Escuela Técnica Superior de Ingeniería Informática

GRADO EN INGENIERÍA INFORMÁTICA

CURSO ACADÉMICO 2020/2021

TRABAJO DE FIN DE GRADO

NUEVAS FUNCIONALIDADES EN JUEZ DE CÓDIGO ONLINE

Autor: Sudip Giri

Directores: José Manuel Colmenar Verdugo

Raúl Martín Santamaría

Agradecimientos

Agradezco a los tutores por la oportunidad de trabajar en este proyecto y por la guía y toda la ayuda proporcionada durante el desarrollo del proyecto.

Agradezco a mi familia por su apoyo y a mis compañeros por su colaboración durante la carrera universitaria.

Muchas gracias por todo y espero que la aplicación este lista pronto y se pueda aprovechar tanto para dar clases presenciales como online.

Siglas

- SW: Software
- web app: aplicacion web
- BBDD: base de datos
- front: front-end
- back: back-end
- API: interfaz de programación de aplicaciones
- REST: Transferencia de Estado Representacional
- CI: integración continua
- CD: despliegue continuo
- ICPC o ACM-ICPC: Competición Internacional Universitaria ACM de Programación
- GOOGLE: Global Organization of Oriented Group Language of Earth
- IA: Inteligencia Artificial
- CERN: Organización Europea para la Investigación Nuclear
- IOI: Olimpiada Internacional de Informática
- CEOI: Olimpiada Centro-Europea de Informática
- OIE: Olimpiada Informática Española
- ESO: Educación Secundaria Obligatoria
- SPOJ: Sphere Online Judge

- UVa OJ: Juez Automático en linea de la Universidad de Valladolid
- USACO: Olimpiada Informática de Estados Unidos
- COCI: Competencia Informática Abierta de Croacia
- DMOJ: El juez moderno (The Modern Judge en inglés)
- JOI: Olimpiada Informática Japonesa
- France-IOI: Olimpiada Informática Internacional Francesa
- TCP: protocolo de control de transmisión
- IP: Protocolo de Internet
- WWW o la web: Red informática mundial
- SMTP: protocolo para transferencia simple de correo
- FTP: protocolo de transferencia de ficheros
- P2P: red de pares, red entre iguales o red entre pares
- URL: localizador de recursos uniforme
- HTTP: protocolo de transferencia de hipertexto
- PHP: preprocesador de hipertexto
- SQL: lenguaje de consulta estructurada
- POO: programación Orientada a Objetos
- HTML: lenguaje de marcado de hipertexto
- CSS: hojas de estilo en cascada
- XHTML: lenguaje de marcado de hipertexto extendido
- JSF: JavaServer Faces
- XP: Programación Extrema
- UI: interfaz del usuario
- DAO: objeto de acceso a datos
- SoC: principio de separación de preocupaciones

- SUT: sujeto bajo prueba
- mvc: modelo-vista-controlador
- HW: hardware
- JSON: JavaScript Object Notation
- YML o YAML: no es un lenguaje de marcado (Ain't Markup Language en inglés)
- POA: programación orientada a aspectos
- JPA: API de persistencia de Java
- ORM: Object Relational Mapping
- MDB: Manejador de base de datos

Resumen

Este trabajo final de grado consiste en añadir nuevas funcionalidades y mejorarlo realizando tareas de desarrollo web y mantenimiento web en *back-end*.

Este TFG es uno de los TFG basados en el juez de código en linea denominado **iudex**. Este TFG es la continuación de los TFGs realizados por Pablo López Parilla sobre la creación de la aplicación y el orquestador (la parte de la aplicación que se encarga de ejecutar las entregas). Pablo en sus documentos detalla claramente todo el proceso seguido, las decisiones tomadas, las implementaciones realizadas y las características del juez.

El juez (iudex) es una aplicación web de código libre desarrollada para impartir clases de programación, algoritmos y bases de datos. Es una aplicación de código libre extensible, cualquiera puede utilizarlo y puede tener distintos tipos de usos.

Este documento esta compuesto por 5 capítulos: introducción (capitulo 1), objetivos (capitulo 2), marco teórico y el estado del arte (capitulo 3), descripción informática (capitulo 4) y conclusión (capitulo 5).

En el capitulo 1 se realiza una breve introducción donde se explican los objetivos del TFG, se habla del juez y se explica que son los jueces de código, se explica el funcionamiento básico de la aplicación y el estado actual de la aplicación.

En el capitulo 2 se describen los objetivos del trabajo y se comentan los objetivos secundarios alcanzados realizando este proyecto.

En el capitulo 3 se habla de la programación competitiva, los jueces en linea, se realiza una comparación del juez (iudex) con el resto de los jueces, se explican algunos conceptos básicos relacionados con la informática y programación, se habla de HTTP, se comentan las peticiones y los códigos de respuesta mas comunes, se habla de la ingeniería del SW, el desarrollo web y se finaliza el apartado hablando de mantenimiento , logs y pruebas.

En el capitulo 4 se habla del juez (iudex) explicando algunas de sus características y funcionamiento, estado inicial, novedades y se muestran algunas de las métricas de Sonarcloud para mostrar como ha ido mejorando la calidad durante la realización del proyecto, se explican las metodologías empleadas, se comentan las tecnologías y herramientas utilizadas como Spring, Docker, GitHub y SonarCloud y se finaliza el apartado explicando todas las tareas (sub-objetivos) realizadas y los problemas encontrados.

En el capitulo 5 se realiza un análisis de todo el proyecto y el proceso de desarrollo y se comentan algunas de las mejoras que se pueden realizar en la aplicación (trabajos futuros).

Palabras clave

El juez (the judge o iudex), aplicación web, web, internet, programación competitiva, programación, juez automático, Docker, contenedores, imágenes, desarrollo web, mantenimiento, logs, pruebas, funcionalidad nueva, mejoras, correcciones, lenguajes de programación, bibliotecas, frameworks, herramientas, tecnologías, metodologías de desarrollo, patrones de arquitectura software, funcionamiento, problemas encontrados, bugs, POO, Java, plataformas web, API, entradas, software, serialización, deserialización, HTML, CSS, JavaScript, JSON, Spring, Ingeniería de SW, Git/GitHub, metodologías ágiles, HTTP, TCP/IP, MySQL, REST, GOOGLE, CI/CD, etc

Índice general

Capítulo 1

Introducción

El trabajo final de grado (TFG) consiste en añadir nuevas funcionalidades en el juez de código online **el juez (the judge o iudex)** y mejorarlo.

1.1. El juez

El juez o **iudex** es una aplicación web de código libre (disponible en [1]) creada para impartir clases de programación, algoritmos y bases de datos. Actualmente esta en desarrollo y proporciona soporte a los lenguajes de programación Java, C, C++, Python y SQL.

Es un juez de código moderno (tendrá una plataforma moderna cuando la parte de *front* este terminada), seguro para la ejecución de código (el código se ejecuta de forma aislada), mantenible, fiable (solo utiliza imágenes de DockerHub para crear los contenedores), flexible (permite añadir nuevos lenguajes con facilidad) y escalable (permite modificar el numero de ejecutores en función de la demanda). Es uno de los pocos jueces que permiten crear problemas de bases de datos, actualmente solo soporta SQL.

Existen otros jueces de código en linea como TopCoder, Coderbyte, HackerRank, CodeChef, USACO, COCI, DMOJ, UVa OJ, etc.

En el desarrollo web se suele separar la aplicación en dos partes: *front* (*front-end*) y *back* (*back-end*). El *front* hace referencia a la parte visible de la aplicación (lo que ven los usuarios y el medio a través del cual interactúan con la aplicación) y el *back* a la parte invisible (la parte que no es visible para los usuarios).

1.2. Jueces de código

Un juez de código se utiliza en los concursos de programación competitiva para evaluar el código implementado por los participantes que compiten para resolver problemas de programación siguiendo un conjunto de especificaciones proporcionadas por los organizadores de los concursos. Existen muchos concursos de programación competitiva, el mas antiguo, mas grande y el mas prestigioso de ellos es el ICPC (consultar el apartado 3.1.4). En los concursos de programación la mayoría de los problemas suelen ser de programación y algoritmos.

1.3. Otros Jueces

La mayoría de los jueces son antiguos, existen desde hace mucho tiempo y utilizan un *front* sencillo. Están disponibles como plataformas web orientadas hacia los concursos de programación, se utilizan para practicar y organizar concursos de programación. Algunos jueces como CodeChef y Codeforces permiten impartir clases y otros como HackerRank permiten realizar entrevistas y contratar empleados. La mayoría (casi todas) de las plataformas no permiten resolver problemas de SQL, se centran en problemas de programación, estructuras de datos y algoritmos.

1.4. Funcionamiento básico de la aplicación

En la figura 1.1 se muestra el camino básico de la aplicación: El profesor crea un concurso y añade los problemas. El profesor gestiona el concurso, el decide que lenguajes se aceptan y añade los equipos de los alumnos. Cuando el profesor añade a los alumnos, ellos obtienen acceso al concurso y pueden realizar las entregas. Cuando los alumnos hayan subido las entregas y estas entregas hayan sido evaluadas, tanto el profesor como los alumnos podrán ver el marcador con la puntuación que ha obtenido cada equipo en ese concurso.

1.5. Alcance y uso

Es una aplicación de código libre, cualquier persona podrá utilizarla. Algunos de los usuarios posibles son institutos, universidades, profesores, comunidades, empresas, etc.

Se puede utilizar tanto para impartir clases (evaluar entregas o dar clases en linea) como para aprender, entrenar, realizar entrevistas y organizar concursos de programación o eventos similares.

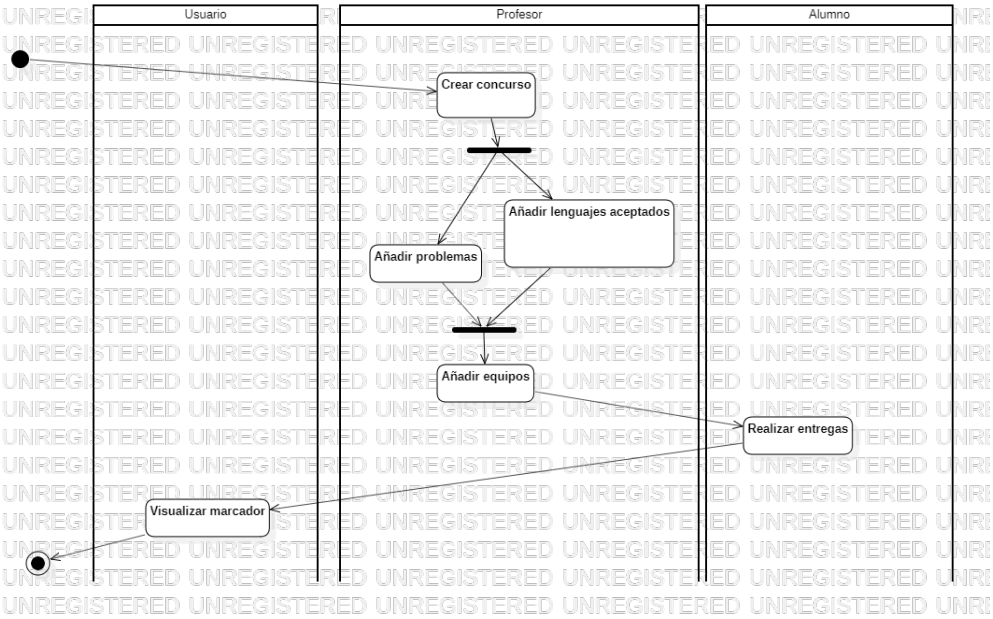

Figura 1.1: Funcionamiento básico de la app

1.6. Estado actual de la aplicación

El juez permite crear concursos, añadir problemas, realizar entregas, visualizar estadísticas de un concurso y mas acciones que se mencionan en el apartado 4.1.3.

La aplicación sigue en desarrollo, pero esta operativa y funcional. Cuando la parte de *front* este lista se podrá utilizar para impartir clases de programación, algoritmos y bases de datos entre otras cosas. El juez da soporte a los lenguajes Java, C, C++, Python y SQL.

Capítulo 2

Objetivos

En este apartado se explican los objetivos del proyecto y se comentan algunos objetivos secundarios cumplidos realizando este proyecto.

2.1. Objetivos

El objetivo del proyecto es añadir nuevas funcionalidades y mejorar la aplicación realizando tareas de mantenimiento y desarrollo web en *back-end*: añadir nuevas funcionalidades, modificar código y corregir fallos detectados. El objetivo se ha dividido en varias tareas que se pueden agrupar en tres categorías: funcionalidad nueva, mejoras y corrección de fallos.

El objetivo final del proyecto es acabar de desarrollar la aplicación para poder utilizarlo en el ámbito docente para dar clases de programación, algoritmos, bases de datos y otras asignaturas relacionadas con estas asignaturas donde sea necesario ejecutar el código desarrollado por los alumnos.

2.2. Objetivos secundarios

La realización de este proyecto ha implicado cumplir con los siguientes objetivos secundarios:

- aprender sobre la programación competitiva y los jueces de código
- aprender sobre los logs, pruebas y metodologías de desarrollo
- utilizar el conocimiento adquirido durante la carrera universitaria para desarrollar un proyecto

- ampliar conocimiento de maven
- ampliar conocimiento de Java y Spring
- aprender sobre las herramientas y tecnologías actuales como Docker (contenedores), Git/GitHub (control de versiones), Sonarcloud (CI/CD y análisis de código) , RabbitMQ (servicio de mensajeria), etc
- utilizar herramientas y tecnologías actuales
- utilizar metodologías actuales como metodologías ágil y CI/CD
- mejorar la calidad de un software aumentando la mantenibilidad y la fiabilidad

Capítulo 3

Marco teórico y estado del arte

En este apartado se habla sobre los jueces de código en linea y la programación competitiva. Se introducen algunos conceptos relacionados con la programación, el desarrollo SW y el desarrollo web y se comentan algunas metodologías de desarrollo y algunos patrones de arquitectura SW. Se finaliza el apartado hablando sobre mantenimiento, logs y pruebas.

3.1. Juez de código en linea

Un juez de código es un sistema de validación de problemas, con una serie de entradas y salidas, que se utiliza en los concursos de programación competitiva. Cuando un participante realiza una entrega (sube el código desarrollado), el juez lo evalúa el código compilando y ejecutando la entrega con los casos de prueba que posee e indica si la entrega es aceptable o no.

