atopan finalizadas.
Precondición	-
Secuencia normal	▪ **Paso 1**: O actor **ACT-01** selecciona consultar o historial de probas rematadas. ▪ **Paso 2**: O sistema recupera aquelas probas que teñan todas as tarefas rematadas. ▪ **Paso 3**: O sistema mostra a lista de probas.
Postcondición	-
Excepcións	-

UC-09 Consultar proba.	
Descrición	O usuario desexa consultar a información asociada a unha proba.
Precondición	A proba existe no sistema.
Secuencia normal	▪ **Paso 1**: O actor **ACT-01** selecciona consultar unha proba concreta. ▪ **Paso 2**: O sistema recupera a información da proba. ▪ **Paso 3**: O sistema mostra o estado das tarefas no caso de non ter rematado algunha delas, ou o resultado das execucións. Este resultado amósase tanto en gráficas que permitan agrupar por algoritmo ou dataset, como táboas de resultados e tests estadísticos que permitan comparar o rendemento dos algoritmos. O sistema calcula o test estadístico máis axeitado en función do número de datasets e algoritmos.
Postcondición	-
Excepcións	-

UC-10 Crear proba.	
Descrición	O usuario desexa crear unha nova proba.
Precondición	-
Secuencia normal	▪ **Paso 1**: O actor **ACT-01** selecciona crear unha nova proba. ▪ **Paso 2**: O sistema recupera os datasets e algoritmos dispoñibles. ▪ **Paso 3**: O actor **ACT-01** selecciona os datasets e algoritmos que desexe. Para cada algoritmo, configura os parámetros necesarios. ▪ **Paso 4**: Para cada algoritmo seleccionado, o sistema crea o entorno de execución no caso de non existir un entorno cos parámetros de adestramento seleccionados. ▪ **Paso 5**: Para cada algoritmo seleccionado, o sistema crea unha tarefa de adestramento no caso de non estar adestrado dito algoritmo cos parámetros indicados. ▪ **Paso 6**: Para cada algoritmo seleccionado, o sistema crea unha tarefa de avaliación por cada un dos datasets seleccionados. ▪ **Paso 7**: O sistema garda a proba correctamente.
Postcondición	A proba queda rexistrada no sistema correctamente.
Excepcións	-

A.4. Rutas do servidor e do cliente web

Servizo	Ruta	Descrición
Algoritmos	**POST /algorithms**	Crea un algoritmo.
	GET /algorithms	Recupera todos os algoritmos.
	GET /algorithms/:id	Recupera un algoritmo por id.
Datasets	**POST /datasets**	Crea un dataset.
	GET /datasets	Recupera todos os datasets.
	GET /datasets/:id	Recupera un dataset por id.
Tarefas	**PUT /tasks/:id**	Notifica a finalización dunha tarefa.
	POST /tasks/:id/results	Notifica o resultado dunha avaliación.
Probas	**POST /benchmarks**	Crea unha proba.
	GET /benchmarks	Recupera unha lista de probas.
	GET /benchmarks/:id	Recupera unha proba por id.
Tests estadísticos	**POST /stadistic-tests**	Crea un test estadístico.

Cadro A.1: Rutas do servidor.

Ruta	Descrición
/	Mostra a páxina principal.
/algorithms	Mostra a lista de algoritmos.
/algorithms/:id	Mostra un algoritmo concreto.
/datasets	Mostra a lista de datasets.
/datasets/:id	Mostra un dataset concreto.
/benchmarks	Mostra o historial de probas.
/benchmarks/new	Mostra a páxina de creación dunha proba.
/benchmarks/executing	Mostra as probas en execución.
/benchmarks/:id	Mostra unha proba concreta.

Cadro A.2: Rutas do cliente web.

Apéndice B

Manuais técnicos

Este manual recopila a información relativa á estrutura do proxecto, a súa posta en funcionamento e os requisitos do sistema. O proxecto está dividido en dúas partes principais que serán descritas en seccións posteriores, e que se desenvolven como proxectos independentes: o servidor e o cliente web. O código do proxecto atópase dispoñible a través da seguinte ligazón:

`https://nubeusc-my.sharepoint.com/:f:/g/personal/david_carracedo_rai_usc_es/EpRYS6tvnoZAmOZMNZqShREBTnclx1pB--mL0GAQRkDeWA?e=7xSEVl`

Dentro deste repositorio atopamos a carpeta ***squad-datasets/***, que contén ficheiros con datasets de exemplo, e ***app/***, que á súa vez contén tres directorios: ***algoritmos/***, coa implementación dos algoritmos adaptados á plataforma, ***cliente/***, co código do cliente web, e ***servidor/***, co código da API REST.

Os ficheiros *execute_conda_environment.sh*, *prepare_environment.sh* e *main_files.zip*, que se atopan no directorio ***app/servidor/scripts***, deben ser subidos ao servidor de execución e colocados dentro da mesma carpeta. Esta carpeta debe estar no mesmo directorio ca outras dúas chamadas *environments* e *uploads* para que os scripts poidan acceder de forma correcta a elas.

O proxecto foi desenvolvido cos editores de código **Visual Studio Code** [13] e **PyCharm** [12], pero a edición de calquera dos elementos do mesmo pode realizarse con calquera editor de texto.

Os requisitos necesarios para executar e probar este proxecto son ter un sistema **Linux** con **Bash** 5.1.4 [10], **Node** 16 [4], **npm** 8 e **MongoDB** 5.0.8 [3]. Ademais, será necesario ter instalado o cliente SSH na máquina onde se execute o servidor e o servidor SSH na máquina onde se executen os algoritmos.

Tamén será preciso xerar un par de chaves para o SSH. Para isto, podemos crealas mediante o comando ***ssh-keygen***. Unha vez xeradas, debemos instalar a chave pública no servidor remoto, operación que podemos realizar co comando ***ssh-copy-id -i chave.pub usuario@servidor***.

