UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Generación de cuestionarios sobre algoritmos de ordenación

Presentado por Mario García Martínez
en Universidad de Burgos — 10 de junio de 2024
Tutor: Juan José Rodríguez Díez

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

Dr. Juan José Rodríguez Díez, profesor del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Mario García Martínez, con DNI 71314794M, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado "Generación de cuestionarios sobre algoritmos de ordenación".

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 10 de junio de 2024

Vº. Bº. del Tutor:

Dr. Juan José Rodríguez Díez

Resumen

El objetivo de este proyecto es la creación de una herramienta que permita la generación de cuestionarios aleatorios sobre algoritmos de ordenación. Desarrollados específicamente para la plataforma Moodle a partir de la generación de ficheros en formato XML que se crean por medio del lenguaje de programación Python.

La herramienta consta de una aplicación web que permite a los usuarios seleccionar el algoritmo de ordenación del cual se desea realizar un banco de preguntas. Posteriormente permite al usuario seleccionar los datos necesarios para generar un cuestionario adaptado a sus preferencias.

Tras la generación de los bancos de preguntas está disponible la descarga de los ficheros XML para su posterior implementación en la plataforma Moodle.

La aplicación web del proyecto es accesible desde: https://saqg-ubu-aeee3c86.koyeb.app

Descriptores

Aplicación web, generador de cuestionarios, algoritmos de ordenación, procesador de lenguaje, XML, Python.

Abstract

The aim of this project is the creation of a tool that allows the generation of randomised questionnaires based on sorting algorithms. Developed specifically for the Moodle platform by generating files in XML format that are created using the programming language known as Python.

The tool consist of a web application that allows users to select the sorting algorithm from which they wish to create a bank of questions. Subsequently offers the user the possibility of selecting the data necessary to generate a questionnaire adapted to their preferences.

Once the question banks have been generated, XML files can be downloaded for subsequent implementation on the Moodle platform.

The web application is accessible from:

https://saqg-ubu-aeee3c86.koyeb.app

Keywords

Web application, quiz generator, sorting algorithms, language processor, XML, Python.

Índice general

Índice general iii

Índice de figuras v

Índice de tablas vi

1. Introducción 1
- 1.1. Estructura de la memoria . . . 2
- 1.2. Recursos adjuntos . . . 2

2. Objetivos del proyecto 5
- 2.1. Requisitos técnicos . . . 5

3. Conceptos teóricos 7
- 3.1. Algoritmos de Ordenación . . . 7
- 3.2. Moodle . . . 16

4. Técnicas y herramientas 19
- 4.1. Python . . . 19
- 4.2. Flask . . . 20
- 4.3. HTML . . . 20
- 4.4. CSS . . . 20
- 4.5. XML . . . 21
- 4.6. JavaScript . . . 21
- 4.7. Docker . . . 21
- 4.8. Metodología SCRUM . . . 22
- 4.9. Microsoft Visual Studio Code . . . 22

4.10. GitHub 23
4.11. Koyeb 24
4.12. Microsoft Teams 24
4.13. LaTeX 25
4.14. BibTeX 25
4.15. Katalon Recorder 25
4.16. SonarQube 25

5. Aspectos relevantes del desarrollo del proyecto 27
5.1. Primeros pasos en el proyecto 27
5.2. Comprensión de los conceptos teóricos 28
5.3. Desarrollo de la lógica del proyecto 29
5.4. Desarrollo de la aplicación web 31
5.5. Despliegue 32

6. Trabajos relacionados 35
6.1. MQGenerator 35
6.2. PLQUIZ 35
6.3. TFGII-Quiz-Grafos 36

7. Conclusiones y Líneas de trabajo futuras 37
7.1. Conclusiones 37
7.2. Líneas de trabajo futuras 38

Bibliografía 41

Índice de figuras

3.1. Esquema del funcionamiento del algoritmo Mergesort. 9
3.2. Esquema del funcionamiento unidireccional del algoritmo Quicksort. 11
3.3. Esquema del funcionamiento bidireccional del algoritmo Quicksort. 12
3.4. Esquema de construcción de un montículo del algoritmo Heapsort. 14
3.5. Esquema de extracción del valor máximo del montículo del algoritmo Heapsort. 15
4.1. Proyecto desplegado en Microsoft Visual Studio Code 23
4.2. Repositorio del proyecto en GitHub 24