3.1.1. Programación competitiva

La programación competitiva es una competencia de programación que se realiza a través de internet o en una red local. Es reconocida y respaldada por varias compañías multinacionales de SW y de internet, como GOOGLE y Facebook. Hay varias organizaciones que organizan concursos de programación competitiva de forma regular. El concurso de programación mas importante es el ICPC.

3.1.2. Concurso de programación competitiva

Los concursos de programación competitiva son concursos en los que los participantes compiten para resolver problemas (la cantidad depende del tipo de competición, en las competiciones europeas y mundiales se proporcionan una gran cantidad de problemas, y del concurso) de programación. En un concurso de programación competitiva

por cada problema se proporciona un enunciado con una serie de especificaciones con los que los participantes deben diseñar e implementar un algoritmo eficiente que resuelva el problema.

El algoritmo se entrega al juez automático que dispone de una serie de casos de prueba con las soluciones esperadas, el juez compila y evalúa el algoritmo generando una salida que indica si el código subido es aceptable o no.

Los problemas típicos suelen ser problemas de lógica, matemáticas, algoritmos, estructuras de datos, búsquedas y optimización, tal y como se menciona en el libro Competitive Programming 3 de Steven Halim y Felix Halim que describen los problemas que se abordan en los concursos de programación IOI y ICPC.

3.1.3. Resultados

Los resultados de una entrega suelen ser de dos tipos: aceptado y no aceptado.

- Aceptado

  Indica que la solución es correcta, que las salidas son las esperadas y el algoritmo supera los casos de prueba preestablecidos (algunos o todos depende del concurso).

- No aceptado

  Indica que la solución es incorrecta. Las causas pueden ser de cualquier tipo: respuesta incorrecta, fallos de presentación, fallos de compilación, errores de ejecución, algún caso de prueba que no se supere o otras características como memoria consumida, el tiempo de ejecución, la complejidad algorítmica, etc.

  Los causas habituales suelen ser de tiempo de ejecución (supera el limite establecido), fallo de ejecución, fallos de compilación y casos de prueba que no se superan.

3.1.4. Algunos concursos de programación

Existen muchos concursos de programación competitiva. El concurso mas importante de todos ellos es el ICPC.

- ACM-ICPC

El Concurso Internacional de Programación Universitaria (ICPC) es el concurso más antiguo, más grande y más prestigioso del mundo. Fue celebrado por primera vez en 1970 por la sociedad de honor UPE de la universidad de Texas A&M [2].

Es un concurso anual dirigido a los estudiantes universitarios que se celebra por todo el mundo. Los estudiantes universitarios participan en grupos de tres personas representando su universidad para resolver problemas reales. La ICPC esta compuesta por distintas fases: locales, regionales y mundial. Es la competencia de programación más antigua, más grande y más prestigiosa del mundo.

- HashCode

  Es un concurso anual de programación organizado por GOOGLE para personas mayores de edad en su país o que hayan cumplido como mínimo los 18. Se puede participar en grupos de 2 a 4 personas para resolver problemas de programación propuesto por GOOGLE. Para cada ronda se facilita un problema. Para poder participar en el mundial de HashCode es necesario superar la fase de cualificación y tener como mínimo 2 miembros en el equipo. [3]

- Kaggle

  Kaggle es una empresa subsidiaria de GOOGLE. Kaggle ha realizado cientos de concursos de aprendizaje automático (Machine Learning) desde que se fundó la empresa. Las competiciones habituales de Kaggle son de aprendizaje automático con problemas de predicción difíciles con fines comerciales [4].

- Facebook Hacker Cup

  Facebook Hacker Cup es un concurso de anual de programación competitiva. Es una competición internacional organizada y administrada por Facebook. Facebook comenzó a organizar concursos de programación desde 2011 para identificar a los mejores talentos de ingeniería para posibles empleos en Facebook.

- Olimpiada Internacional de Informática (IOI)

  La Olimpiada Internacional de Informática es una de las Olimpiadas Internacionales de Ciencias, que tiene como objetivo potenciar el aprendizaje de la informática en alumnos de secundaria y preparatoria. La IOI es una competencia anual organizada cada año por uno de los países participantes que

normalmente envían una delegación de 4 alumnos y 2 adultos que los acompañan [5].

En la IOI los problemas están orientados a los algoritmos. Para participar en la IOI se debe clasificar a través de las olimpiadas nacionales como la Olimpiada Informática Española (OIE).

La Olimpiada Informática Española es el evento anual de programación más importante de España. Es un concurso individual y se celebra entre los meses de abril y junio. La OIE pone a prueba las habilidades de programación y los conocimientos de algoritmia de los estudiantes de la ESO y preparatoria de España.

3.2. Otros jueces

Existen varios jueces en linea disponibles como plataformas de programación competitiva accesibles en la web con problemas similares a los de las competiciones para que los usuarios puedan aprender, practicar y participar en los concursos.

3.2.1. Las 10 mejores plataformas de programación competitiva

Según el post de Daniel Borowski en freecodecamp [6] las 10 mejores plataformas de programación competitiva son TopCoder, Coderbyte, Project Euler, HackerRank, CodeChef, Exercism.io, Codewars, LeetCode, SPOJ y CodinGame.

Project Euler, CodeChef, Exercism no son jueces de código por tanto no se explican en este documento.

TopCoder

Topcoder (anteriormente TopCoder) es una empresa de crowdsourcing con una comunidad global abierta de diseñadores, desarrolladores, científicos de datos y programadores competitivos. Topcoder organiza el torneo anual Topcoder Open y una serie de eventos de programación competitiva regionales.

Topcoder permite practicar resolviendo desafíos y participar en concursos, guarda estadísticas de todos sus usuarios (talentos) y permite realizar búsqueda de talentos a las empresas. También ofrece servicios de gestión de proyectos en su plataforma. Hay dos formas de participar en los concursos: Web Arena (en desarrollo)

y Java Applet Arena. Para utilizar el applet es necesario tener instalado el entorno de ejecución de java (JRE).

En Topcoder los participantes participan en los concursos (suelen ser de una ronda) a través del sistema de Topcoder (Web Arena o Java Applet Arena) y deben superar un conjunto de fases: registro, asignación de sala, codificación, desafió y prueba. Los participantes cuando terminan de codificar deben compilar el código y realizar sus pruebas con el sistema de Topcoder en la fase de codificación. En la fase de prueba las entregas deben superar las pruebas ejecutadas por el sistema para que el participante pueda obtener puntuación. [7]

Coderbyte

Coderbyte es una aplicación web creada con el objetivo de ayudar a practicar y mejorar las habilidades de programación. Coderbyte ofrece una colección de desafíos de código, cursos de desarrollo web, kits de entrevistas y recursos de carreras a los usuarios (desarrolladores). Coderbyte ofrece servicio de búsqueda de talento a los empleadores. [8]

En Coderbyte los usuarios pueden resolver los desafíos en su editor en linea, donde ellos pueden ejecutar el código, ejecutar los casos de prueba y realizar la entrega.

Coderbyte da soporte a 17 lenguajes (C, C++, C#, Java, Python, SQL, etc), bibiotecas y framworks (Angular, Dart, React, etc), Nodejs (entorno de ejecución multiplataforma de código libre) y base de datos MySQL.

HackerRank

HackerRank es una empresa tecnológica que se enfoca en desafíos de programación competitiva tanto para desarrolladores como para empresas. HackerRank organiza concursos (a menudo denominados CodeSprints por HackerRank), muestra estadísticas de los desafíos y otorga insignias. HackerRank ofrece desafios, servicio de búsqueda de trabajo, certificaciones y permite participar en concursos a los usuarios (desarrolladores) y búsqueda de talento a las empresas.

En HackerRank los desarrolladores pueden practicar y participar en concursos para resolver desafíos de programación en una variedad de lenguajes. Pueden obtener certificaciones, buscar trabajo y preparar entrevistas de trabajo. Su funcionamiento es similar a Coderbyte, permite resolver desafios en su editor en linea, ejecutar el código y entregar. HackerRank permite subir código fuente, proporciona estadísticas y un foro de discusión por cada desafió.[9]

HackerRank da soporte a 6 lenguajes: C, C++, C# Java, Python y SQL.

Codewars

Codewars es una plataforma para aprender, entrenar y mejorar habilidades de programación. Codewars permite resolver kata (desafíos de programación centrados en mejorar la habilidad y la técnica) de muchos tipos y niveles de dificultad.

Codewars proporciona un editor en linea para resolver los problemas (kata) donde se puede ejecutar los tests y entregar. Por cada kata que se resuelve Codewars actualiza tu rango y proporciona puntos de honor. [10]

Codewars proporciona soporte a mas de 29 lenguajes C, C++, C#, Java, Python, SQL, etc.

LeetCode

LeetCode es una plataforma para mejorar habilidades de programación, ampliar conocimiento y preparar para entrevistas técnicas. Es una de las pocas plataformas que permiten resolver problemas de bases de datos.

LeetCode proporciona una serie de desafíos y problemas para practicar, organiza concursos semanales que otorgan puntos según el marcador, proporciona cursos o tutoriales de estructuras de datos y permite practicar entrevistas (con cuenta premium). LeetCode también proporciona servicio de búsqueda de talentos a las empresas. En LeetCode no se pueden ver las soluciones de otros usuarios pero proporciona estadísticas de las características del código. [11]

LeetCode proporciona un editor en linea para resolver los problemas que permite ejecutar el código y añadir algunas funciones de bibliotecas (Snippet) proporcionados por LeetCode.

LeetCode proporciona soporte a 14 lenguajes populares, algunos de los cuales son C, C++, C#, Java, Python, etc.

SPOJ

Sphere Online Judge (SPOJ) es un juez en línea que ofrece más de 13.000 desafíos de codificación. SPOJ organiza sus propios concursos y tiene un área de discusión de los desafíos. En su plataforma hay unas estadísticas (ranking) de todos los participantes.

Para evaluar las entregas SPOJ dispone de dos *clusters* llamados Pyramid y Cube, el creador del problema decide cual se utiliza. Para realizar las entregas SPOJ proporciona un editor en linea, también permite subir código fuente y entregar.[12]

SPOJ da soporte a mas de 45 lenguajes y compiladores como C, C++, C#, Java, Python, etc.

CodinGame

CodinGame es una plataforma un poco distinta comparada con el resto, se resuelven problemas codificando juegos. Es una plataforma web para programar mejorar habilidades programando juegos. También permite realizar búsqueda de trabajo a los desarrolladores y búsqueda de talento a las empresas. También proporciona tutoriales para aprender lenguajes de programación. [13]

Cada juego viene con una descripción, casos de prueba y un editor en linea donde se programa en uno de los lenguajes soportados por CodinGame. Aunque es distinta comparada con las demás sigue siendo popular entre los aficionados.

CodinGame da soporte a 25 lenguajes, algunos de los cuales son C, C++, C#, Java, Python, etc.

3.2.2. Los mejores jueces en linea disponibles para entrenar para IOI

Según el comentario de Andrew Fassi en quora [14] los mejores jueces en linea de programación competitiva y plataformas de entrenamiento son USACO, COCI, Codeforces, CodeChef, DMOJ, HackerRank y UVa Online Judge.

Se ha quitado France-IOI de la lista porque solo esta disponible en Francés.

USACO

USACO apoya la educación informática en los Estados Unidos y en todo el mundo mediante la identificación, motivación y capacitación de estudiantes de informática de secundaria en todos los niveles.[15]

En su plataforma web proporcionan cientos de horas de recursos gratuitos para entrenamiento, organizan concursos periódicamente (hasta 6 concursos al año), organizan campamentos de entrenamiento y seleccionan a los 4 mejores para las

olimpiadas.

USACO da soporte a los lenguajes de programación C++, Java and Python. Los participantes deben escribir su propio código y las competiciones son individuales para proporcionar un entorno de competición similar a la de IOI.

COCI

En la plataforma web de COCI [16] organizan concursos en linea para las olimpiadas. Se organizan 6 concursos cada año y permiten descargar los casos de prueba y las soluciones de los concursos pasados.

COCI da soporte a los lenguajes de programación C, C++, Java, Python y Pascal.

Codeforces

Codeforces es un proyecto de colaboración de los aficionados a los concursos de programación competitiva patrocinado por Telegragram. Es una red social dedicada a la programación y concursos de programación. Es una plataforma web para aprender y entrenar donde se realizan concursos de forma regular [17].

En su plataforma web proporcionan problemas para practicar, muestran las estadísticas, permiten realizar entregas, permiten crear y ver grupos, permiten impartir cursos, muestran los métodos de la API.

Codeforces da soporte a varios lenguajes como C, C++, Java y Python.

DMOJ

DMOJ es un juez en linea de código libre moderno. En su plataforma web se almacenan los problemas de CCC (concurso de programación Canadiense), CCO, COCI (competición informática abierta Croata), IOI (Olimpiada Internacional Informática), JOI (Olimpiada Informática Japonesa) y otros sitios. [18]

En su plataforma web proporcionan problemas en los cuales se pueden realizar entregas. También permiten ver el estado de las entregas de los participantes y el estado de las mejores soluciones. Permiten consultar la lista de usuarios (algunos con perfiles públicos donde se muestran las estadísticas de realización de entregas) y participar en concursos.

DMOJ utiliza tres jueces (Crwys, lowerth y Lletya) para ejecutar las entregas

(se puede observar en [19]) y da soporte a varios lenguajes, algunos de los cuales son C, C++, C# y Python.

UVa Online Judge

UVa Online Judge es un juez de código en línea para problemas de programación alojado por la Universidad de Valladolid con más de 4300 problemas y el registro de usuario está abierto a todos. Tiene mas de 100000 usuarios registrados. Y se pueden realizar entregas en los lenguajes de programación ANSI C (C89), C ++ (C ++ 98), Pascal, Java , C ++ 11 y Python. UVa OJ (UVa Online Judge) también organizan concursos con un tiempo limitado.

Actualmente UVa OJ es una plataforma para la creación de sitios web de jueces en linea como https://onlinejudge.org. Se esta actualizando y es de código libre [20].

3.3. Comparativa entre el juez y los jueces de código actuales

3.3.1. Alcance y uso

La mayoría de los jueces mencionados en este documento son jueces disponibles como plataformas web pero existen muchos mas como Kattis, DOMjudge y Acepta el reto. La mayoría de los jueces son antiguos, existen desde hace mucho tiempo y utilizan un *front* sencillo o complicado (no es intuitivo). Están orientadas hacia los concursos de programación, se utilizan para practicar y organizar concursos de programación. Hay algunas que se utilizan para enseñar como CodeChef y Codeforces y otros como HackerRank que permiten realizar entrevistas y contratar empleados.