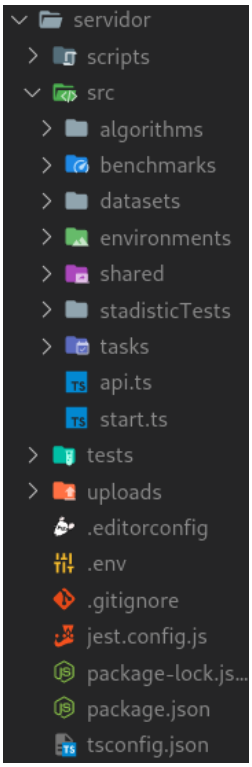


Figura B.1: Estrutura de ficheiros do servidor.

B.1. Servidor

A Figura B.1 representa a estrutura de ficheiros e directorios do proxecto do servidor.

O ficheiro **package.json** recolle as dependencias do proxecto e as accións que se poden executar sobre o mesmo. Para instalar as dependencias executamos o comando ***npm install***. Para arrancar o servidor en modo desenvolvemento (dispoñible en *http://localhost:9999* por defecto) executamos o comando ***npm run dev***. Para xerar unha versión lista para o desplegue en produción, podemos executar o comando ***npm run build***, que compilará o código, seguido do comando ***npm start*** para arrancalo. Co comando ***npm run test:unit*** executamos os tests unitarios e co comando ***npm run test:unit:coverage*** obtemos os resultados da cobertura das probas.

En canto á estrutura de ficheiros, destacamos os seguintes elementos:

- **scripts/**: directorio cos scripts necesarios para a execución dos algoritmos.
- **src/algoritmos/**: directorio coa funcionalidade relacionada cos algoritmos (servizo).
- **src/benchmarks/**: directorio coa funcionalidade relacionada coas probas (servizo).

- **src/datasets/**: directorio coa funcionalidade relacionada cos datasets (servizo).
- **src/environments/**: directorio coa funcionalidade relacionada cos entornos (servizo).
- **src/shared/**: directorio cos elementos comúns a todo o servidor, como interfaces comúns, as rutas, utilidades e configuracións.
- **src/stadisticTests/**: directorio coa funcionalidade relacionada cos test estadísticos (servizo).
- **src/tasks/**: directorio coa funcionalidade relacionada coas tarefas (servizo).
- **tests/**: directorio cos tests unitarios e mocks.
- **uploads/**: directorio para o almacenamento dos ficheiros subidos polos usuarios.

A **configuración** do servidor realízase a través do ficheiro ***src/shared/config/index.ts***, que contén unha serie de campos con valores asociados a cada un dos parámetros de configuración do sistema.

Dentro dos directorios asociados aos servizos, encontramos un ficheiro coa definición dos tipos e modelos ([modulo].ts), un ficheiro co servizo ([modulo]Service.ts), un ficheiro co controlador e rutas asociadas no caso de telas ([modulo]Controller.ts), unha interface coa definición do repositorio no caso de telo ([modulo]Repository.ts), e unha implementación do repositorio cando proceda (mongo[modulo]Repository.ts).

B.2. Cliente web

Recóllese na Figura B.2 a estrutura de ficheiros e directorios do proxecto do cliente web.

O ficheiro **package.json** recolle as dependencias do proxecto e as accións que se poden executar sobre o mesmo. Para instalar as dependencias executamos o comando ***npm install***. Para arrancar o cliente web (dispoñible en *http://localhost:3000* por defecto) executamos o comando ***npm start***.

En canto á estrutura de ficheiros, destacamos os seguintes elementos:

- **src/assets/**: directorio cos recursos gráficos empregados.
- **src/components/**: directorio cos compoñentes da aplicación. Cada compoñente está nun ficheiro propio, podendo ter un directorio agrupando subcompoñentes ou estilos.

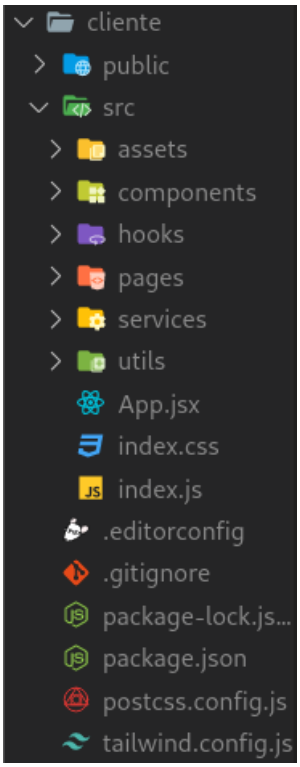


Figura B.2: Estrutura de ficheiros do cliente web.

- **src/hooks/**: directorio coas funcións relacionadas co estado e reutilizables en toda a aplicación.
- **src/pages/**: directorio coas distintas rutas do cliente web. Cada páxina é un ficheiro e contén só a estrutura do HTML e compoñentes necesarios. A funcionalidade realízase nos compoñentes.
- **src/services/**: directorio coas funcións asociadas á comunicación con outras partes do sistema, como as chamadas á API REST.
- **src/utils/**: directorio con funcións auxiliares comúns a toda a aplicación.
- **src/App.jsx/**: este ficheiro define a estrutura da aplicación e todas as rutas dispoñibles.
- **src/index.css/**: ficheiro cos estilos globais a toda a aplicación.
- **tailwind.config.js**: ficheiro que permite editar a configuración dos estilos da librería *Tailwind* [32].

Apéndice C

Manuais de usuario

A plataforma está composta por dous compoñentes que permiten a interación co sistema: o servidor e o cliente web. O primeiro compoñente permite a comunicación directa co servidor a través dunha API REST. O segundo, permite unha interación a través dunha interface web.

Nesta sección recóllese a información necesaria para a utilización da plataforma por parte dos usuarios.

C.1. Servidor

A continuación, descríbense as rutas dos diferentes servizos do servidor, os seus parámetros e as posibles respostas.

C.1.1. Servizo de algoritmos

Ruta	**POST /algorithms**
Descrición	Crea un algoritmo.
Parámetros (body - multipart)	▪ **file** (ficheiro *ZIP*)
Resposta HTTP	▪ **201**: creado.
Respostas HTTP erros	▪ **400**: parámetros incorrectos. ▪ **409**: o nome do algoritmo xa existe. ▪ **500**: erro almacenando o ficheiro.