Índice de tablas

3.1. Algoritmos incluidos y su complejidad algorítmica 7

1. Introducción

La sociedad cada vez depende más de las nuevas tecnologías. Vemos como los tiempos avanzan. Vamos empleando nuevos recursos para todo tipo de cosas y campos sociales que tienen mucho que ver con el mundo de la informática. El siguiente proyecto está relacionado directamente con el mundo de la educación. Su objetivo principal es facilitar tanto a profesores como alumnos la generación de cuestionarios online.

La necesidad de este tipo de proyectos está directamente relacionada con el confinamiento que gran parte de la humanidad sufrió durante el año 2020. A final de curso, vimos como colegios y universidades tuvieron que adaptarse a estas nuevas circunstancias en un corto periodo de tiempo. Eso se sumó a los requerimientos de los propios estudiantes para no perder el curso debido a esta circunstancia excepcional.

Por todo lo expuesto, *Moodle* se ha convertido en una herramienta que favorece la enseñanza desde el ámbito online. Gracias a las distintas funcionalidades que ofrece, ha permitido tanto a profesores como alumnos continuar con su aprendizaje. Esta plataforma también tuvo su debut en gran medida en el grado de ingeniería informática y ahora es un factor imprescindible para todos sus miembros.

Una de las herramientas que nos aporta la plataforma es la realización de cuestionarios de forma telemática. Dicha herramienta facilitó a los docentes la creación de cuestionarios tanto de forma manual como de forma automática.

Es por esta última razón por la que se sustenta la necesidad de este proyecto. Este trabajo permite desarrollar cuestionarios sobre los algoritmos y su estructura, concretamente sobre los algoritmos de ordenación de vectores. La aplicación ofrece una serie de preguntas ya estructuradas que el usuario puede elegir para su cuestionario sobre tres distintos algoritmos de

ordenación muy conocidos en su campo: *Mergesort*, *Quicksort* y *Heapsort*. Estos cuestionarios generan aleatoriamente una serie de listas desordenadas para plantear los distintos problemas que deberán resolver los alumnos durante la realización del cuestionario. Se podrán encontrar de preguntas sobre las distintas fases que realiza un algoritmo hasta preguntas sobre el número de comparaciones que realiza. Todo ello con el fin de facilitar la comprensión de su funcionamiento a los alumnos.

1.1. Estructura de la memoria

La memoria se estructura de la siguiente forma:

- **Introducción.** Descripción y contexto del trabajo realizado junto a la estructura de los documentos presentados y los recursos que se adjuntan.
- **Objetivos del proyecto.** Objetivos que se desean cumplir con el desarrollo del proyecto.
- **Conceptos teóricos.** Conceptos en los que se basa el desarrollo del proyecto.
- **Técnicas y herramientas.** Elementos útiles que se han requerido para la realización del proyecto.
- **Aspectos relevantes del desarrollo del proyecto.** Descripción que cuenta el progreso llevado durante el proyecto.
- **Trabajos relacionados.** Proyectos similares presentado por otros alumnos.
- **Conclusiones y líneas de trabajo futuras.** Determinaciones que se han adoptado tras la realización del proyecto junto con las posibles mejores a implementar en un futuro.
- **Bibliografía.** Origen de los materiales requeridos en este documento.

1.2. Recursos adjuntos

Se adjuntan los siguientes recursos:

- Aplicación compilada y disponible para su ejecución en equipos con sistema operativos Windows.
- Código fuente de la aplicación.
- Documentacion (memoria y anexos) en formato PDF.
- fichero de texto TXT con los requerimientos.