El juez se ha desarrollado con el objetivo de impartir clases de programación, algoritmia y bases de datos. Aunque el objetivo inicial solo sea eso, se puede utilizar en distintos ámbitos. Es una aplicación moderna creada con herramientas y tecnologías actuales, cuando la parte de *front* este lista tendrá una interfaz moderna. Es una aplicación de código libre cualquier persona podrá utilizarla. Algunos de los posibles usuarios son institutos, universidades, profesores, grupos, comunidades, empresas, etc.

3.3.2. Entorno de ejecución

En los concursos de programación es necesario ejecutar el código de los participantes para poder valorar su trabajo. Uno de los problemas de la programación competitiva es la ejecución de código de forma segura. Algunas técnicas empleadas para la ejecución de código de forma segura son:

- Filtrar el código

  Consiste en escanear el código del usuario en búsqueda de vulnerabilidades. Es el sistema más rápido pero limita las instrucciones que pueden utilizar los participantes. Es ineficiente, no se pueden detectar todas las vulnerabilidades y añaden mas complejidad a los problemas.

- Acotar el entorno de ejecución

  Consiste en limitar el entorno de ejecución de las entregas como puede ser limitar el uso de la memoria, limitar los permisos o limitar el tiempo de ejecución. Es el método más utilizado pero tiene sus fallos.

- Utilizar virtualización

  Consiste en utilizar tecnologías o herramientas de virtualización como pueden ser las maquinas virtuales para la ejecución del código creando entornos de ejecución. Es lento pero permite ejecutar cada entrega en un entorno separado. Es la técnica mas moderna y necesita un sistema que se encargue de crear los entornos de ejecución para cada entrega.

La mayoría de los jueces como utilizan la técnica de acotar el entorno de ejecución (algunos de los cuales son TopCoder, SPOJ y Codeforces) para ejecutar las entregas de forma segura, pero pueden haber algunos problemas y fallos. El juez utiliza virtualización para la ejecución de las entregas, dispone de un orquestador de contenedores que se encarga de crear los contenedores para la ejecución de las entregas de forma aislada. Los contenedores son mas ligeros que las maquinas virtuales y ofrecen los mismos beneficios.

3.3.3. Lenguajes soportados

Los jueces normalmente soportan un conjunto de lenguajes (los mas comunes son C, C++, C#, Java y Python) y tardan bastante en añadir soporte para un lenguaje nuevo. El juez es una aplicación flexible (permite añadir lenguajes con facilidad) y extensible (permite aumentar las colas de mensajes y el numero de ejecutores

según la demanda). Es uno de los pocos que dan soporte a SQL como HackerRank, Codewars, LeetCode y SPOJ.

3.4. Conceptos básicos

3.4.1. Internet

«Es un conjunto de redes de comunicación interconectadas entre si utilizando el protocolo TCP/IP. Internet suele ser confundido con la web (World Wide Web), uno de los servicios que ha tenido mas éxito en internet. En internet existen muchos servicios y protocolos como SMTP, FTP, P2P, etc» [21].

3.4.2. Web

La web (World Wide Web o red informática mundial) es un servicio que funciona a través de internet por el cual se pueden transferir diversos tipos de datos a través de Protocolo de Transferencia de Hipertextos (HTTP). La web es un sistema de documentos interconectados por enlaces de hipertexto. Fue creada por Tim Berners Lee en 1990. Actualmente la web ha evolucionado bastante y se pueden hacer muchas cosas: consultar información, compartir información (documentos, vídeos, imágenes, etc), comprar, jugar, aprender, etc.

Para conectarse a la web es necesario tener un dispositivo con un navegador web. Un navegador web es un programa que permite conectarse a la web mediante el uso de la URL.

3.4.3. Lenguajes de programación

Los lenguajes de programación son las herramientas básicas de los programadores. Hay distintos tipos de lenguajes y de gran variedad. Algunos de los lenguajes de programación mas utilizados son C, Java y Python [22].

3.4.4. Lenguajes de marcado

Un lenguaje de marcado o lenguaje de marcas es una forma de codificar un documento que, junto con el texto, incorpora etiquetas o marcas que contienen información adicional acerca de la estructura del texto o su presentación. El lenguaje de marcas más extendido es el HTML, fundamento de la WWW.

3.4.5. Hojas de estilos

Las hojas de estilo son conjuntos de instrucciones, a veces en forma de archivo anexo, que se asocian a los archivos de texto y se ocupan de los aspectos de formato y de presentación de los contenidos: tipo, fuente y tamaño de letras, alineación y posicionamiento del texto, colores y fondos, etc. Las hojas de estilos mas utilizados en el desarrollo web son las hojas de estilo en cascada (Cascading Style Sheets o CSS), fundamento de la WWW.

3.4.6. Base de datos

Una base de datos (BBDD) es una colección organizada de información estructurada, o datos, típicamente almacenados electrónicamente en un sistema de computadora. La mayoría de las bases de datos utilizan lenguaje de consulta estructurado (SQL) para escribir y consultar datos.

Existen distintos tipos de bases de datos. Actualmente los tipos de bases de datos mas utilizados son las bases de datos SQL (como MySQL, mariaDB y PostgreSQL) y las bases de datos NO SQL (como MongoDB, Cassandra o Apache Cassandra y Redis).

3.4.7. Formatos de intercambio de datos

La serialización es el proceso de convertir el estado de un objeto en un formato que se pueda almacenar o transportar.

Los formatos de serialización o intercambio de datos Los formatos son archivos que utilizan las aplicaciones para trasportar gran cantidad de información. XML es el primero en utilizarse para este fin, muchas de las aplicaciones almacenan información mediante este formato. El formato de intercambio de datos mas utilizado en el desarrollo web es JSON, es un formato ligero de intercambio de datos.

3.4.8. API

API significa interfaz de programación de aplicaciones (Application Program Interface en inglés). Una API es una interfaz que facilita la comunicación entre dos programas distintos.

Las APIs permiten que las aplicaciones y programas se intercomuniquen entre sí. Es decir permiten utilizar aplicaciones antiguas sin realizar modificaciones para crear aplicaciones nuevas facilitando el trabajo de los desarrolladores.

3.4.9. REST API

«REST API es una API creada por el informático Roy Fielding en el año 2000, la cual se ajusta a los límites de la arquitectura REST o Transferencia de Estado Representacional (en inglés Representational State Transfer) y permite la interacción con los servicios web de RESTful» [23].

3.4.10. Bibliotecas

Una biblioteca o librería es un conjunto de clases, que poseen una serie de métodos y atributos que facilitan el trabajo de los programadores. En java hay distintas bibliotecas útiles como JUnit, Mockito, GOOGLE Guava, Jackson, etc.

3.4.11. Frameworks

Un framework, según Wikipedia, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Algunos de los frameworks útiles para el desarrollo web con java son Spring, Hibernate, Struts y JSF.

3.4.12. Sistema de control de versiones

El control de versiones es la gestión de los cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Una versión, revisión o edición de un producto, es el estado en el que se encuentra el mismo en un momento dado de su desarrollo o modificación.

3.4.13. CI/CD

La integración continua o CI es un proceso de automatización para los desarrolladores en el cual los cambios del código nuevo se diseñan, se prueban y se combinan periódicamente en un repositorio compartido.

La entrega continua, despliegue continuo o CD se refiere a los cambios que se implementan, se prueban (normalmente con pruebas automáticas) y se cargan en un repositorio (como GitHub o un registro de contenedor) para implementarlos en un entorno de producción en vivo.

3.5. Ingeniería de SW

La ingeniería de software es una disciplina formada por un conjunto de métodos, herramientas y técnicas que se utilizan en el desarrollo de programas informáticos. Los programas informáticos son conocidos como software (SW).

La ingeniería de software engloba toda la gestión de un proyecto. Desde el análisis previo de la situación, el planteamiento del diseño hasta su implementación, pasando por las pruebas recurrentes para su correcto funcionamiento.

3.5.1. Ciclo de vida

El proceso del desarrollo de software, también denominado ciclo de vida del desarrollo de software es un conjunto de fases que contribuyen de manera global a la obtención de los productos software necesarios para la gestión del proceso y el alcance de los objetivos finales.

El ciclo de vida de un SW esta compuesto por las siguientes fases:

- Especificación de requisitos

  La fase de especificación de requisitos consiste en obtener la descripción completa del sistema por ejemplo determinar el objetivo del sistema, determinar la necesidad de los usuarios, determinar la necesidad de los clientes, determinar las acciones que se pueden realizar, etc.

- Análisis

  La fase de análisis consiste en comprender los requisitos y determinar los elementos que intervienen en el sistema, la estructura, las relaciones, evolución temporal, etc. En esta fase se intenta determinar la descripción, las funcionalidades y el comportamiento del sistema.

- Diseño

  En la fase de diseño se utiliza la información recolectada en la fase de análisis. La principal tarea de la etapa de diseño es desarrollar un modelo o las especificaciones para el producto o componentes del Sistema.

- Desarrollo

  La fase de desarrollo también conocida como implementación consiste en

la creación del sistema (codificación) siguiendo las decisiones tomadas en la fase de diseño. En esta fase se escribe el código.

- Pruebas

  En la fase de pruebas se verifica el comportamiento del sistema. Las pruebas pueden ser tanto manuales como automáticas.

- Entrega

  En la fase de entrega se despliega o se entrega el producto SW al cliente. En esta fase termina el ciclo de desarrollo SW. El ciclo de desarrollo SW esta compuesto por las fases anteriores.

- Mantenimiento

  La fase de mantenimiento de SW consiste en realizar modificaciones de un producto de software después de su entrega o despliegue para añadir nuevas funcionalidades, mejorar el comportamiento del sistema, corregir errores, mejorar el rendimiento, etc.

- Retirada

  Es la ultima fase de ciclo de vida de un SW. El SW se considera muerto cuando se retira del mercado (no se realiza mantenimiento) o los usuarios dejan de utilizarlo.

3.5.2. Ciclo de desarrollo

El ciclo de desarrollo SW es una parte de ciclo de vida de un SW. En el ciclo de desarrollo se abarcan las fases de requisitos, análisis, diseño y implementación. El ciclo de desarrollo termina cuando se entrega el producto SW al cliente o se despliega la aplicación y empieza la fase de mantenimiento.

3.6. Desarrollo web

El desarrollo web es un término que define la creación de un sitio web para internet mediante el uso de tecnologías software de lado del servidor y el cliente.

El cliente es el usuario, normalmente utiliza el navegador para poder acceder y interactuar con la página web pero también puede ser un desarrollador que utiliza

el contenido de la página o aplicación web mediante una API para desarrollar una nueva aplicación web.

El desarrollo web se suele separar en dos partes: *front* o *front-end* y *back* o *back-end*.

3.6.1. Front

Front o *front-end* se refiere a la parte visible, todo aquello relacionado con la vista (la parte de la página web que se ejecuta en el navegador web, la parte visible al usuario). Suelen ser documentos HTML, hojas de estilos CSS y código JavaScript que se ejecuta en el navegador web.

3.6.2. Back

Back o *back-end* se refiere a la parte oculta, todo aquello que no sea la parte visible (la parte de la página o la aplicación web que se ejecuta en el servidor, la parte invisible para el usuario). Suelen ser códigos PHP, Java o Python y la base de datos (BBDD) entre otras cosas.

3.6.3. Página web

Una página web es un documento con texto, imágenes, vídeos, programas, enlaces, hipervínculos y muchas otras cosas, adaptada para la web. Para acceder a las páginas web y a la web es necesario utilizar el navegador web.

Las páginas web se crean con lenguajes de marcado como HTML o XHTML, hojas de estilo en cascada, scripts (JavaScript) y otros lenguajes de programación que se pueden ejecutar en el servidor web.

Las páginas web pueden estar almacenadas en un computador o en un servidor web remoto. El servidor web puede restringir el acceso únicamente a redes privadas, por ejemplo, en una intranet corporativa, o puede publicar las páginas en la WWW. El acceso a las páginas web se realiza mediante una transferencia desde servidores, utilizando HTTP.

3.6.4. Servicios web

«Un *web service* o servicio web es un tipo de tecnología que, a través de ciertos protocolos y estándares, habilita la comunicación entre distintas computadoras y permite intercambiar datos entre ellas» [24].

Existen distintos tipos de protocolos y estándares utilizados en los servicios web. El protocolo mas común es el HTTP.

3.6.5. Aplicaciones web

Una aplicación es un programa informático diseñado para realizar operaciones o funciones específicas. Generalmente, son diseñadas para facilitar tareas complejas y/o rutinarias.

Una aplicación web es una aplicación con página web que se ejecuta en un navegador. Los usuarios interactúan con la aplicación a través de la página web desde el navegador.

3.6.6. Plantillas web

Las plantillas son documentos HTML. En desarrollo web se utilizan sistemas de plantillas web para crear páginas web dinámicas. Un sistema de plantillas web esta formada por tres elementos: motor de plantillas, datos, plantillas (documentos HTML).

Motor de plantillas

El motor de plantillas o procesador de plantillas es un software diseñado para combinar plantillas con los datos para producir documentos. El motor de plantillas obtiene los datos (varia según el lenguaje de plantillas utilizado), modifica la plantilla y añade los valores y genera documentos (en el desarrollo web normalmente son documentos HTML).

3.7. Mantenimiento

El mantenimiento SW es el proceso de modificación de un componente o SW después de su entrega para corregir los defectos, mejorar los atributos y/o adaptarlo a los cambios. Es un proceso continuo, se realiza después de la entrega del SW hasta su retirada.

El mantenimiento es necesario para proveer continuidad de servicio, realizar cambios necesarios, dar soporte a las peticiones de los usuarios, seguir cumpliendo los reglamentos y las leyes (pueden cambiar) y facilitar los futuros trabajos de mantenimiento.

Tipos de mantenimiento:

- Mantenimiento correctivo

  El mantenimiento correctivo consiste en modificar el SW tras la detección de defectos, ambigüedades o errores. Los problemas pueden venir de los requisitos, de diseño o implementación. Los defectos pueden ser problemas de HW, SW o documentación.

- Mantenimiento adaptativo o evolutivo

  El mantenimiento adaptativo consiste en modificar el sistema para acomodarlo a cambios físicos del entorno. El mantenimiento adaptativo puede ser necesario por necesidades internas, por competencia y requisitos externos.