Ruta	**GET /algorithms**
Descrición	Recupera todos os algoritmos.
Parámetros	-
Resposta HTTP	▪ **200**: correcto.
Respostas HTTP erros	-

Ruta	**GET /algorithms/:id**
Descrición	Recupera un algoritmo por id.
Parámetros (URL)	▪ **:id** (identificador do algoritmo)
Resposta HTTP	▪ **200**: correcto.
Respostas HTTP erros	▪ **404**: algoritmo non atopado.

C.1.2. Servizo de datasets

Ruta	**POST /datasets**
Descrición	Crea un dataset.
Parámetros (body - multipart)	▪ **file** (ficheiro *JSON*) ▪ **nome** (string)
Resposta HTTP	▪ **201**: creado.
Respostas HTTP erros	▪ **400**: parámetros incorrectos. ▪ **409**: o nome do dataset xa existe. ▪ **500**: erro almacenando o ficheiro.

Ruta	**GET /datasets**
Descrición	Recupera todos os datasets.
Parámetros	-
Resposta HTTP	▪ **200**: correcto.
Respostas HTTP erros	-

Ruta	**GET /datasets/:id**
Descrición	Recupera un dataset por id.
Parámetros (URL)	▪ **:id** (identificador do dataset)
Resposta HTTP	▪ **200**: correcto.
Respostas HTTP erros	▪ **404**: dataset non atopado.

C.1.3. Servizo de tarefas

Ruta	**PUT /tasks/:id**
Descrición	Notifica a finalización dunha tarefa.
Parámetros (URL)	▪ **:id** (identificador da tarefa)
Parámetros (body - json)	▪ **success** (boolean)
Resposta HTTP	▪ **204**: sen contido.
Respostas HTTP erros	-

Ruta	**POST /tasks/:id/results**
Descrición	Notifica o resultado dunha avaliación.
Parámetros (URL)	▪ **:id** (identificador da tarefa)
Parámetros (body - json)	resultados avaliación SQuAD 2.0 [19].
Resposta HTTP	▪ **204**: sen contido.
Respostas HTTP erros	-

C.1.4. Servizo de probas

Ruta	**POST /benchmarks**
Descrición	Crea unha proba.
Parámetros (body - json)	▪ **name** (string) ▪ **datasets** (lista de): (string) ▪ **algorithms** (lista de): • **id** (string) • **name** (string) • **trainParameters** (lista de): ◦ **param** (string) ◦ **value** (string, float, integer) • **evaluationParameters** (lista de): ◦ **param** (string) ◦ **value** (string, float, integer)
Resposta HTTP	▪ **201**: creado.
Respostas HTTP erros	-

Ruta	**GET /benchmarks**
Descrición	Recupera unha lista de probas.
Parámetros (query)	▪ **filter**: "history" para recuperar o historial ou "progress" para recuperar as probas en execución.
Resposta HTTP	▪ **200**: correcto.
Respostas HTTP erros	▪ **400**: filtro inválido.

Ruta	**GET /benchmarks/:id**
Descrición	Recupera unha proba por id.
Parámetros (URL)	▪ **:id** (identificador da proba)
Resposta HTTP	▪ **200**: correcto.
Respostas HTTP erros	▪ **404**: proba non atopada.

C.1.5. Servizo de tests estadísticos

Ruta	**POST /stadistic-tests**
Descrición	Crea un test estadístico.
Parámetros (query)	▪ **significanceLevel**: (float)
Parámetros (body - json)	Dicionario con entradas: ▪ [**nome-algoritmo**] (lista de): valores numéricos do resultado *F1* para cada dataset.
Resposta HTTP	▪ **200**: correcto.
Respostas HTTP erros	▪ **400**: parámetros incorrectos.

C.2. Cliente web

O cliente web permítelle aos usuarios empregar a plataforma a través dunha interface web, accesible a través dun navegador. Nesta sección descríbense as diferentes vistas da aplicación, xunto cunha descrición de cada unha e unha explicación de cómo traballar con elas.

C.2.1. Pantalla principal

Pantalla principal	
Descrición	Vista de acceso á plataforma.
Ruta	/
Parámetros	-

A pantalla principal é a vista que se lle amosa ao usuario unha vez accede á plataforma. Dende esta pantalla, o usuario pode acceder ao resto de pantallas a través dun menú horizontal. Tamén pode consultar o histórico de probas e as probas en execución mediante dúas táboas que se atopan na zona inferior. Podemos ver a representación desta pantalla na Figura C.1.

Figura C.1: Vista da pantalla principal da plataforma.

O usuario pode premer sobre as tarxetas que se mostran no apartado *Navegación* da Figura C.1. Isto permite acceder á pantalla indicada.

C.2.2. Pantalla de datasets

Pantalla de datasets	
Descrición	Permite consultar a lista de datasets rexistrados no sistema.
Ruta	/datasets
Parámetros	-

A pantalla de datasets permite que o usuario consulte a lista de datasets que se atopan rexistrados no sistema. Esta lista represéntase a través dunha táboa coa información básica de cada dataset. O usuario pode premer sobre cada unha

das filas, o que permite acceder á páxina asociada a cada un dos datasets. A representación desta pantalla pode verse na Figura C.2.

Figura C.2: Vista da pantalla de datasets.