2. Objetivos del proyecto

Este apartado concreta de forma concisa los objetivos que se desean cumplir por medio del desarrollo de este proyecto así como algunos aspectos requeridos para la correcta implementación del software:

1. Desarrollar una aplicación web que permita la generación de archivos en formato *XML Moodle*.
2. Generar preguntas en formato *XML Moodle* para la inclusión de cuestionarios sobre algoritmos de ordenación.
3. Aportar datos aleatorios para las diferentes preguntas que se desean elaborar e incluir durante la generación de los cuestionarios.
4. Implementar diferentes algoritmos de ordenación para procesar correctamente los datos aleatorios que se generen para las diferentes preguntas de los cuestionarios.

2.1. Requisitos técnicos

A continuación se muestran una serie de requisitos que han sido necesarios especificar para contribuir con el cumplimiento de los objetivos establecidos para este proyecto:

1. Se debería desplegar la aplicación web preferiblemente desde un equipo con sistema operativo *Windows*.
2. Se debe generar un archivo *Dockerfile* para el correcto despliegue del contenedor de la aplicación web en cualquier equipo.

3. Se debe incluir un framework de desarrollo, preferiblemente *Flask*.
4. Se debe realizar la lógica del proyecto preferiblemente en el lenguaje *Python*.
5. Se utilizará *GitHub Desktop* para la gestión del proyecto y las tareas a desarrollar.
6. Se facilitará una aplicación web sencilla para la creación de los diferentes bancos de preguntas.

3. Conceptos teóricos

En este apartado procederemos a concretar aquellos conceptos teóricos en los que se ha basado el desarrollo del proyecto de forma que se pueda comprender correctamente.

3.1. Algoritmos de Ordenación

Los algoritmos se tratan de un procedimiento computacional que permiten solucionar problemas o procesar diferentes datos para llevar a cabo diferentes actividades. Para ello se recibe una estado inicial o una serie de datos que deben ser procesados y tras seguir una serie de pasos establecidos por el algoritmo se obtiene un nuevo estado o nuevos datos que solucionen el problema planteado para el método [6].

En este caso, los algoritmos de ordenación reciben una lista o vector con una secuencia de números. Posteriormente, se reordena la secuencia devolviéndola de nuevo con valores correctos y según las pautas de ordenación que se le exijan al algoritmo.

Algoritmos	Peor caso	Mejor caso
Mergesort	$O(n \log n)$	$O(n \log n)$
Quicksort	$O(n \log n)$	$O(n^2)$
Heapsort	$O(n \log n)$	$O(n \log n)$

Tabla 3.1: Algoritmos incluidos y su complejidad algorítmica

Con el tiempo, los algoritmos para ordenar diferentes valores se han ido utilizando cada vez más en computación. Es por ello que han ido evolucionando con el fin de ordenar los vectores de forma mas rápida y eficiente pues son requeridos para múltiples usos en la computación actual [10].

Algoritmo de ordenación por mezcla (Mergesort)

El algoritmo de ordenación por mezcla o *Mergesort* se basa en una estrategia algorítmica conocida como 'divide y vencerás'. Como su nombre indica, consiste en dividir el vector de valores que se desea ordenar por la mitad y recursivamente ir dividiéndolo por la mitad hasta reducir los componentes de la lista a 1.

Posteriormente, será necesario ir juntando las lista poco a poco de manera que se compruebe si un valor de cada lista es mayor que el otro valor de cada lista. Si es mayor el mayor valor pasa en la posición posterior de la lista y si es menor el valor pasa a la posición anterior de la lista.

Este algoritmo es normalmente recursivo de ordenamiento externo. Su complejidad algorítmica se comprende como $O(n \log n)$ donde 'n' corresponde al tamaño de los datos de entrada, en nuestro caso la longitud del vector correspondiente, y donde '$\log n$' supondría el número de divisiones que se producirían. Esta notación de complejidad representa el peor de los casos donde se realizaría el máximo número de comparaciones del algoritmo y por lo tanto sería el caso menos eficiente de todos los posibles ya que esto depende de la estructura de los datos de entrada [13].