- Mantenimiento perfectivo

  El mantenimiento perfectivo consiste en mejorar el SW para cumplir con las nuevas necesidades o requerimientos de los usuarios o del negocio. El mantenimiento perfectivo incluye mejoras en el SW para aumentar la eficiencia, el rendimiento, etc. También incluye actividades necesarias para adaptarlo a nuevos requisitos.

- Mantenimiento preventivo

  El mantenimiento preventivo consiste en modificar el SW para mantener la fiabilidad y la eficiencia. En el mantenimiento preventivo se realizan revisiones y pruebas periódicamente.

  El mantenimiento preventivo reduce el riesgo al aumentar la calidad, reduce el coste de mantenimiento y aumenta la mantenibilidad del sistema.

- Mantenimiento estructural

  El mantenimiento estructural consiste en modificar la arquitectura interna del sistema para mejorar la mantenibilidad. El mantenimiento estructural se basa en el análisis de Pareto.

  Según Pareto en muchas situaciones alrededor de 80 % de los efectos son debidos a los 20 % de las causas. El análisis de Pareto consiste en identificar esos 20 %.

3.8. Logs

En informática se utiliza el término registro, log o historial de log para referirse a la grabación secuencial en un archivo o en una base de datos de todos los acontecimientos que afectan a un proceso particular. De esta forma constituye una evidencia del comportamiento del sistema.

Los logs son archivos que contienen información sobre los hechos producidos en un sistema. Los logs generalmente suelen ser guardados en los servidores pero es común mostrarlos por la consola para facilitar el trabajo de los desarrolladores.

3.8.1. Niveles de log

Existen distintos niveles de log que proporcionan información adicional sobre el suceso: TRACE, DEBUG, INFO, NOTICE, WARNING, ERROR, FATAL.

- TRACE: se utiliza durante el desarrollo para rastrear errores.
- DEBUG: se utiliza durante la depuración para registrar acciones o eventos que se van produciendo en el sistema.
- INFO: se utiliza para registrar las acciones de los usuarios o las acciones especificas del sistema.
- NOTICE: se utiliza en producción para registrar todas los eventos notables que no se consideren error.
- WARNING: se utiliza para registrar todos los eventos que potencialmente se pueden convertir en errores.
- ERROR: se utiliza para registrar los errores o los fallos del sistema.
- FATAL: se utiliza para registrar errores bloqueantes que pueden tener efectos secundarios en el sistema.

3.9. Pruebas

Las pruebas son conjunto de actividades que se realizan para comprobar el funcionamiento del sistema.

Existen distintas formas de clasificar las pruebas:

- según las características que prueban: pruebas funcionales y pruebas no funcionales

- según el SUT o sujeto bajo prueba: pruebas de unidad o pruebas unitarias, pruebas de componente, pruebas de integración y pruebas del sistema
- según la ejecución: pruebas manuales y pruebas automáticas
- según los objetivos: pruebas de aceptación, pruebas de humo, pruebas de sanidad
- según el nivel de abstracción: pruebas de caja blanca y pruebas de caja negra

3.9.1. Tipos de pruebas

- Pruebas funcionales

  Las pruebas funcionales se realizan para comprobar que el sistema funciona correctamente.

- Pruebas no funcionales

  Las pruebas no funcionales se realizan para comprobar las características del sistema como el rendimiento, carga, estabilidad, estrés, usabilidad, seguridad y mas cosas.

- Pruebas del sistema

  Las pruebas del sistema se realizan para comprobar que el sistema funciona correctamente, se prueba el sistema completo. Las pruebas del sistema son costosas, frágiles, poco flexibles y lentas (dado que se prueba el sistema completo).

- Pruebas unitarias

  Las pruebas unitarias se realizan para comprobar que las distintas partes del sistema funcionan correctamente, se prueba cada parte por separado. Las pruebas unitarias son rápidas y menos costosas. Pueden haber dependencias, para probar los SUT se utilizan dobles. Los dobles son objetos con comportamiento definido.

- Pruebas de integración

  Las pruebas de integración se realizan para comprobar que las distintas partes del sistema funcionan juntas correctamente. En las pruebas de integración se verifica que la comunicación entre los subsistemas sea correcto.

- Pruebas manuales

  Las pruebas manuales son realizadas por un usuario (normalmente un tester o programador) interactuando con el SUT. El usuario puede seguir algún orden guión.

- Pruebas automáticas

  Las pruebas automáticas son pruebas implementadas que se ejecutan en una maquina de forma automática mediante el uso de un SW. Las pruebas automáticas son mas eficaces y eficientes. Pueden ser costosas de implementar dependiendo del SUT.

- Pruebas de aceptación

  Las pruebas de aceptación son realizadas para validar que el SW cumple con los requisitos.

- Pruebas de humo

  Las pruebas de humo o smoke test son realizadas para verificar que el sistema este desplegado.

- Pruebas de sanidad

  Las pruebas de sanidad se realizados para verificar las funcionalidades básicas del sistema, los caminos críticos.

- Pruebas de caja negra

  Las pruebas de caja negra son pruebas ejecutadas por usuarios que desconocen la estructura interna del sistema. En las pruebas de caja negra se verifican que el sistema produzca las salidas correctas según las entradas.

- Pruebas de caja blanca

  Las pruebas de caja blanca son pruebas realizadas por los desarrolladores para comprobar que el sistema funciona correctamente, los usuarios (los desarrolladores) conocen la estructura interna del sistema.

Capítulo 4

Descripción Informática

En este apartado se habla sobre el juez (the judge o iudex) y se explican las tareas realizadas. Se comentan las tecnologías y herramientas utilizadas y se finaliza el apartado comentando los problemas encontrados en la fase de implementación.

4.1. El juez

El juez (the judge o iudex) es un juez de código automático en linea (online) moderno implementado empleando metodologías, técnicas, tecnologías y herramientas actuales como metodología ágil, REST API, contenedores, Docker, GitHub/Git, Swagger, RabbitMQ o rabbit, Sonarcloud, etc.

4.1.1. Implementación

El juez es una aplicación web implementada con HTML, CSS, SCSS, JavaScript, Java, Spring, SQL y MySQL. Actualmente sigue en proceso de desarrollo y da soporte a los lenguajes de programación Java, C, C++, Python y SQL.

4.1.2. Arquitectura

El juez esta compuesto por distintas partes tal y como me muestra en la figura 4.1: el front (actualmente en desarrollo), el back, el orquestrador o el ejecutor, el servicio de mensajería rabbit y la base de datos MySQL.

El juez funciona mediante el uso de contenedores. Para ejecutar la aplicación se crean contenedores mediante un Docker Compose que ejecuta cada parte de la aplicación en un contenedor y todos estos contenedores se agrupan en un contenedor grande que es el juez (the judge o iudex).

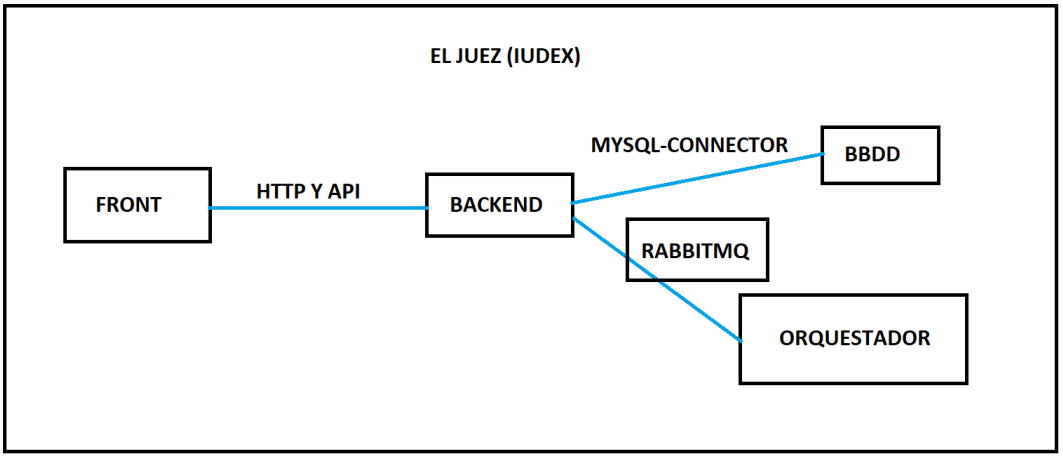

Figura 4.1: Arquitectura del sistema

Todas las operaciones entre front y el back se realizan a través de HTTP y los métodos de la API. En el back hay una parte que esta separada del resto, el orquestador o el ejecutor. En el back se utiliza el servicio de mensajería rabbit para la comunicación con el orquestador y mysql-connector para la conexión con la base de datos MySQL.

4.1.3. Características del juez

El juez o iudex es un juez de código en linea implementado utilizando metodologías, herramientas y tecnologías actuales como Docker, contenedores, Git, GitHub, tablero Kanban, RabbitMQ, Sonarcloud, etc. El juez esta en proceso de desarrollo. Es una aplicación web:

- segura para realizar entregas

  Gracias a Docker y los contenedores, el código se ejecuta de manera aislada en un contenedor

- flexible

  Se pueden añadir lenguajes nuevos de forma sencilla (siempre que sea posible y exista una imagen oficial en DockerHub para cuestiones de seguridad,

fiabilidad y mantenibilidad) siguiendo las instrucciones facilitados por Pablo Lopez Parilla en su TFG.

- extensible

  Gracias al servicio de mensajería rabbit y el orquestador de contenedores se pueden modificar las colas de mensajería y el numero de ejecutores para responder en función de la demanda

- mantenible

  Gracias a su arquitectura, la encapsulación, la modularización y la facilidad que proporciona Spring para el desarrollo backend se pueden realizar cambios de forma sencilla y eficiente.

Con el juez los organizadores, en este caso los profesores, pueden:

- crear, modificar y borrar concursos
- consultar lenguajes soportados por el juez
- añadir lenguajes permitidos en los concursos, deben ser los que son soportados por el juez
- crear, modificar y borrar problemas
- añadir problemas a los concursos
- quitar problemas de los concursos
- añadir, modificar o borrar casos de pruebas de los problemas
- verificar los resultados de los casos de pruebas con entregas de validación
- crear, modificar y borrar equipos
- crear, modificar y borrar usuarios
- añadir usuarios a los equipos
- quitar usuarios de los equipos
- consultar los resultados de los distintos intentos de todos los usuarios
- visualizar el marcador de un concurso

Con el juez los participantes, en este caso los alumnos, pueden:

- visualizar concursos con todos los problemas que lo componen
- visualizar problemas y realizar entregas
- consultar los resultados de los distintos intentos de todos los usuarios
- visualizar el marcador de un concurso

Es uno de los pocos jueces que permiten crear problemas de bases de datos, actualmente solo soporta SQL. La mayoría (casi ninguna) de las plataformas permiten resolver problemas de SQL, se centran en problemas de algoritmos.

4.1.4. Funcionamiento

El juez permite crear concursos, problemas, usuarios (participantes), equipos, realizar entregas, consultar marcador y mas cosas. El juez evalúa la entrega y genera una salida que indica si el código subido (entrega) es aceptable o no. El juez es capaz de evaluar entregas realizadas en los lenguajes de programación Java, C, C++, Python y SQL.

Para crear un problema es necesario utilizar un archivo zip con el formato indicado, con el enunciado (opcional), los resultados esperados y los casos de prueba.

Formato de fichero para la creación de un problema

El fichero zip debe contener:

- data: directorio con los casos de prueba
  - sample: directorio con casos de prueba visibles
  - secret: directorios con casos de prueba ocultos

  Cada entrada debe tener una salida y los nombre de cada entrada debe coincidir con su salida. Las entradas deben tener la extensión .in y las salidas deben tener la extensión .ans.

  El nombre se utiliza para saber que entrada corresponde con que salida y para crear un caso de prueba (Sample, clase que representa los casos de prueba).
- enunciado del problema en formato pdf (opcional)
- fichero de configuración: con valores de configuración para el problema
- fichero yaml con la configuración del proyecto (opcional)

Como ejemplo de los formatos se proporcionan las figuras 4.2 y 4.3.

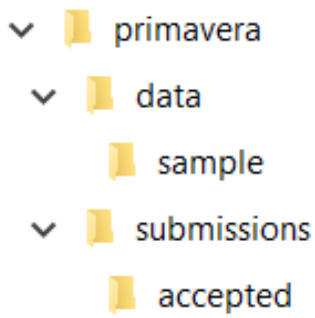

Figura 4.2: Ejemplo 1 de formato del fichero

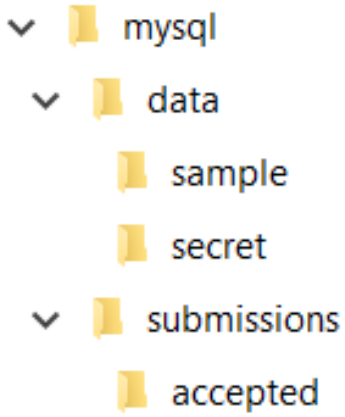

Figura 4.3: Ejemplo 2 de formato del fichero

Evaluación de las entregas

Cuando se realizan las entregas en el front, el front se comunica con el back mediante HTTP/IP y los métodos de la API REST. En el back se tranfieren los ficheros para que las ejecute el orquestrador. Para transferir los ficheros (código) se utiliza el servicio de mensajería rabbit (RabbitMQ). Los ficheros recibidos se ponen a la cola para su evaluación. Las entregas se ejecutan en el back creando contenedores. Cuando termina la evaluación se actualizan las bases de datos y se puede consultar el resultado de la ejecución en el front a través del marcador.

4.2. Estado inicial de la aplicación

A la hora de empezar el proyecto el juez estaba implementado como una aplicación mvc (modelo-vista-controlador) con contenedores para rabbit y el ejecutor. Permitía crear concursos, problemas, usuarios y equipos. La API estaba creada y se utilizaba un front sencillo con plantillas HTML que permitían realizar las operaciones básicas.

El código estaba subido a GitHub en un repositorio privado, actualmente esta en un repositorio publico denominado iudex.

4.3. Novedades

- pruebas de integración con JUnit y Mock, pruebas de humo y pruebas del sistema con RestTemplate y ObjectMapper

- modificación de logs existentes y incorporación de logs
- integración con Sonarcloud para la detección de fallos y vulnerabilidades
- soporte para C, C++ y SQL
- modificación de los mensajes de respuesta
- marcador para los concursos
- y mas cosas que se detallan en los apartados 4.7, 4.7.1 y 4.7.2.