Debaixo da táboa de datasets atopamos un recadro que permite subir un novo dataset ao sistema. O usuario pode arrastrar un ficheiro *JSON* co datasets enriba do recadro, ou simplemente premer sobre el e seleccionar o ficheiro no modal que se lle abrirá (a estrutura deste ficheiro descríbese na Figura C.3). Unha vez seleccionado o ficheiro, aparece a información do mesmo e un campo de texto que o usuario debe cubrir co nome que desexa asignarlle ao novo dataset (Figura C.4). Este campo é obrigatorio e debe ser único en todo o sistema.

```
{
  "version": {
    "type": "string"
  },
  "data": {
    "type": "array",
    "items": {
      "type": "object",
      "properties": {
        "title": {
          "type": "string"
        },
        "paragraphs": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "qas": {
                "type": "array",
                "items": {
                  "type": "object",
                  "properties": {
                    "id": {
                      "type": "string"
                    },
                    "question": {
                      "type": "string"
                    },
                    "is_impossible": {
                      "type": "boolean"
                    },
                    "answers": {
                      "type": "array",
                      "items": {
                        "type": "object",
                        "properties": {
                          "text": {
                            "type": "string"
                          },
                          "answer_start": {
                            "type": "integer"
                          }
                        }
                      }
                    },
                    "plausible_answers": {
                      "required": false,
                      "type": "array",
                      "items": {
                        "type": "object",
                        "properties": {
                          "text": {
                            "type": "string"
                          },
                          "answer_start": {
                            "type": "integer"
                          }
                        }
                      }
                    }
                  }
                }
              },
              "context": {
                "type": "string"
              }
            }
          }
        }
      }
    }
  }
}
```

Figura C.3: Estrutura do ficheiro *JSON* dun dataset.

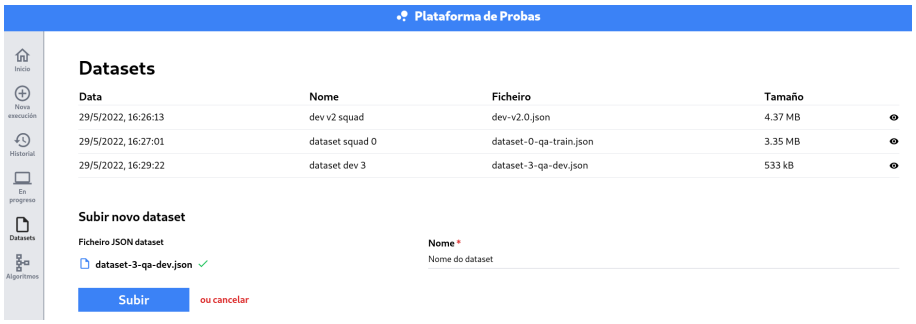


Figura C.4: Vista da subida dun dataset.

Finalmente, o usuario pode finalizar o rexistro do dataset ou cancelar a operación mediante os botóns na zona inferior.

Erros

Mensaxe	Descrición
O dataset [nome] xa existe	O nome introducido no campo de texto xa se corresponde co nome rexistrado por algún dataset do sistema. O usuario debe cambiar o nome e repetir a operación de subida ou cancelar a operación.

C.2.3. Pantalla de consulta dun dataset

Pantalla de consulta dun dataset	
Descrición	Permite consultar un dataset concreto.
Ruta	**/datasets/:id**
Parámetros	**:id** - identificador do dataset

A pantalla de consulta dun dataset permite consultar a información relativa a un dataset concreto. Dentro desta pantalla o usuario pode ver a información do dataset así como un resumo do seu contido, incluíndo os temas que inclúen as súas cuestións e unha vista previa dos datos. Pódese ver un exemplo de consulta dun dataset na Figura C.5.

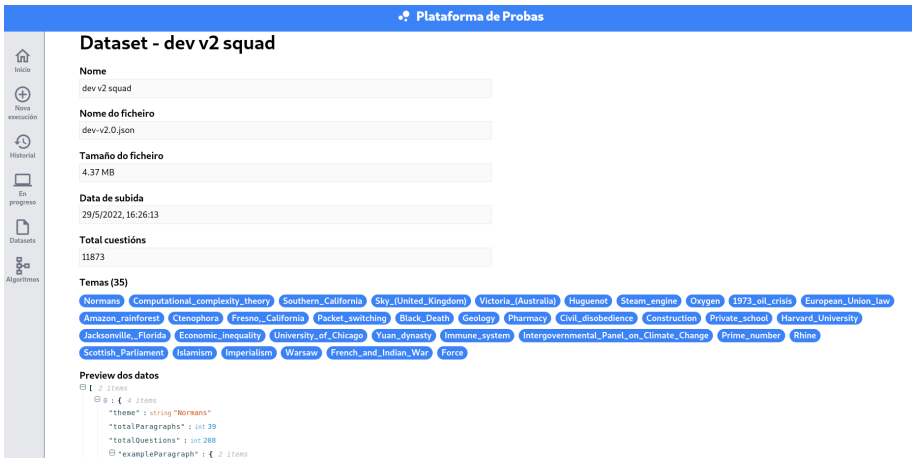


Figura C.5: Vista da pantalla de consulta dun dataset.

C.2.4. Pantalla de algoritmos

Pantalla de algoritmos	
Descrición	Permite consultar a lista de algoritmos rexistrados no sistema.
Ruta	**/algorithms**
Parámetros	-

A pantalla de algoritmos permite que o usuario consulte a lista de algoritmos que se atopan rexistrados no sistema, representados mediante unha táboa coa información básica de cada un. O usuario pode premer sobre cada unha das filas e acceder á páxina asociada a cada algoritmo. A representación desta pantalla pode verse na Figura C.6.

Figura C.6: Vista da pantalla de algoritmos.

Debaixo da táboa de algoritmos atopamos un recadro que permite subir un novo algoritmo ao sistema. O usuario pode arrastrar un ficheiro comprimido *ZIP* enriba do recadro, ou simplemente premer sobre el e seleccionar o ficheiro no

modal que se lle abrirá. Unha vez seleccionado o ficheiro, aparece a información do mesmo e dous botóns para finalizar o rexistro ou cancelar a operación (Figura C.7).

Figura C.7: Vista da subida dun algoritmo.

A estrutura do ficheiro comprimido que contén o algoritmo é a representada na Figura C.8. O contido do ficheiro de configuración do algoritmo descríbese na Figura C.9.