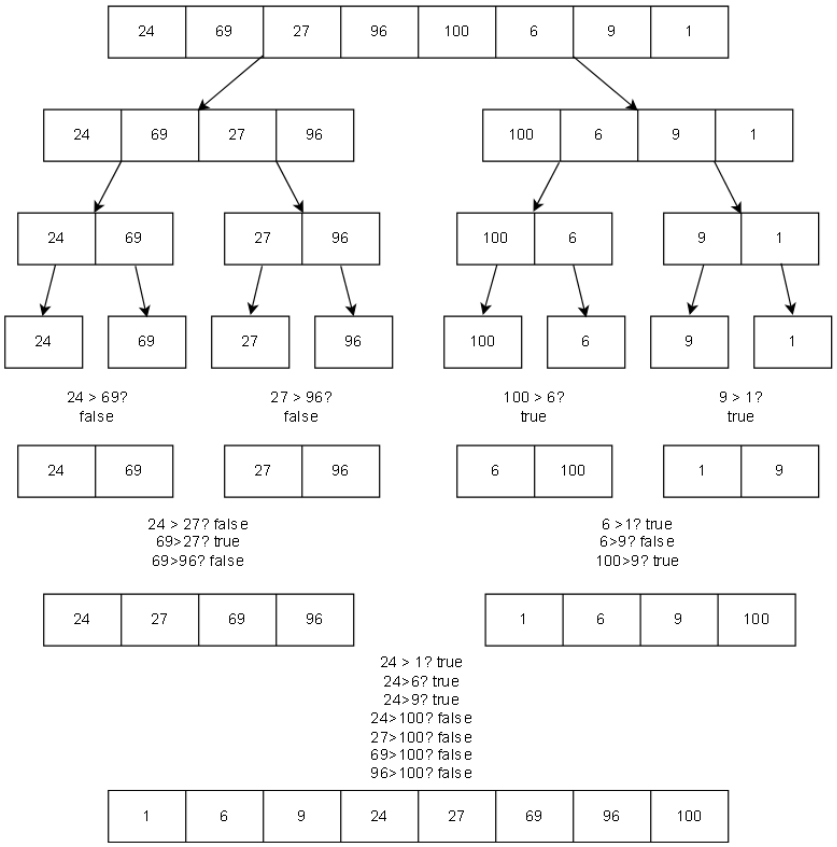

Figura 3.1: Esquema del funcionamiento del algoritmo Mergesort.

En esta imagen se puede observar un ejemplo de funcionamiento del algoritmo Mergesort. El vector se divide secuencialmente en dos mitades hasta reducirlo a vectores con un único valor solamente. Posteriormente se procede paso a paso a comparar las listas divididas para ver si es necesario reordenar.

El algoritmo compara siempre los primeros valores de las sublistas para ver si hay que reordenar. Si el valor del primer vector es mayor que el valor del segundo vector el valor del segundo se coloca el primero en la nueva lista y se pasa al segundo valor del segundo vector. Pero en el caso de que el valor del primer vector no sea mayor que el valor del segundo vector el valor del primero se coloca en la nueva lista y se pasa al siguiente valor del primer vector para volver a comparar.

Si se finaliza de comparar uno de los dos vectores antes de finalizar con el otro vector, se colocan los valores que queden del vector que todavía no ha finalizado puesto que ya son los mayores valores ordenados.

Algoritmo de ordenación rápida (Quicksort)

El algoritmo de ordenación rápida o *Quicksort* al igual que el Mergesort, también se basa en la estrategia de 'Divide y Vencerás'. Se irá realizando particiones más pequeñas y se irán reordenando de forma recursiva.

A diferencia del algoritmo anterior, el algoritmo Quicksort realiza la división de las particiones en base a uno de los valores del vector que es designado como el valor pivote. Durante el proceso de partición de el algoritmo Quicksort el objetivo será colocar todos aquellos valores mayores que el pivote a la derecha del mismo y todos los valores menores que el pivote a su izquierda.