4.3.1. Métricas de Sonarcloud

Como se muestra en las figuras 4.4 y 4.6 actualmente en el sistema hay 7 bugs, 1 vulnerabilidad, 14 problemas de seguridad, una deuda técnica de 3 días y 106 problemas para revisar y corregir.

A lo largo del proceso del desarrollo del proyecto se ha mejorado la calidad del sistema reducido la deuda técnica a 3 días, el numero de bugs a 7, las vulnerabilidades a 1 y los problemas de código a 106.

Métricas a la hora de integrar con Sonarcloud

Tal y como se muestra en la figura 4.5 el día 20 de junio justo después de integrar la aplicación con Sonarcloud se detectaron 20 bugs, 122 vulnerabilidades, 16 fallos de seguridad y 1269 problemas de código.

Métricas actuales

Tal y como se muestra en las figuras 4.4 y 4.6 actualmente hay 7 bugs, 1 vulnerabilidad, 14 fallos de seguridad y 106 fallos en el código.

4.3.2. Ejecución de pruebas

Las pruebas se ejecutan automáticamente (después de la integración con Sonarcloud) en cada pull request del repositorio de GitHub cada vez que se añaden los cambios con push.

Para ejecutar manualmente las pruebas:

- pruebas de integración: click derecho con ratón y run en Intellij o utilizando comandos de Maven

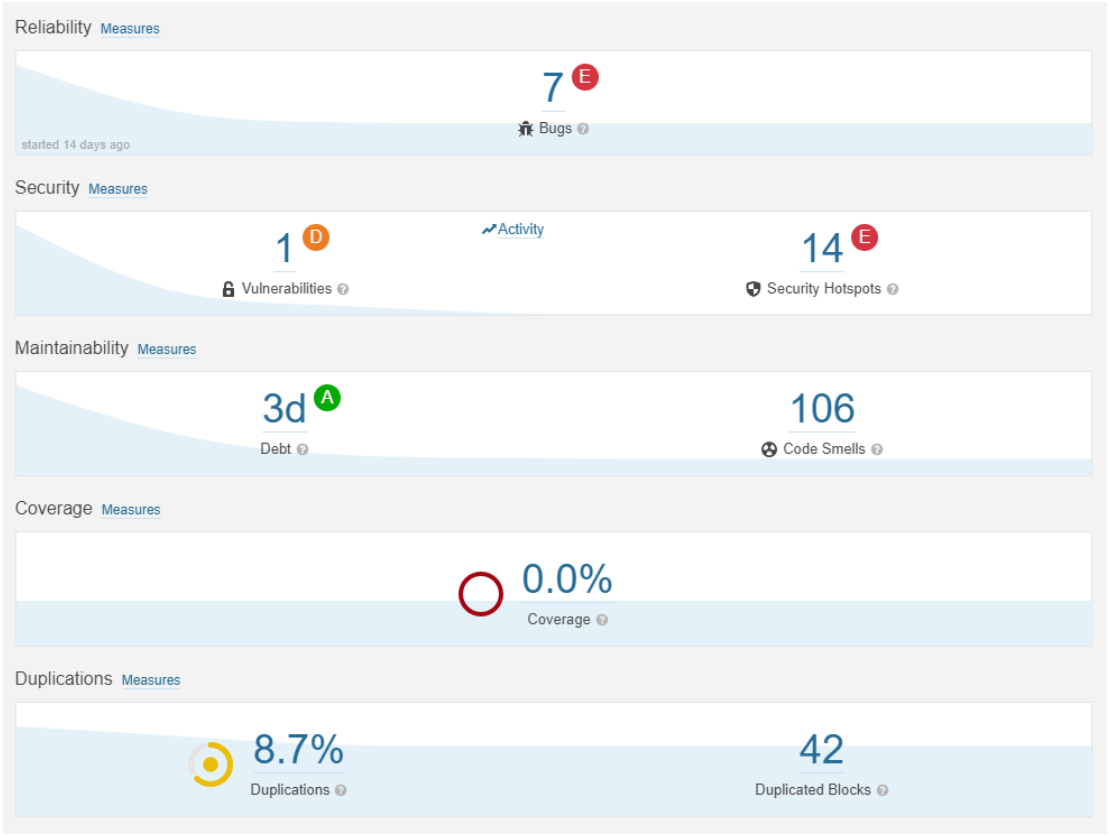

Figura 4.4: Estado actual de la aplicación

Figura 4.5: Métricas del día 20 de junio de 2021

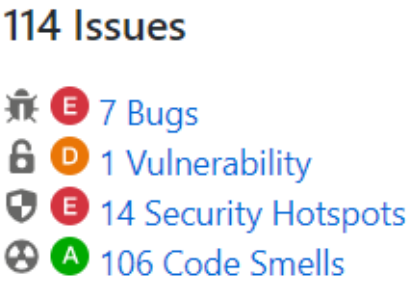

Figura 4.6: Métricas del día 1 de julio de 2021

- smoke test y pruebas del sistema: igual que con las pruebas de integración pero **es necesario ejecutar la BBDD y el rabbit antes para poder ejecutar las pruebas**.

Se han añadido 44 pruebas. Se ejecutan 33 pruebas sin fallos y se ignoran 11 por los problemas explicados en el apartado 4.8.3.

4.4. Metodología empleada

Para llevar a cabo el proyecto se han empleado las metodologías utilizado buenas practicas de programación y desarrollo como integración continua, evaluación continua y mejora continua con reuniones periódicas a través de teams y comunicación a través de teams y Outlook.

4.4.1. Metodología empleada de forma detallada

Para el desarrollo del proyecto se ha procedido de la siguiente manera:

1. empezar con alguna tarea existente

   La aplicación ya estaba en desarrollo. varias personas habían colaborado en el proyecto dejando una lista de tareas o modificaciones que se podían realizar.

   La idea era ir añadiendo nuevas funcionalidades, mejorar el código existente o corregir los fallos en función de las tareas apuntadas.

2. crear distintas ramas para separar las tareas

   La idea era separar las tareas en distintas ramas para organizar mejor el proceso de desarrollo y gestionar mejor los cambios realizados.

3. anotar los fallos o mejoras

   La idea era anotar los fallos y las mejoras que nos iban ocurriendo *issues* en GitHub para poder evaluar ese cambio y tener un seguimiento de las tareas realizadas y los cambios producidos.

4. utilizar el tablero Kanban

   La ideas es utilizar el tablero Kanban que proporciona GitHub como una de las herramientas de colaboración para mantener el seguimiento de la evolución del proyecto.

5. por cualquier duda o problema contactar por teams o Outlook

   La idea era mantenerse en contacto para realizar consultas, resolver dudas y colaborar. Para la comunicación se ha utilizado tanto teams como Outlook y para realizar reuniones periódicamente se ha utilizado teams.

6. crear pull requests

   La idea era crear pull request para una o varias tareas realizadas para poder evaluar los cambios realizados y aprobarlos antes de juntarlo con la branca *master* en GitHub.

4.5. Tecnologías y herramientas de desarrollo

Existen distintos tipos de herramientas y tecnologías útiles que facilitan el trabajo de los desarrolladores como interfaces de desarrollo integrado, sistema de control de versiones, herramientas para CI/CD, herramientas de gestión de dependencias, bibliotecas, frameworks, etc.

El juez es una aplicación web implementada en el lenguaje de programación java con lenguajes de marcado HTML y XML, las hojas de estilos CSS, formatos de intercambio de datos JSON y YAML, bases de datos MySQL, el sistema operativo Linux, bibliotecas de Java SpringFox, guava, snakeyaml, jackson, docker-java, mysql-connector, JUnit, mustache y bibliotecas habituales de Spring, el procesador JSON Jackson, el motor de plantillas Mustache, el framework Spring y las herramientas Swagger, Git, GitHub, Sonarcloud, Docker, RabbitMQ, Postman, el JDK o el kit de desarrollo para Java y la IDE Intellij IDEA.

A continuación se describirán algunas de las herramientas utilizadas: Spring, Docker, Git, GitHub, SonarCloud, Rabbitmq, Swagger, Postman, las bases de datos y las bibliotecas de Java utilizadas.

4.5.1. Spring

Spring es un framework de código abierto que facilita el desarrollo de aplicaciones Java. A diferencia de otros frameworks de Java, como Struts o Hibernate, Spring ayuda a estructurar aplicaciones completas de una manera sencilla, consistente y productiva. Spring permite utilizar otras bibliotecas y frameworks.

Spring facilita el proceso de desarrollo. Permite mantener el código limpio y organizado con la inyección de dependencias y programación orientada a aspectos.

La aplicación ha sido desarrollada con Spring.

Inyección de dependencias

La inyección de dependencias es un patrón de diseño orientado a objetos, en el que se suministran objetos a una clase en lugar de ser la propia clase la que cree los objetos. La inyección se refiere al subministro de la dependencia (el servicio) a un objeto (el cliente) que lo utiliza.

La clase que utiliza el objeto es el cliente y el objeto que se le subministra a la clase es el servicio.

Programación orientada a aspectos

«la Programación Orientada a Aspectos (POA) es un paradigma de programación que basa su filosofía en tratar las obligaciones transversales de nuestros programas como módulos separados (aspectos) para lograr una correcta separación de responsabilidades.»[25]

Una obligación transversal es aquella que se repite en varias partes de un programa independientemente de si las secciones en las que aparece tienen relación directa.

AOP no se utiliza mucho pero es la clave para entender el funcionamiento del Spring y sus componentes.

Spring Boot

Spring Boot es una herramienta que permite crear aplicaciones basadas en Spring con muy poca configuración. Permite hacer una configuración automática para que Spring funcione correctamente.

JPA

JPA o Java Persistence API es la API de persistencia desarrollada para Java. JPA es la propuesta estándar que ofrece Java para implementar un framework Object Relational Mapping (ORM), que permite interactuar con la base de datos por medio de objetos, de esta forma, JPA es el encargado de convertir los objetos Java en instrucciones para el Manejador de base de datos (MDB).

Jackson

Jackson es una librería de utilidad de Java para serializar (convertir un objeto Java en una cadena de texto JSON), y deserializar (convertir una cadena de texto JSON de un objeto en un objeto Java) objetos JSON.

Jackson es capaz de serializar y deserializar objetos de forma automática mediante el uso de los métodos clásicos de la programación orientada a objetos setters y getters.

4.5.2. Docker

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

Con Docker se pueden crear y utilizar contenedores de forma similar a las máquinas virtuales. Con estos contenedores se pueden crear, implementar, compartir y/o mover aplicaciones de un entorno a otro.

Docker se utiliza para crear los contenedores de la aplicación y desplegarlos.

Dockerfile

Es un archivo de texto plano con una serie de instrucciones para crear una imagen. Las imágenes se utilizan para crear contenedores y ejecutar esos contenedores como aplicaciones separadas.

Dockerfile se utiliza para crear las imágenes de los contenedores de la aplicación.

Docker Compose

Docker Compose es una herramienta desarrollada para ayudar a definir y compartir aplicaciones de varios contenedores. Docker Compose permite crear un archivo YAML para ejecutar la aplicación.

Docker Compose se utiliza para construir (build) y ejecutar la aplicación.

Contenedor

Un contenedor es una forma de empaquetar una aplicación con todas las dependencias y configuraciones necesarias.

Los contenedores de Docker son como máquinas virtuales ligeras y rápidas que permiten crear aplicaciones portables con todas las dependencias y las configuraciones necesarias.

Los contenedores son:

- entornos de ejecución para las imágenes (paquetes)
- entornos virtuales que se ejecutan en el host

Los contenedores se utilizan para separar las distintas partes de la aplicación.

Paquete

Los paquetes en programación permiten agrupar distintas partes del programa.

Los paquetes se pueden compartir y transportar de forma sencilla y facilitan la estructuración interna del programa (podemos crear estructuras jerárquicas).

En Docker los paquetes serían las imágenes.

4.5.3. Git / GitHub

Github es una plataforma para almacenar códigos con un sistema de control de de versiones propio (Git). Con GitHub se puede colaborar y compartir código de manera fácil. Git es el sistema de control de versiones que utiliza GitHub. Se puede acceder a Git desde el terminal y está disponible para Windows, Linux/Unix y macOS.

GitHub se utiliza para guardar el código y gestionar el proceso de desarrollo de la aplicación.

4.5.4. Sonarcloud

Sonarcloud es un servicio de análisis de código continuo en linea para la detección de fallos y vulnerabilidades disponible en la nube para otros servicios de la nube como GitHub.

Con Sonarcloud se puede obtener métricas sobre el estado actual del código, detectar donde hay fallos, *bugs*, vulnerabilidades, etc. Sonarcloud no solo muestra donde están los fallos sino que también proporciona información sobre el fallo detectado (figura 4.7). Además de eso proporciona métricas sobre cada fichero de código (figura 4.8).

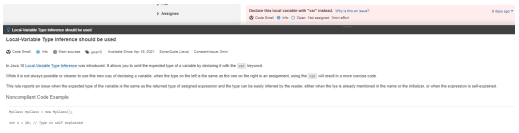

Figura 4.7: Información sobre el fallo

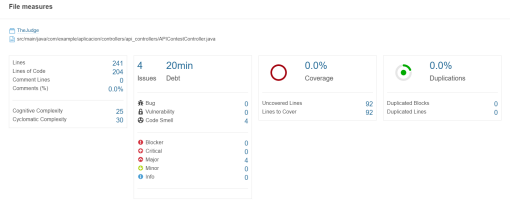

Figura 4.8: Métricas de la clase API-ContestController

Sonarcloud se ha utilizado para detectar los fallos y mejorar la calidad del código de la aplicación tal y como se puede apreciar con las métricas de Sonarcloud en el apartado 4.3.1.

4.5.5. RabbitMQ (rabbit)

Rabbit es un software de negociación de mensajes de código abierto que funciona como un middleware de mensajería.

Rabbit se utiliza para intercambiar datos entre el back y el orquestador, enviar y recibir los documentos entregados (ficheros de entrega) para su evaluación. Cuando el orquestador recibe los materiales crea un contenedor para ejecutar el fichero entregado con los casos de prueba. Cuando termina la ejecución rabbit devuelve el resultado de la prueba a backend que se encarga de actualizar la BBDD.

4.5.6. Swagger

Swagger es un conjunto de herramientas de software de código abierto para diseñar, construir, documentar, y utilizar servicios web RESTful.