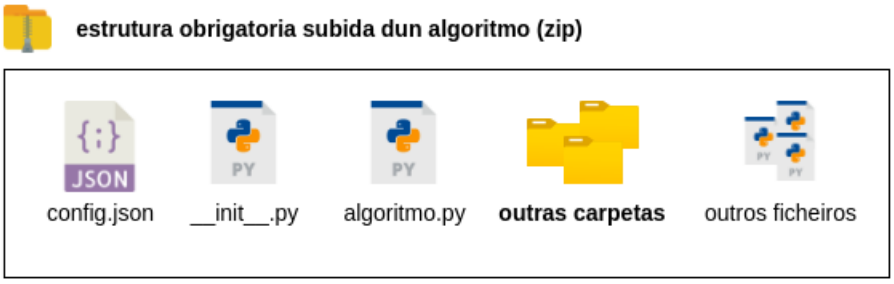


Figura C.8: Estrutura do ficheiro comprimido dun algoritmo.

```
{
    "algorithm_name": {
        "type": "string"
    },
    "algorithm_class_file": {
        "type": "string"
    },
    "algorithm_class_name": {
        "type": "string"
    },
    "environment": {
        "type": "object",
        "properties": {
            "python_version": {
                "type": "string"
            },
            "dependencies": {
                "type": "array",
                "items": {
                    "type": "object",
                    "properties": {
                        "lib": {
                            "type": "string"
                        },
                        "version": {
                            "type": ["string", "null"]
                        }
                    }
                }
            },
            "extra_python_commands": {
                "type": "array",
                "items": {
                    "type": "string"
                }
            }
        }
    },
    "args": {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
                "name": {
                    "type": "string"
                },
                "required": {
                    "type": "boolean"
                },
                "for_train": {
                    "type": "boolean"
                },
                "for_evaluation": {
                    "type": "boolean"
                },
                "type": {
                    "type": "string",
                    "enum": ["string", "float", "int"]
                },
                "default": {
                    "type": ["string", "float", "int", "null"]
                },
                "choices": {
                    "type": ["array", "null"],
                    "items": {
                        "type": ["string", "float", "int"]
                    }
                },
                "help": {
                    "type": ["string", "null"]
                },
                "ls_path": {
                    "type": "boolean"
                }
            }
        }
    }
}
```

Figura C.9: Estrutura do ficheiro de configuración dun algoritmo.

Erros

Mensaxe	Descrición
O algoritmo [nome] xa existe	O nome asociado ao algoritmo, que aparece no ficheiro de configuración do mesmo, correspóndese co nome rexistrado por algún algoritmo do sistema. O usuario debe cambiar o nome e repetir a operación de subida ou cancelar a operación.

C.2.5. Pantalla de consulta dun algoritmo

Pantalla de consulta dun algoritmo	
Descrición	Permite consultar un algoritmo concreto.
Ruta	**/algorithms/:id**
Parámetros	**:id** - identificador do algoritmo

A pantalla de consulta dun algoritmo permite consultar a información relativa a un algoritmo concreto. Dentro desta pantalla o usuario pode ver a información do algoritmo, incluíndo a lista de dependencias e os parámetros que admite. Estes parámetros móstranse nunha táboa xunto cos tipos e valores admitidos. A Figura C.10 representa un exemplo de consulta dun algoritmo.

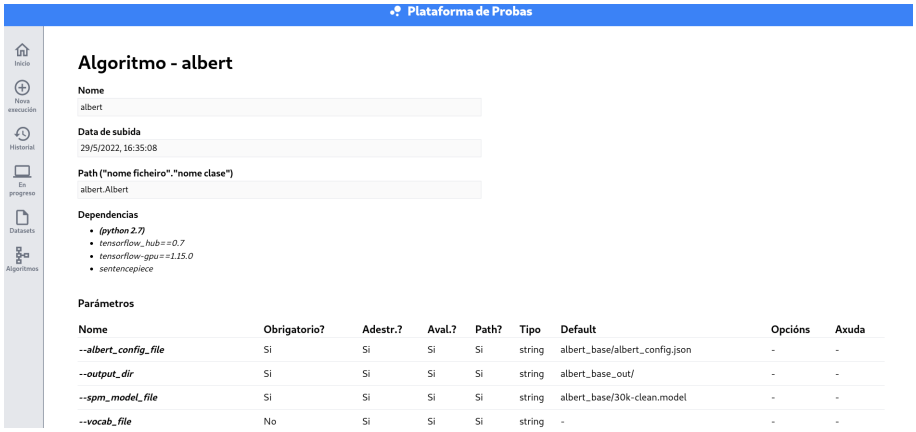


Figura C.10: Vista da pantalla de consulta dun algoritmo.

C.2.6. Pantalla do historial de execución

Pantalla do historial de execución	
Descrición	Permite consultar o historial de probas executadas na plataforma.
Ruta	**/benchmarks**
Parámetros	-

A pantalla do historial de execución permite consultar as probas que se executaron na plataforma e están rematadas (non teñen ningunha tarefa asociada pendente). Esta pantalla inclúe unha táboa coa lista de probas e, premendo sobre cada unha das filas, o usuario pode acceder á pantalla de consulta dunha proba concreta. Móstrase esta pantalla na Figura C.11.

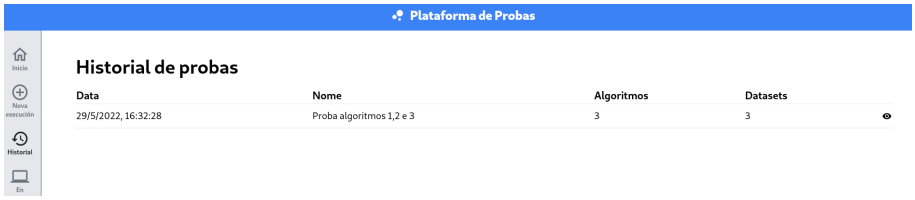


Figura C.11: Vista da pantalla do historial de execución.

C.2.7. Pantalla de probas en execución

Pantalla de probas en execución	
Descrición	Permite consultar as probas que se atopan en execución.
Ruta	**/benchmarks/executing**
Parámetros	-

A pantalla de probas en execución permite consultar as probas que se están executando no momento da consulta. Esta pantalla inclúe unha táboa coa lista de probas e, premendo sobre cada unha das filas, o usuario pode acceder á pantalla de consulta dunha proba concreta. Móstrase a representación desta pantalla na Figura C.12.

Figura C.12: Vista da pantalla de probas en execución.