Swagger se utiliza para documentar las API.

SpringFox

SpringFox es un conjunto de librerías o bibliotecas que permite generar automáticamente la documentación de la API en formato de Swagger.

4.5.7. Postman

Es un software de colaboración para desarrollar APIs que permite crear APIs de forma sencilla y rápida. [26].

Postman se ha utilizado para probar la funcionalidad de la API.

4.5.8. Bases de datos

Las base de datos utilizada en la aplicación es la base de datos relacionales MySQL.

MySQL es la BBDD de la aplicación, es donde se guardan todos los datos del sistema. MySQL se ejecuta en un contenedor por separado y se conecta con la aplicación de forma automática con Spring. Y también se utiliza para ejecutar las entregas de SQL (en un contenedor por separado con el orquestador) y evaluarlas.

4.5.9. GOOGLE Guava

GOOGLE Guava es es una serie de bibliotecas de código abierto para java desarrollado principalmente por los ingenieros de GOOGLE. Guava proporciona utilidades adicionales a los que proporciona java. [27].

4.5.10. Bibliotecas de java

Bibliotecas habituales de Spring

- spring-boot-starter: para crear aplicaciones básicas con Spring
- spring-boot-starter-web: para crear aplicaciones web con Spring
- spring-boot-devtools: para utilizar herramientas de desarrollo internas de Spring
- spring-boot-starter-test: para realizar pruebas con Spring
- spring-boot-starter-jpa: para gestionar las bases de datos con Spring y JPA

Otras bibliotecas

- docker-java: para Docker y contenedores
- spring-boot-starter-amqp: para el servicio de paso de mensajes por cola RabbitMQ
- springsox-starter: para documentar la API con Swagger
- mysql-connector: para la bases de datos MySQL
- snakeyaml: se utiliza para serializar texto yaml o yml en objetos java y deserializar objetos java en textos yaml o yml

- Jackson: para la conversión automática de objetos java a JSON y vice-versa

  Para Jackson se utilizan 3 bibliotecas: jackson-core, jackson-annotations y jackson-databind

- guava: no se utiliza de momento

4.6. Otras herramientas

Aparte de las herramientas de desarrollo indicadas en el apartado anterior se han utilizado Overleaf , Teams y Outlook.

4.6.1. Overleaf

Overleaf es un editor de texto en linea que se utiliza para escribir, editar y publicar documentos científicos con latex.

Se ha utilizado para redactar la documentación del TFG (este documento ha sido redactado con Overleaf).

Latex

Latex es un sistema de composición de texto. Permite crear cualquier tipo de documento presentable en papel o pantallas como manuscritos, cartas, artículos de revistas y tesis. Además de eso también permite separar el contenido y el formato del documento.

4.6.2. Teams

Teams o microsoft teams es una herramienta desarrollada para colaboración. En teams se puede:

- crear equipos
- realizar reuniones grupales
- realizar reuniones privadas
- enviar mensajes
- y mas cosas

Es una herramienta bastante útil que esta siendo utilizado tanto para impartir clases como para realizar proyectos y reuniones.

Se ha utilizado durante el desarrollo del TFG para realizar reuniones periódicas.

4.6.3. Outlook

Es una herramienta útil disponible como plataforma. Al principio es un servicio de mensajería pero con OutlookOffice facilita un conjunto de herramientas útiles y productivas como servicios.

Se ha utilizado el servicio de mensajería de Outlook para estar en contacto durante todo el desarrollo del proyecto.

4.7. Tareas

El objetivo del proyecto se ha dividido en varios sub-objetivos (tareas) que se distribuyen en las siguientes categorías:

- funcionalidad nueva

  Las funcionalidades nuevas son tareas que se realizan para añadir una funcionalidad nueva en la aplicación como puede ser añadir soporte para un lenguaje nuevo como C, C++ y SQL.

- mejoras

  Las mejoras son tareas que se realizan para mejorar algo ya existente como puede ser añadir pruebas para poder probar distintas partes del sistema, añadir logs para detectar más fácil y rápidamente los fallos y reformatear (simplificar) una parte del código.

- corrección de fallos

  La corrección de fallo esta compuesto por las tareas realizadas para resolver algún fallo encontrado.

Los problemas encontrados en cada una de las tareas están explicadas en el apartado de problemas encontrados.

4.7.1. Funcionalidad nueva

Las funcionalidades nuevas son tareas llevadas a cabo para añadir algo nuevo en la aplicación.

Las funcionalidades nuevas añadidas son las siguientes:

- añadir soporte para C
- añadir soporte para C++
- añadir soporte para MySQL / MariaDB
- añadir entradas nuevas en la API
- añadir marcador

Añadir soporte para C

El juez tenía soporte para Java y Python, permitía realizar entregas en Java y Python. Esta tarea consistía en añadir soporte para C, es decir permitir que se puedan realizar entregas en C.

La dificultad estaba en saber en qué lugares había que realizar los cambios.

No hubo ningún problema a la hora de realizar esta tarea. Además de eso se había proporcionado un documento con las instrucciones para añadir soporte para un lenguaje nuevo.

Añadir soporte para C++

Esta tarea era similar a la anterior. Había que seguir exactamente los mismos pasos.

No hubo ningún problema a la hora de realizar esta tarea.

Añadir soporte para MySQL / MariaDB

Esta tarea era similar a las dos anteriores pero con algunos problemas a la hora de crear un contenedor de MySQL y ejecutar las entregas en SQL.

Los problemas están detallados en la parte de problemas encontrados.

Añadir nuevas entradas en la API

Esta tarea esta compuesta de diversas tareas, se han juntado como una sola tarea en el documento por que el objetivo de esas tareas son similares.

- añadir entradas para samples

  Los samples son datos de pruebas de los problemas que se ejecutan cuando se añade una nueva entrega para verificar que la entrega sea la correcta.

  La tarea consistía en añadir nuevas entradas en la API para permitir que se pueda añadir, modificar y borrar los datos de pruebas de un problema.

- añadir entradas para lenguajes

  La tarea consistía en añadir nuevas entradas en la API para poder consultar, añadir y borrar los lenguajes de programación (tipos de ficheros) soportados por el programa.

  Las entradas para poder añadir y borrar nuevos lenguajes no se han podido añadir en la API por la complejidad que suponían realizar esas tareas. Actualmente se realizan manualmente. Para poder añadir las entradas habría que automatizar todo el proceso de creación de un lenguaje nuevo y eliminación de un lenguaje que puede ser bastante complejo.

  La entrada para consultar los lenguajes soportados por el programa (el juez) se ha añadido sin problemas.

- añadir entradas nuevas para los equipos

  La tarea consistía en añadir nuevas entradas para poder añadir y borrar equipos (participantes) de un concurso.

  Para poder añadir y eliminar equipos de un concurso se han añadido operaciones para poder añadir y borrar los equipos en masa. Poder añadir o borrar varios equipos a la vez mejora la experiencia del usuario.

- añadir entradas nuevas para lenguajes de cara al concurso

  La tarea consistía en permitir que los creadores del concurso o los usuarios con los permisos necesarios puedan especificar y borrar los lenguajes son soportadas en ese concurso con la restricción de que esos lenguajes sean uno de los lenguajes

soportados por el juez.

Para poder realizar esta tarea se han añadido operaciones para poder añadir y eliminar lenguajes en masa (varios a la vez).

A la hora de añadir estos cambios en la API se encontraron algunos bugs y fallos que están detallados en la parte de correcciones de fallos. Estos bugs y fallos fueron corregidos sin problemas.

Añadir marcador

Esta tarea consistía en realizar las modificaciones necesarias para poder obtener el marcador de un concurso.

Para poder añadir el marcador se ha añadido una entrada en la API y modificado tanto el ContestService como el controlador para que devuelva el marcador.

No hubo ningún problema a la hora de realizar esta tarea.

4.7.2. Mejoras

Las mejoras son tareas llevadas a cabo para mejorar algo ya existente en la aplicación.

Las mejoras realizadas son las siguientes:

- establecer de forma automática la ruta de conexión con Docker
- actualizar problema en vez de crear uno nuevo cuando se crea más de una vez a partir del mismo zip
- refactorizar (simplificar) código
- mejorar el sistema de logs
- añadir logs
- añadir pruebas
- reemplazar el uso de null por Optional
- añadir métodos equals y hashcode a las entidades
- reformatear ficheros y eliminar comentarios que sobran

- utilizar transaccional en las operaciones en masa
- utilizar Set en lugar de List
- utilizar herramienta de integración continua como Jenkins
- añadir métodos de consulta existsBy a los repositorios

Establecer de forma automática la ruta de conexión con Docker

Cuando se pone en marcha la aplicación en el local, el sistema se pone en contacto con Docker de manera remota a través de una URL. Esta URL se modificaba manualmente.

Esta tarea consistía en mejorar este proceso. Habían dos opciones:

1. obtener la URL del fichero *properties* de Spring
2. añadir código para seleccionar la URL de forma automática según el sistema operativo

Elegí la segunda opción porque si se añadía la ruta en el fichero *properties* el problema no se resolvía, solo se cambiaba el sitio donde se realizaba ese cambio de URL.

La tarea fue realizada sin problemas con las propiedades del sistema que proporciona java [28].

No hubo ningún problema a la hora de realizar esta tarea.

Actualizar problema en vez de crear uno nuevo cuando se crea más de una vez a partir del mismo zip

A la hora de crear problemas en el juez, se podía crear problemas desde un zip. El problema era que si se intentaba crear otro problema pues el juez lo permitía aunque no hubiera ninguna diferencia entre estos problemas.

Este problema se podía resolver de dos formas:

1. no permitir que se pueda hacer esto mostrando un aviso o lanzando una excepción
2. actualizar el problema antiguo

Elegí la segunda opción porque eso permitiría modificar los problemas de forma sencilla solo adjuntando el zip que contiene los datos correctos.

No hubo ningún problema a la hora de realizar esta tarea.

Refactorizar (simplificar) código

Esta tarea esta compuesta de distintas subtareas que se llevaron a cabo desde el inicio hasta el final del proyecto. Las tareas de simplificación de código son las siguientes:

- refactorizar el uso de InNOut

  InNOut es la clase que se utilizaba para representar los datos asociados a un problema. Cada problema tenia asociado una serie de datos. Estos datos representan los casos de prueba.

  Los casos de prueba pueden ser ocultos o visibles. Los casos de prueba ocultos no están visibles, solo el creador del problema o concurso pueden acceder a ellos. Y los casos de prueba visibles no están ocultos (están visibles), cualquier persona que tenga acceso al problema puede acceder a ellos.

  Para realizar la tarea se ha:

  - modificado el nombre de las clase InNOut a ProblemData que es más descriptivo
  - añadido un enum para representar el tipo de dato
  - eliminado los atributos que representan los datos del problema
  - añadido un atributo único para representar los datos del problema

  No hubo ningún problema a la hora de realizar esta tarea.

- simplificar ProblemData y cambiar el nombre a Sample

  En el juez las entradas y las salidas de los casos de prueba se almacenaban en ProblemData de forma separada, es decir se creaban dos objetos para cada caso de prueba. Para cada entrada y salida se utilizaba la clase ProblemData y la única diferencia entre la entrada y la salida era el valor del atributo text (un String para guardar el valor de la entrada o la salida).

  La tarea consistía en juntar las entradas y las salidas para representar cada caso de prueba con un solo objeto y renombrar la clase a Sample.

Para realizar la tarea se ha:

- modificado el nombre de ProblemData a Sample
- renombrado el atributo text a inputText
- añadido un nuevo atributo de tipo String outputText
- eliminado el enum ProblemDataType
- añadido un atributo de tipo booleano para saber si es un caso de prueba oculto o visible
- renombrado el repositorio para que coincida con la clase
- modificado código en los sitios donde se utilizaba ProblemData para adaptarlo a los nuevos cambios

Fue una tarea sencilla pero rigurosa. Había que realizar cambios en distinto sitios.

Mejorar el sistema de logs

Los logs son información que se registran o se guardan sobre los acontecimientos que se van produciendo en un sistema o una aplicación. Sirven para detectar y resolver problemas (fallos, errores, comportamiento sospechoso) de forma más rápida.

En la aplicación ya se utilizaban logs. Esta tarea consistía en mejorar el sistema de logs existente y añadir nuevos en los sitios donde era necesario o no se había añadido.

Existen distintos tipos de logs y los tipos de logs mas comunes son:

- trace
- debug
- info
- warning
- error
- fatal

Como hay distintos tipos y cada uno con un significado propio decidimos utilizar solo debug, info y error.

No hubo ningún problema. Fue una tarea rigurosa, había que buscar documentación sobre los tipos de logs para saber en que caso seria mejor utilizar cada tipo de log.

Añadir pruebas

Esta tarea consistía en añadir código para poder realizar pruebas para comprobar el funcionamiento del sistema. El objetivo era poder automatizar las pruebas añadiendo pruebas unitarias, pruebas de integración y pruebas software.

En el juez las clases están interconectadas, dependen una de la otra por tanto no podemos añadir pruebas unitarias. Se ha añadido pruebas de integración, prueba de humo (smoke test) y pruebas del sistema utilizando mocks y RestTemplate [29] para comprobar el funcionamiento básico del sistema.

RestTemplate es una clase que proporciona Spring para utilizar servicios REST. RestTemplate proporciona métodos útiles que nos permiten realizar peticiones, obtener respuesta, configurar las peticiones y acceder a las distintas partes del mensaje de respuesta. **RestTemplate acabara siendo deprecated en las futuras versiones de Spring, se recomienda utilizar Webclient.**

Los problemas están detallados en la parte de problemas encontrados. Fue una tarea bastante rigurosa. Había que buscar documentación y automatizar las pruebas.

Se han añadido 44 pruebas, 8 de las cuales son parte de las pruebas del sistema separadas en trozos. Se ejecutan 33 pruebas y se ignoran 11 por los problemas explicadas en la parte de problemas encontrados. **Hay que añadir mas pruebas**.

Reemplazar el uso de null por Optional

En java se utiliza null para representar objetos que no tienen valor. Cuando se realiza alguna consulta a un objeto de tipo null lanza un NullPointerException.

Esta tarea consistía en evitar el uso de null con las consultas que se realizaban a las bases de datos mediante la API.

Java facilita una clase Optional que se puede utilizar con cualquier objeto java. En otras palabras, se puede obtener el Optional de cualquier objeto y también

obtener el objeto de su Optional.

La clase Optional proporciona métodos útiles como:

- isEmpty: indica si el objeto esta vació, es decir no tiene ningún valor asociado
- isPresent: indica si el objeto tiene algún valor asociado, es decir que no está vacío

No hubo ningún problema a la hora de realizar esta tarea.

Añadir métodos equals y hashCode a las entidades

Los métodos equals y hashCode son métodos de la clase Object de java y se utilizan para comparar objetos en java.

Esta tarea consistía en redefinir los métodos equals y hashCode de java para poder comprobar las instancias de las entidades con su id.

No hubo ningún problema a la hora de realizar esta tarea.

Reformatear ficheros y eliminar comentarios que sobran

En el código existían códigos comentados que no se utilizaban y comentarios que sobraban.

Esta tarea consistía en eliminar esos comentarios y reformatear los ficheros para mantener el código limpio y bien formateado.

Intellij proporciona una opción para cambiar el formato de código que optimiza los imports y redefine los tabulados entre otras cosas.

No hubo ningún problema. Esta tarea la fue realizada poco a poco mientras se realizaban las otras tareas.

Utilizar transaccional en las operaciones en masa

A la hora de añadir operaciones en masa como añadir lenguajes permitidos o añadir varios usuarios a un concurso no se utilizaba transaccional. A veces si alguna de las operaciones falla puede ser necesario deshacer las operaciones que no han fallado, como por ejemplo los posibles problemas que podrían dar si no se podía añadir todos los lenguajes o equipos indicados.

Esta tarea consistía en utilizar transaccional en las operaciones en masa (bulk operations) para resolver este tipo de problemas.

Transaccional es una opción que nos facilitan los SGBD para realizar este tipo de operaciones.

No hubo ningún problema a la de realizar esta tarea.

Utilizar Set en lugar de List

En la aplicación se utilizan listas (List) en todos los atributos. Hay casos en los que no se necesita utilizar una lista, se puede utilizar conjuntos (Set).

Esta tarea consistía en reemplazar el uso de List a Set donde fuera posible en las entidades por tres razones:

1. no había ninguna razón concreta para utilizar List
2. las operaciones de añadir y borrar con Set O(1) tienen menos coste que con List O(n)
3. se guardan elementos sin repetición

Se ha reemplazado el uso de List por Set en los atributos de las entidades de la aplicación sin problemas.

No hubo ningún problema a la de realizar esta tarea.

Utilizar software de pruebas como Jenkins

En la aplicación se habían añadido varias pruebas y era necesario ejecutar las manualmente cada vez que se realizaban modificaciones en la aplicación.

Esta tarea consistía en utilizar herramienta que facilitara la ejecución de las pruebas, una herramienta que ejecute las pruebas automáticamente cada vez que se realizan los cambios y se suben a GitHub.

Para las pruebas se ha utilizado Sonarcloud, una herramienta potente que además de ejecutar las pruebas automáticamente se puede integrar fácilmente con los repositorios de GitHub y realiza análisis de código proporcionado distintos tipos de métricas como mantenibilidad, bugs, deuda técnica, etc.

Con Sonarcloud se detectaron 20 bugs y 122 vulnerabilidades, 16 posibles fallos

de seguridad y 1269 posibles problemas de código (code smells). Con todas estas métricas había una deuda técnica de 17 días.

Los problemas detectados con Sonarcloud se fueron corrigiendo poco a poco mientras se realizaban las otras tareas logrando conseguir una aplicación con 8 bugs, 1 vulnerabilidad, 14 fallos posibles de seguridad y 106 posibles problemas de código (code smells) con una deuda técnica de 3 días.

No hubo ningún problema a la de realizar esta tarea.

Añadir métodos de consulta existsBy a los repositorios

En la aplicación no se utilizaban métodos de consulta se utilizaban métodos de búsqueda en situaciones donde había que consultar si un objeto estaba guardado en la base de datos (repositorios) o no.

Esta tarea consistía en añadir métodos de consulta a los repositorios para esas situaciones.

Se han añadido los métodos existsBy a los repositorios que indican según los parámetros si ese dato esta almacenado y se han modificado el código para reemplazar los métodos de búsqueda por los métodos de consulta en los lugares necesarios.

No hubo ningún problema a la de realizar esta tarea.

Completar Docker

La aplicación se ha desarrollado con Docker y contenedores. Para ejecutar la aplicación era necesario ejecutar rabbit y la base de datos manualmente.

Esta tarea consistía en evitar la repetición de todo este proceso y ejecutar la aplicación de forma sencilla.

Para realizar esta tarea:

- se ha creado un Docker Compose que ejecuta la base de datos, rabbit y el backend
- se ha creado un Dockerfile para ejecutar el backend que reconstruye el jar y se ejecuta ese jar (aplicación)

Con todo este proceso se consigue ejecutar la aplicación en un contenedor con todos los elementos necesarios. Se le ha proporcionado privilegios adicionales al contenedor

de backend para poder crear y manejar el resto de contenedores necesarios para la aplicación.

Para ejecutar la aplicación hay que ejecutar el comando docker-compose up –build o docker compose up –build.

Los problemas se detallan en el apartado de problemas encontrados.

4.7.3. Corrección de fallos

Las correcciones son las tareas que se realizan para arreglar algún fallo existente de la aplicación.

Las correcciones realizadas son las siguientes:

- visualizador de pdfs falla cuando no hay pdf
- añadir un aviso o lanzar excepción en caso de un sistema operativo (so) desconocido
- mejorar validación del problema
- problema sin nombre
- no se puede eliminar problema
- no se puede eliminar concurso
- mejorar el uso de Optional
- faltan datos en la API
- evitar el uso de cascade
- NullPointerException cuando falla la creación de un concurso
- modificar la relación entre Result y Sample

Visualizador de pdf falla cuando no hay pdf

Cuando se crea un problema se puede añadir un documento pdf (enunciado) para mostrarlo a la hora de mostrar el problema. Aunque el problema no tenia ningún pdf la aplicación cargaba el visualizador de pdf.

La tarea consistía en no cargar el visualizador de pdf si el problema no tenia

un pdf.

Para realizar la tarea había que modificar la clase ProblemController y la API para evitar que se cargara el visualizador del pdf.

Los problemas están detallados en el apartado de problemas encontrados.

Añadir un aviso o lanzar excepción en caso de un SO desconocido

A la hora de realizar la automatización de la ruta de Docker no se tuvo en cuenta el caso de un SO desconocido.

Se había contemplado solo los siguientes Sistemas Operativos:

- Windows
- Linux/Unix
- macOS

Esta tarea consistía en añadir código para notificar si se trataba de un SO desconocido y parar la ejecución, ya que el juez por ahora solo se puede ejecutar en estos sistemas operativos Windows, Linux/Unix y macOS.

Para realizar la tarea había que modificar el método que proporcionaba la ruta según el sistema operativo para que lanzará una excepción si se trataba de un so desconocido.

No hubo ningún problema a la hora de realizar esta tarea.

Mejorar validación del problema

En el juez se pueden crear problemas a partir de los ficheros zip que contienen los datos necesarios. Pero había un fallo que permitía crear problemas con ficheros zip que no contenían casos de prueba.

Para resolver este problema se ha añadido código para verificar si existe algún caso de prueba e interrumpir el proceso si no existía.

No hubo ningún problema a la hora de realizar esta tarea.

Problema sin nombre

En el juez se pueden crear problemas a partir de los ficheros zip que contienen los datos necesarios. Había un fallo que permitía que cualquier zip arbitrario (un zip sin nombre “.zip.zip” o ".zip") era aceptado para crear un problema.

Esta tarea consitia en no permitir que se puedan crear problemas sin nombre (, ".zip", ".zip.zip").

Para realizar esta tarea solo había que verificar que el formato del zip era correcto. En caso de que no fuese correcto se detenía la creación del problema mostrando un aviso.

No hubo ningún problema a la hora de realizar esta tarea.

No se puede eliminar problema

Este problema se debe a las relaciones que hay entre las distintas entidades que no permitían el borrado.

La tarea consistía en averiguar porque no se podía borrar los problemas y arreglarlo. Al principio se creyó que la causa era el uso de Optional ya que no se había utilizado get en algunos sitios (se olvido y no se detecto el fallo). Pero el problema estaba en las relaciones entre las entidades.

Para resolver el fallo había que investigar las relaciones entre las entidades y realizar modificaciones sin romper el programa.

La tarea fue un poco costosa pero no hubo ningún problema.

No se puede eliminar concurso

Esta tarea es similar a la anterior y la causa también era la misma. Había problemas con la declaración de la relación entre las entidades con JPA.

Esta tarea era similar a la anterior y fue resuelta sin problemas.

No hubo ningún problema a la hora de realizar la tarea.

Mejorar el uso de Optional

A la hora de utilizar Optional para evitar el uso de null y posible situación de NullPointerException se olvido utilizar get en las operaciones con la bases de datos y esto provoco algunos fallos.

La tarea consistía en revisar el uso de Optional y arreglar fallos si había.

No hubo ningún problema a la hora de realizar la tarea.

Faltan datos en la API

El juez utiliza una API para el funcionamiento. Pero como la API estaba en proceso de desarrollo se utilizaban directamente los controladores y las plantillas.

A la hora de trabajar en el front (otro compañero) faltaban datos que se necesitaban para mostrar en el front:

- no hay submissions en Team a través del TeamController

  En un concurso los equipos (participantes) realizan entregas a los problemas de ese concurso. Cada equipo tiene una lista de entregas realizadas, una lista de concursos en los que ha participado, etc. Estos datos no se podían obtener en el front.

  Esta tarea consistía en averiguar donde estaba el fallo y arreglarlo. El objetivo era mostrar los datos que faltaban en el front.

  El fallo estaba en que no se almacenaban los objetos en los repositorios aparte de los posibles problemas de configuración de las relaciones entre las distintas entidades con la entidad Team (equipo).

  Con Spring y JPA almacenando un objeto (por ejemplo una instancia de clase Team que contiene los concursos y las entregas) debería de ser suficientes, el resto debería de guardarse automáticamente (el concurso y las entregas) o debería de mostrar algún fallo de foreign key. Como Spring no muestra ningún fallo es posible que las relaciones no estén bien configuradas.

  El fallo fue arreglado al principio con cascade, pero cascade provocaba acciones no deseadas. La corrección de los problemas con cascade están apuntados como otra tarea. **Actualmente falta verificar que las relaciones entre las distintas entidades estén bien configuradas, se ha creado una issue**

en GitHub para resolverlo en el futuro (cuando se pueda).

Aparte del problema de cascade no hubo ningún problema.

- falta ProblemAPI en SubmissionAPI

  En la api las entregas (clase interna Submission y clase para la API SubmissionAPI) la clase que se utiliza para la API no contenía ningún dato sobre el problema. Desde front cuando se obtenida la lista de entregas realizadas no se podía saber a que problema pertenecían.

  Esta tarea consistía en arreglar ese fallo. El objetivo era conseguir que desde front cuando se obtienen las entregas se pueda saber a que problema pertenecen.

  La tarea se ha resuelto añadiendo el dato que faltaba en la clase SubmissionAPI. Se ha añadido un ProblemAPI (clase que se utiliza para los problemas para la API) con los mínimos datos necesarios

  No hubo ningún problema a la hora de realizar esta tarea.

- nuevos parámetros en la API Contest

  En el front faltaban algunos parámetros de los concursos (clase interna Contest y clase que se utiliza para la API ContestAPI). En el front no se podía obtener la fecha de inicio y fin del concurso y tampoco se podía añadir participantes a los concursos.

  Esta tarea consistía en añadir los parámetros que faltan en ContestAPI. El objetivo era añadir los parámetros que faltaban y realizar los cambios necesarios para que desde front se pueda añadir usuarios a los concursos.

  La tarea de añadir nuevo endpoint para poder añadir usuarios a los concursos esta explicada en el apartado de funcionalidades nuevas.

  Los parámetros que faltaban se han añadido utilizando la clase LocalDateTime de java en la clase Contest y long en la clase ContestAPI. Se ha utilizado long en ContestAPI porque Jackson no podía convertir de LocalDateTime a JSON y tampoco de JSON a LocalDateTime.

  No hubo ningún problema a la hora de realizar estas tareas.

Evitar el uso de cascade

En el juez es necesario conservar algunos datos aunque otros datos con los que están relacionados se borran por eso no convenía utilizar la opción CascadeType.ALL.

Cuando los problemas de las relaciones de la entidad Teams con el resto fueron corregidos para poder mostrar los datos que faltaban de los equipos en la api se utilizó CascadeType.ALL que realizaba todas las operaciones en cascada.

Esta tarea consistía en evitar el uso de CascadeType.ALL. Había dos opciones utilizar CascadeType con otras opciones que nos facilita spring aparte de CascadeType.ALL o resolver el problema sin utilizar cascade.

La tarea ha sido realizada sin utilizar cascade.

No hubo ningún problema a la hora de realizar esta tarea.

NullPointerException cuando falla la creación de una entrega

En el código había un fallo, cuando no se podía crear la entrega (Submission) se intentaba mostrar en id del objeto, que no se había guardado en la base de datos, en los logs y se provocaba un NullPointerException.

Esta tarea consistía en arreglar ese fallo.

La tarea fue realizada modificando el log mostrando solo el mensaje del error producido a la hora de crear y ejecutar las entregas (Submission).

No hubo ningún problema a la hora de realizar esta tarea.

Modificar la relación entre Result y Sample

Después de simplificar el código y renombrar ProblemData a Sample se detecto un problema con la relación establecida entre la entidad Result y Sample. Según las especificaciones del sistema un Sample (entrega) puede estar relacionado con varias o ningún Result pero se habia definido con una relación OneToOne.

Esta tarea consistía en corregir ese fallo modificando la relación de OneToOne a ManyToOne.

Para resolver el fallo se ha modificado la relación OneToOne a ManyToOne en la entidad Result.

No hubo ningún problema a la de realizar esta tarea.

4.8. Problemas encontrados

Los problemas encontrados a la hora de realizar las tareas se detallan en esta sección.

4.8.1. Añadir soporte para MySQL/MariaDB

A la hora de añadir soporte para MySQL/MariaDB se produjeron varios problemas:

- problemas de conexión con la base de datos

  A la hora de crear el contenedor para MySQL/MariaDB no se podía establecer conexión con él, se producía un fallo. El problema estaba en el Dockerfile, estábamos modificando el entrypoint por lo que el servicio de la base de datos no se llegaba a arrancar.