C.2.8. Pantalla de creación dunha proba

Pantalla de creación dunha proba	
Descrición	Permite crear unha nova proba.
Ruta	**/benchmarks/new**
Parámetros	-

A pantalla de creación dunha proba é a responsable de configurar os parámetros necesarios para a execución dunha nova proba. Como se pode apreciar na Figura C.13, o usuario dispón dun campo de texto para establecer o nome da proba, pode seleccionar os datasets a través dun selector onde aparecen todos os datasets dispoñibles, e pode consultar a lista de algoritmos seleccionados cos parámetros correspondentes a cada un.

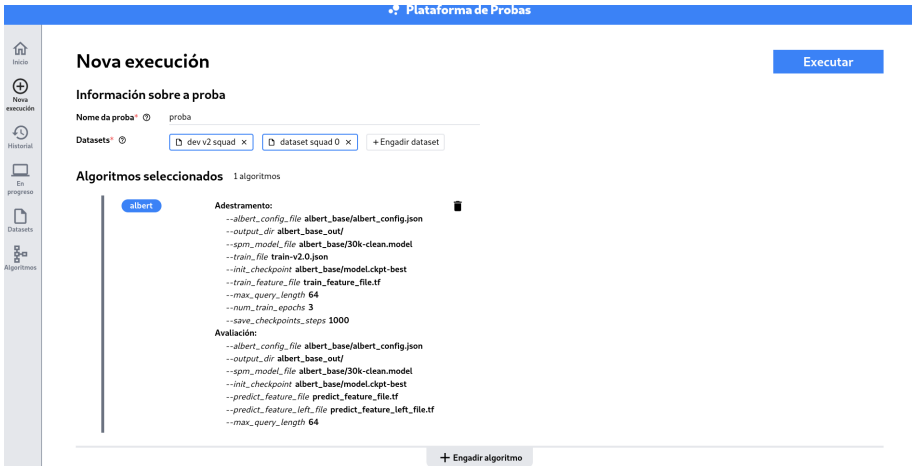


Figura C.13: Vista da pantalla de creación dunha proba.

Para engadir un algoritmo á proba, o usuario pode premer sobre o botón situado na parte inferior da pantalla (co texto “Engadir algoritmo”), o que

abrirá un modal. Este modal está formado por tres vistas. A primeira (Figura C.14) permítelle ao usuario seleccionar un algoritmo de entre os dispoñibles na plataforma. A segunda (Figura C.15) e terceira vista permiten configurar os valores dos parámetros para o adestramento e avaliación do algoritmo. O sistema permite avanzar á seguinte vista no caso de que os datos sexan válidos. Na última vista o usuario pode confirmar a súa selección e engadir o algoritmo configurado á proba.

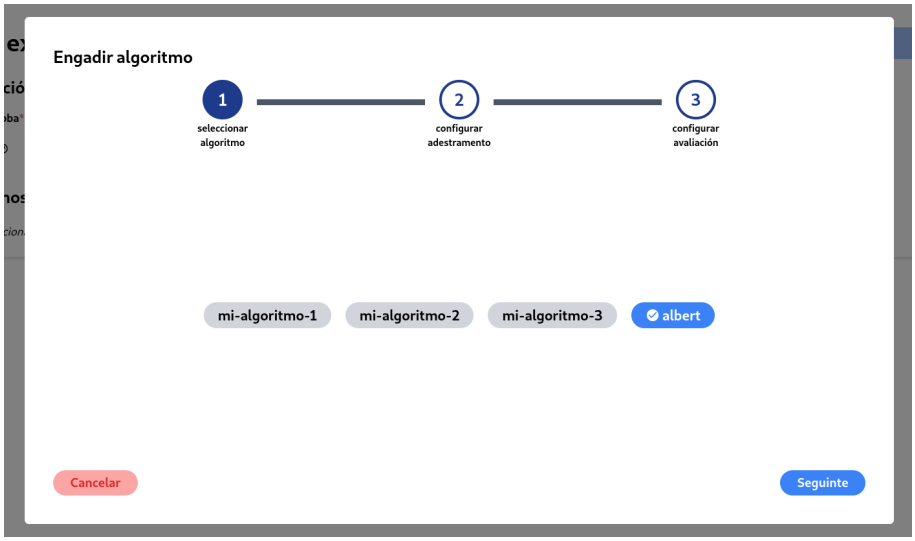


Figura C.14: Vista do modal de selección dun algoritmo (selección).

Figura C.15: Vista do modal de selección dun algoritmo (configuración).

Erros

Mensaxe	Descrición
[parámetros] é obrigatorio	Indica que un parámetro de configuración do algoritmo é obrigatorio e debe ser ter un valor asociado antes de continuar con outra operación.
[parámetro] ten que ser: [opcións]	Indica que un parámetro ten que ter un dos valores indicados nas opcións amosadas no erro.
[parámetro] ten que ser un decimal (float)	Indica que o valor dun parámetro de configuración do algoritmo debe ter un valor asociado a un número decimal.
[parámetro] ten que ser un enteiro (int)	Indica que o valor dun parámetro de configuración do algoritmo debe ter un valor asociado a un número enteiro (sen decimais).
Selecciona un algoritmo antes de continuar	O usuario debe seleccionar un algoritmo para poder configuralo para a proba nos seguintes pasos.

C.2.9. Pantalla de consulta dunha proba

Pantalla de consulta dunha proba	
Descrición	Permite consultar o estado e resultados dunha proba concreta.
Ruta	**/benchmarks/:id**
Parámetros	**:id** - identificador da proba

A pantalla de consulta dunha proba permite consultar a información dunha proba concreta que se está executando ou que xa rematou. Da proba, sempre se mostra a información básica da mesma, como o nome e os datasets seleccionados.

No caso de que a proba aínda non se atope finalizada, o usuario pode consultar as tarefas asociadas á mesma. Como se pode ver na Figura C.16, o usuario pode ver unha lista de tarefas, onde para cada unha se mostra o estado, a acción asociada e o seu identificador. As tarefas rematadas aparecen cunha cor verde, mentres que as que se atopan pendentes aparecen en azul.