  La solución a este problema la encontró Raúl (codirector del TFG) con la ayuda de Isaac. La solución consistía en arrancar el contenedor de MySQL sin modificar el entrypoint. Y ejecutar de forma automática los comandos que se había que ejecutar cuando el contenedor estaba preparado desde la clase DockerContainerMySQL, el lugar desde donde se arrancaba y se detenía el contenedor para MySQL.

- problemas a la hora de ejecutar las instrucciones

  A la hora de evaluar una entrega las instrucciones de MySQL no se llegaban a ejecutar cuando se creaba un problema y se realizaba la entrega. El problema era que el contenedor no estaba preparado para poder ejecutar las instrucciones, se estaba arrancando.

  El problema fue detectado por Raúl (el codirector del TFG) y se resolvió añadiendo tiempos de espera.

- contenedor en ejecución en segundo plano sin detenerse

  A la hora de corregir el fallo de la conexión con Docker se provoco otro fallo que hacia que el contenedor de MySQL se quedara ejecutando en segundo

plano. Debido a este fallo el contenedor de rabbit se quedaba bloqueado y no se podía ejecutar nuevas entregas.

Este fallo fue resuelto por Raúl junto con el fallo anterior añadiendo un tiempo de espera y deteniendo la ejecución del comando para que el contenedor pueda detenerse cuando había terminado de ejecutar las instrucciones.

- falta de documentación

  A la hora de crear los Dockerfile y arreglar algunos de los problemas que hubo con Docker intente consultar la documentación oficial de Docker y de las clases de java y las interfaces que trabajaban en Docker.

  En la documentación oficial de Docker falta la sección donde explica como se puede trabajar con Docker desde java. Las clases y las interfaces que trabajan con Docker provienen de GitHub pero no incluyen documentación y tampoco los comentarios que suelen venir en las clases e interfaces de java que indican la tarea que realiza cada método y que son los parámetros que recibe.

  Debido a esto tuve problemas para resolver los fallos que hubo a la hora de añadir soporte para MySQL/mariaDB. Después de darme cuenta que mirar tanto en las implementaciones como en la documentación no me ayudaría busque algunos ejemplos en internet. Pero la mayoría no contenían explicación y generalmente los que contenían información trabajaban con los comandos desde un terminal.

4.8.2. Actualizar problema en vez de crear uno nuevo cuando se crea más de una vez a partir del mismo zip

A la hora de crear un problema desde un zip la segunda vez se creaba otro problema distinto con el mismo nombre.

Para arreglar este fallo se intento modificar el método (addProblemFromZip) que se encargaba de la creación de los problemas pero no se hacía de forma directa, era muy lioso. Se llamaba a otro método que creaba el problema y ese método llamaba al método anterior, era difícil de entender y no se podía modificar fácilmente.

Como la parte de añadir problemas era muy lioso, este problema fue resuelto añadiendo un if else para ver si era un problema existente e invocar al método (updateProblem) que se utilizaba para actualizar los problemas antes de la llamada al método que se encargaba de crear el problema.

4.8.3. Añadir pruebas

A la hora de añadir pruebas hubo algunos problemas:

- problemas a la hora de probar algunos métodos de la API

  Para implementar las pruebas del sistema se ha utilizado RestTemplate. Con RestTemplate no se pueden enviar objetos como parámetros de la petición por eso hay métodos de la API de las que no se puede probar su funcionamiento porque utilizan objetos concretos.

  Este problema no se ha podido resolver aún. Se recomienda cambiar el código para que no utilice objetos concretos o utilizar otra clase de Spring que permita utilizar objetos en las operaciones con una API REST para las pruebas del sistema.

4.8.4. Visualizador de pdf falla cuando no hay pdf

Cuando se intentaba mostrar el problema el navegador cargaba el visualizador del pdf para mostrarlo aunque no existiera. Era una característica del navegador.

Para resolver el fallo del visualizador del pdf, como es una característica del navegador, se intento modificar la plantilla HTML para no mostrar el pdf si no existía pero no dio resultado. Se intento añadir un script en la plantilla para poder hacer lo mismo y tampoco dio resultado. Como no funcionaba ninguna de las opciones mas evidentes, era el comportamiento del navegador y no había nada relacionado con este tipo de problema en la web no se podía hacer nada.

El problema fue resuelto con la ayuda de Raúl el codirector del TFG, siguiendo sus indicaciones se ha modificado el controlador y la API de manera que cuando se realice la petición para obtener el pdf de un problema se devuelva un 404 NotFound si el pdf no existe.

4.8.5. Utilizar un software de pruebas como Jenkins

A la hora de integrar el proyecto con Sonarcloud hubo algunas complicaciones:

- rabbit no estaba en ejecución

  A la hora de ejecutar las pruebas del sistema es necesario tener el servicio de mensajería rabbit en ejecución para poder enviar las entregas. El build con estaba configurado para ejecutar rabbit antes de llegar hasta la fase de mvn verify.

Para resolver este problema se ha modificado el build para ejecutar rabbit antes de ejecutar mvn verify que ejecuta los tests por defecto.

- carga de contexto fallido

  Cuando se ejecutan las pruebas con SpringBootTest se realiza una carga de contexto (configuración el entorno de pruebas, se realiza automáticamente). A la hora de ejecutar build no se podía realizar correctamente la carga de contexto, la carga de contexto fallaba.

  Para resolver este problema se ha utilizado la opción webEnvironment de SpringBootTest para que configure el entorno de pruebas correctamente (para las aplicaciones web).

- connection refused

  En las pruebas del sistema se utiliza la API. Para las pruebas se utilizaba el puerto 8080 pero eso provocaba el fallo de connection refused en GitHub con Maven cuando se ejecutaba el build ya que a la hora de configurar el entorno de pruebas (carga de contexto, se hace automáticamente) no se utilizaba el puerto 8080.

  Para resolver este problema se ha utilizado el valor RandomPort para que ejecutara los tests en un puerto aleatorio.

- problemas con la BBDD

  A la hora de ejecutar los tests después de la cargar el contexto correctamente daba fallos con la base de datos, problemas con las operaciones de ddl. La aplicación inicialmente utilizaba la base de datos h2 en memoria y cuando se ejecutaba la aplicación se añadían los lenguajes y el concurso de prueba con el usuario por defecto pabloXd. Debido a esto daba fallo de ddl.

  Para resolver este problema se ha añadido la propiedad spring.jpa.hibernate.ddl-auto con el valor update para que no intente crear la base de datos desde cero y modificado la parte del código donde se añadían los lenguajes a la BBDD y se creaba el concurso de prueba para que solo se realizaran estas operaciones si no existían previamente en la BBDD con los métodos existsBy de los repositorios.

4.8.6. Completar Docker

A la hora de arrancar toda la aplicación en un contenedor hubo algunas complicaciones:

- problemas a la hora de empaquetar la aplicación

  Maven por defecto, cuando empaqueta un proyecto y crea el jar, ejecuta las pruebas. Debido a esto daba fallo a la hora de realizar la conexión con la base de datos.

  Este problema fue arreglado ejecutando el comando con la opción -DskipTests en el Dockerfile.

- problemas a la hora de hacer build en el CI

  A la hora de hacer build en GitHub no se podía conectar con la BBDD, el fallo estaba en uno de los parámetros de la conexión (la contraseña estaba mal escrita).

  El problema fue resuelto por Raúl, el codirector del TFG que fue el que descubrió el fallo.

- problemas de conexión a la hora ejecutar la aplicación

  A la hora de ejecutar la aplicación daba fallo de conexión con la base de datos. Esto era debido a que era necesario conservar el volumen de la base de datos. Como se conservaba el volumen se utilizaban los credenciales que había al principio (usuario sa sin credenciales) para configurar la BBDD en el contenedor.

  No se conocía la razón por la cual se provocaba este fallo. Después de días intentando arreglar el fallo y buscando información en la web se el motivo del fallo estaba explicado en una issue en GitHub. [30].

  El problema fue resuelto eliminando los volúmenes antiguos de la BBDD y ejecutando la aplicación de nuevo.

Capítulo 5

Conclusiones

En este apartado se explican las conclusiones y se comentan algunos de los trabajos futuros posibles.

5.1. Conclusión

Este trabajo de fin de grado consiste en añadir nuevas funcionalidades en el juez de código online el juez (the judge en inglés) también conocido como iudex y mejorarlo. Para cumplir el objetivo se han realizado tareas de desarrollo web y mantenimiento web utilizando los conocimiento de la Ingeniería de SW, conocimientos de programación en Java y las Estructuras de Datos, conocimientos de desarrollo web y pruebas entre otras cosas.

Durante todo el proceso de desarrollo del proyecto (aplicación) se han empleado herramientas y tecnologías actuales como Docker, contenedores, Git, GitHub, Sonarcloud, etc. El desarrollo se ha llevado a cabo empleando metodologías ágiles como el tablero Kanban. Se han realizado las tareas con implementación continua y pruebas continuas.

Con el empleo de las herramientas mencionadas en los apartados anteriores se ha podido cumplir todos los objetivos planteados, además de descubrir nuevas tareas y trabajos futuros relevantes que se continuaran en desarrollo como la gestión de la autenticación y el uso de *web sockets*. Se ha podido cumplir con los objetivos iniciales que eran añadir funcionalidades nuevas, básicamente añadir soporte para C, C++ y SQL y mejorar el código (se pueden observar las métricas de Sonarcloud). Se han mejorado los logs, se han añadido pruebas y se ha integrado la aplicación con Sonarcloud y las tareas descritas en los apartados

5.2. Trabajos futuros

El juez es una aplicación web de código libre que se puede emplear en distintos campos y puede ser utilizado por distintos tipos de usuarios.

La aplicación puede ser mejorada de diversas maneras algunas de las cuales son:

- utilizar Websockets o tecnología similar que permita recargar páginas automáticamente cuando hay cambios en la base de datos para mejorar la experiencia de usuario
- mejorar el código para mejorar la calidad y la mantenibilidad del sistema

  Hay varias secciones de código que no tienen normas o patrones concretos (por ejemplo en los servicios y los controladores no se actúa de la misma forma en acciones similares) y a medida que se vayan realizando cambios el mantenimiento del sistema sera mas costoso.

  Siguiendo la lógica del sistema los controladores no deberían de utilizar directamente las entidades, para eso están los servicios. En los controladores algunos datos se obtienen de los repositorios (base de datos) y otros de los servicios.
- añadir mas pruebas
- añadir distintas formas de puntuar las entregas

  Actualmente se puntúa como se hace normalmente en los concursos de programación, el ganador (el equipo con mas puntuación) es el que resuelve mas problemas mas rápidamente y con menos intentos fallidos. Para evaluar el trabajo realizado por alumnos que normalmente están empezando no sería una buena experiencia, puede causar perdida de interés y motivación hacia la asignatura.

  Sería mejor añadir otras formas de puntuar las entregas que evite estos problemas.
- añadir métricas a las soluciones para que los usuarios puedan ver como de bueno es su algoritmo en los problemas donde la eficiencia, el rendimiento, el tiempo de ejecución o alguna otra característica se considere mas importante
- añadir gestión de autenticación
- añadir gestión de roles

- añadir soporte para mas lenguajes
- separar la aplicación en dos partes modificando solo el *back-end* y conservando el resto

Bibliografía

[1] *repositorio de iudex*. URL: https://github.com/URJC-CP/iudex.

[2] *About ICPC*. URL: https://icpc.global/regionals/abouticpc.

[3] *Welcome to Hash Code 2021*. URL: https://codingcompetitions.withgoogle.com/hashcode/rulesandterms.

[4] *Competiciones*. URL: https://www.kaggle.com/docs/competitions.

[5] *Contests*. URL: https://ioinformatics.org/page/contests/10.

[6] Daniel Borowski. *The 10 Most Popular Coding Challenge Websites [Updated for 2021]*. URL: https://www.freecodecamp.org/news/the-10-most-popular-coding-challenge-websites-of-2016-fb8a5672d22f/.

[7] *HOW TO COMPETE IN SRMS*. URL: https://www.topcoder.com/thrive/articles/How%20To%20Compete%20in%20SRMs.

[8] *Sobre Coderbyte*. URL: https://coderbyte.com/about.

[9] *Dashboard*. URL: https://www.hackerrank.com/dashboard.

[10] *Codewars*. URL: https://docs.codewars.com/.

[11] *LeetCode*. URL: https://leetcode.com/.

[12] *About the SPOJ System*. URL: https://www.spoj.com/info/.

[13] *STEP UP YOUR CODING GAME*. URL: https://www.codingame.com/start.

[14] Ahmed Fessi. *How would you rank online judges and training sites of IOI? Which is better for what?* URL: https://qr.ae/pGPuiu.

[15] *USA Computing Olympiad*. URL: http://www.usaco.org/.

[16] *CROATIAN OPEN COMPETITION IN INFORMATICS*. URL: https://hsin.hr/coci/.

[17] *Frequently Asked Questions*. URL: https://codeforces.com/help#q1.

[18] *About*. URL: https://dmoj.ca/about/.

[19] *Status*. URL: https://dmoj.ca/status/.

[20] *repositorio de online judge*. URL: https://github.com/TheOnlineJudge/ojudge.

[21] *Familia de protocolos de internet*. URL: https://es.wikipedia.org/wiki/Familia_de_protocolos_de_internet.

[22] *TIOBE Index for July 2021*. URL: https://www.tiobe.com/tiobe-index/.

[23] howtodoinjava. *¿Qué es una API de REST?* URL: https://www.redhat.com/es/topics/api/what-is-a-rest-api.

[24] *Web-service*. URL: https://conectasoftware.com/glosario/web-service/.

[25] Uriel Ruelas. *¿Qué es la programación orientada a aspectos (POA)?* URL: https://codingornot.com/que-es-la-programacion-orientada-a-aspectos-aop.

[26] Postman. *The Collaboration Platform for API Development*. URL: https://codingornot.com/que-es-la-programacion-orientada-a-aspectos-aop.

[27] *User Guide*. URL: https://github.com/google/guava/wiki.

[28] Oracle. *System Properties Java*. URL: https://docs.oracle.com/javase/tutorial/essential/environment/sysprop.html/.

[29] baeldung. *The Guide to RestTemplate*. URL: https://www.baeldung.com/rest-template.

[30] yosifkit. *Cannot connect to mysql database: Access denied*. URL: https://github.com/docker-library/mysql/issues/51#issuecomment-76989402.