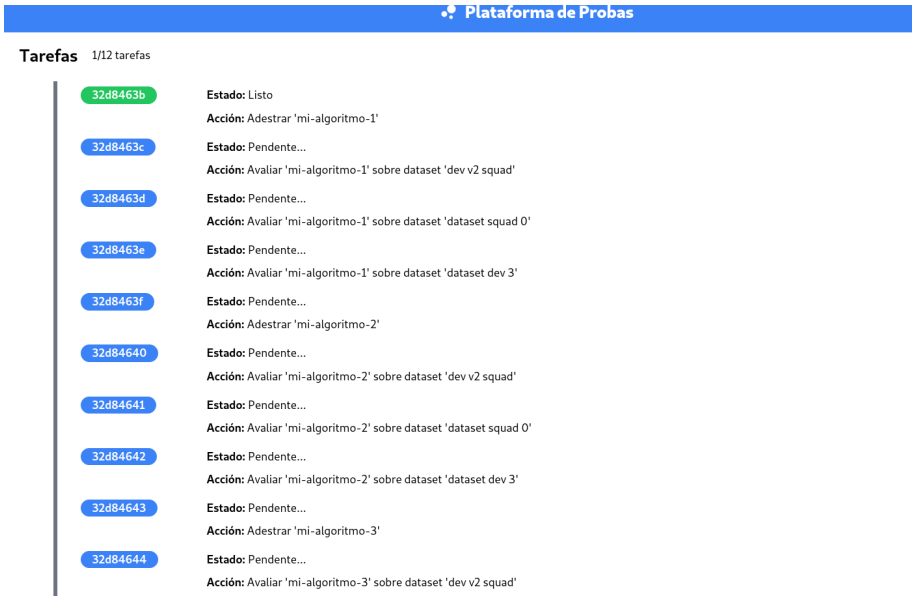


Figura C.16: Vista das tarefas dunha proba en execución.

Se a proba está finalizada pero houbo algún erro nunha tarefa, a vista será a mesma que no caso de non ter rematado pero a tarefa que fallou mostrarase nunha cor vermella indicando o erro.

No caso ter rematado a proba existosamente, o usuario pode seleccionar os algoritmos que se executaron na proba. Coa selección de cada algoritmo, engádense os resultados deste aos datos mostrados. A información dos resultados divídese en tres seccións: gráfica de resultados, táboas de métricas e test estadístico executado.

A primeira sección é un histograma que permite comparar os resultados dos algoritmos sobre os datasets, podendo agrupar por dataset ou por algoritmo mediante o desplegable que se atopa na parte superior da gráfica (Figura C.17).

Figura C.17: Vista dos resultados dunha proba (gráfica).

A segunda sección (Figura C.18) permite consultar os resultados dos algoritmos seleccionados, mostrando métricas como o valor *F1* e *Exact* das avaliacións realizadas.

Resultados *F1*

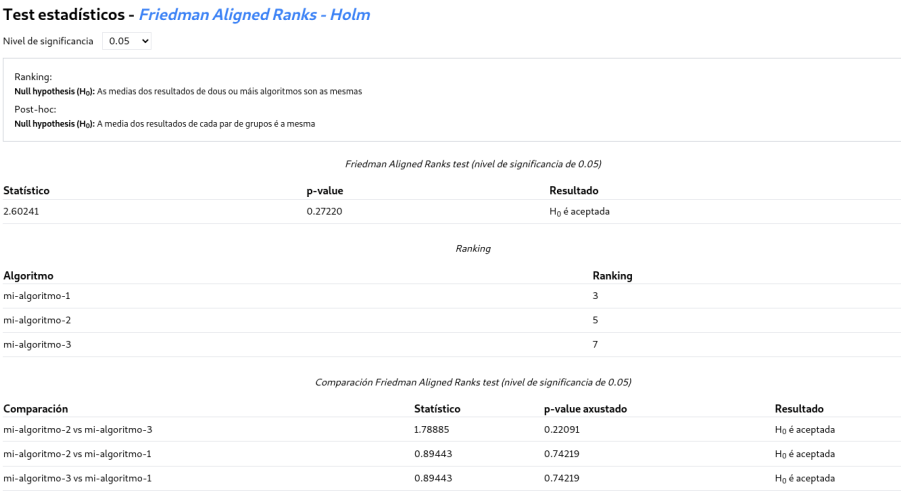
Algoritmo	dev v2 squad	dataset squad 0	dataset dev 3
mi-algoritmo-1	85.14	76.50	84.48
mi-algoritmo-2	85.25	75.84	86.69
mi-algoritmo-3	84.52	75.72	84.91

Resultados *EM*

Algoritmo	dev v2 squad	dataset squad 0	dataset dev 3
mi-algoritmo-1	85.14	76.50	84.48
mi-algoritmo-2	85.25	75.84	86.69
mi-algoritmo-3	84.52	75.72	84.91

Figura C.18: Vista dos resultados dunha proba (táboas).

Finalmente, a última sección mostra o test estadístico máis axeitado executado en función dos datasets e algoritmos seleccionados. O usuario pode seleccionar o nivel de significancia que desexa aplicar sobre o test estadístico, podendo así avaliar diversas hipóteses. Deste test estadístico executado, aparece o seu nome, o significado das hipóteses aplicadas e os resultados obtidos.



Test estadísticos - *Friedman Aligned Ranks - Holm*

Nivel de significancia 0.05

Ranking:
Null hypothesis (H_0): As medias dos resultados de dous ou máis algoritmos son as mesmas

Post-hoc:
Null hypothesis (H_0): A media dos resultados de cada par de grupos é a mesma

Friedman Aligned Ranks test (nivel de significancia de 0.05)

Statístico	p-value	Resultado
2.60241	0.27220	H_0 é aceptada

Ranking

Algoritmo	Ranking
mi-algoritmo-1	3
mi-algoritmo-2	5
mi-algoritmo-3	7

Comparación Friedman Aligned Ranks test (nivel de significancia de 0.05)

Comparación	Statístico	p-value axustado	Resultado
mi-algoritmo-2 vs mi-algoritmo-3	1.78885	0.22091	H_0 é aceptada
mi-algoritmo-2 vs mi-algoritmo-1	0.89443	0.74219	H_0 é aceptada
mi-algoritmo-3 vs mi-algoritmo-1	0.89443	0.74219	H_0 é aceptada

Figura C.19: Vista do test estadístico dunha proba.

Bibliografía

[1] What is Question Answering?. Artigo de MarketMuse. Consultado o 20 de maio do 2022 en *https://blog.marketmuse.com/glossary/question-answering-definition*

[2] Documentación de Python. Dispoñible en *https://www.python.org/*.

[3] Documentación de MongoDB. Dispoñible en *https://www.mongodb.com/*.

[4] Documentación Node.js. Dispoñible en *https://nodejs.org/es/*.

[5] JavaScript. Artigo de MDN Web Docs. Dispoñible en *https://developer.mozilla.org/es/docs/Web/JavaScript*.

[6] Typescript: JavaScript With Syntax for Types. Dispoñible en *https://www.typescriptlang.org/*.

[7] React - Una biblioteca de JavaScript para construir interfaces de usuario. Dispoñible en *https://es.reactjs.org/*.

[8] Documentación Git. Dispoñible en *https://git-scm.com/*.

[9] GitHub: Where the world builds software. Dispoñible en *https://github.com*.

[10] Bash. Artigo da Wikipedia. Consultado o 23 de maio do 2022 en *https://es.wikipedia.org/wiki/Bash*.

[11] Documentación Conda. Dispoñible en *https://docs.conda.io/en/latest/*.

[12] PyCharm: el IDE de Python para desarrolladores profesionales. Dispoñible en *https://www.jetbrains.com/es-es/pycharm/*.

[13] Visual Studio Code - Code Editing. Dispoñible en *https://code.visualstudio.com/*.

[14] Overleaf, Editor de LaTex online. Dispoñible en *https://es.overleaf.com/*.

[15] Almacenamiento personal en la nube de OneDrive. Dispoñible en *https://www.microsoft.com/es-es/microsoft-365/onedrive/online-cloud-storage*.

[16] draw.io – Diagrams for Confluence and Jira. Dispoñible en *https://drawio-app.com/*.

[17] Adobe XD — Herramienta rápida y potente de diseño y colaboración de experiencias e interfaces de usuario. Dispoñible en *https://www.adobe.com/es/products/xd.html*.

[18] Question Answering - Awesome List. Artigo de Awesome List. Consultado o 24 de maio do 2022 en *https://asmen.icopy.site/awesome/awesome-qa/*.

[19] The Stanford Question Answering Dataset. Raj Purkar. Consultado o 23 de maio do 2022 en *https://rajpurkar.github.io/SQuAD-explorer/*.

[20] I. Rodríguez-Fdez, A. Canosa, M. Mucientes, A. Bugarín, STAC: a web platform for the comparison of algorithms using statistical tests, in:Proceedings of the 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2015.

[21] Software for Question Answering Research. UKP-SQuARE. Consultado o 24 de maio do 2022 en *https://square.ukp-lab.de/*.

[22] Chatbot: ¿Qué es, para qué sirve y cómo funcionan?. Artigo de Bloo Media. Consultado o 30 de maio do 2022 en *https://bloo.media/blog/por-que-implementar-chatbot-en-tu-estrategia-de-marketing/*.

[23] API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos. Artigo de BBVA API_Market. Consultado o 30 de maio do 2022 en *https://www.bbvaapimarket.com/es/mundo-api/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos/*

[24] Secure Shell. Artigo da Wikipedia. Consultado o 30 de maio do 2022 en *https://es.wikipedia.org/wiki/Secure_Shell*

[25] SCP Linux – Securely Copy Files Using SCP examples. Artigo de Hayden James. Consultado o 30 de maio do 2022 en *https://haydenjames.io/linux-securely-copy-files-using-scp/*

[26] HTTP. Artigo de MDN Web Docs. Dispoñible en *https://developer.mozilla.org/es/docs/Web/HTTP*.

[27] Documentación de Express. Dispoñible en *https://expressjs.com/*.

[28] Documentación de Yup. Dispoñible en *https://github.com/jquense/yup*.

[29] Documentación do driver MongoDB para Node.js. Dispoñible en *https://www.mongodb.com/docs/drivers/node/current/*.

[30] Documentación de Validator.js. Dispoñible en *https://github.com/validatorjs/validator.js*.

[31] Documentación de Recharts. Dispoñible en *https://recharts.org/en-US/*.

[32] Documentación de Tailwind CSS. Dispoñible en *https://tailwindcss.com/*.

[33] Interfaces y Abstract Base Class (ABC). Artigo de El libro de Python. Dispoñible en *https://ellibrodepython.com/abstract-base-class*.

[34] Zhang, Zhuosheng and Zhao, Hai and Wang, Rui. Zhuosheng Zhang. "Machine Reading Comprehension: The Role of Contextualized Language Models and Beyond", 2020. Dispoñible en *https://github.com/cooelf/AwesomeMRC*.

[35] Zhang, Zhuosheng and Yang, Junjie and Zhao, Hai. "etrospective reader for machine reading comprehension", 2020. Dispoñible en *https://github.com/cooelf/AwesomeMRC*.

[36] Albert. Google Research. Dispoñible en *https://github.com/google-research/albert*.

[37] First in, first out. Artigo da Wikipedia. Consultado o 30 de maio do 2022 en *https://es.wikipedia.org/wiki/First_in,_first_out*.

[38] Documentación de React Json View. Dispoñible en *https://github.com/mac-s-g/react-json-view*.

[39] Documentación de React Router. Dispoñible en *https://reactrouter.com/*.

[40] Los 10 principios heurísticos de Jakob Nielsen. Artigo de Medium. Consultado o 31 de maio do 2022 en *https://medium.com/pildorasux/10-heuristicos-nielsen-abc9c6ad04c0*.

[41] Documentación de RabbitMQ. Dispoñible en *https://www.rabbitmq.com/*.

[42] Documentación de Kafka. Dispoñible en *https://kafka.apache.org/*.