En este caso, este algoritmo tiene dos métodos destacables para su reordenación. Existe un método unidireccional donde los dos índices a comprobar empiezan desde el extremo izquierdo del algoritmo y van avanzando hacia la derecha. El método bidireccional por otra parte, dispone de dos índices que comienzan cada uno en uno de los extremos y van avanzando en sentidos opuestos.

Este algoritmo es normalmente recursivo de ordenamiento externo. Su complejidad algorítmica se comprende como $O(n \log n)$ solo en el mejor de los casos donde el pivote queda exactamente en el medio de las dos sublistas, lo que hace que las dos sublistas sean del mismo tamaño.

Para el peor de los casos la complejidad algorítmica será de $O(n^2)$ que corresponderá a aquel caso en el que el pivote quede en un extremo de toda la lista pues se habrán comparado todos los valores [14].

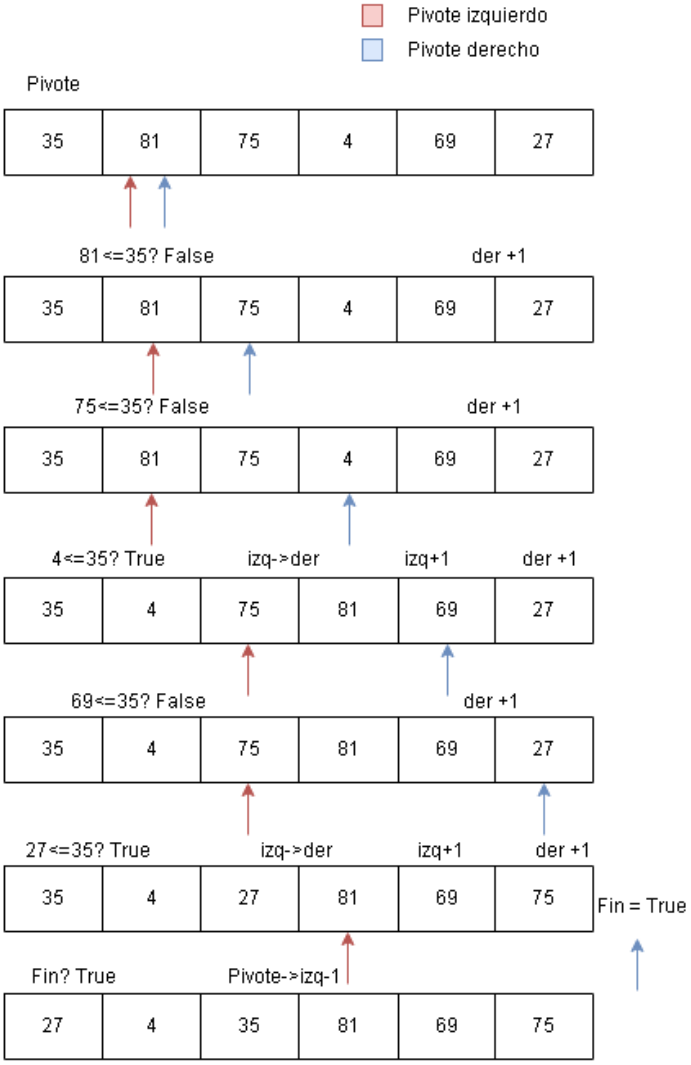

Figura 3.2: Esquema del funcionamiento unidireccional del algoritmo Quicksort.

Como se puede comprobar en esta imagen, en el método unidireccional del algoritmo *Quicksort* los dos índices comienzan en el extremo izquierdo. Se compara siempre el valor que marca el índice derecho con el valor del pivote. Si el valor es más pequeño que el pivote se intercambian los valores de las posiciones de los dos índices y se incrementan en 1 la posición de los dos índices.

En el caso en el que el valor del índice derecho no sea menor que el pivote únicamente se incrementará la posición del índice derecho. La partición del algoritmo finaliza cuando el índice derecho llega al final del vector y finalmente se intercambian las posiciones del pivote con la posición anterior a la del índice izquierdo.

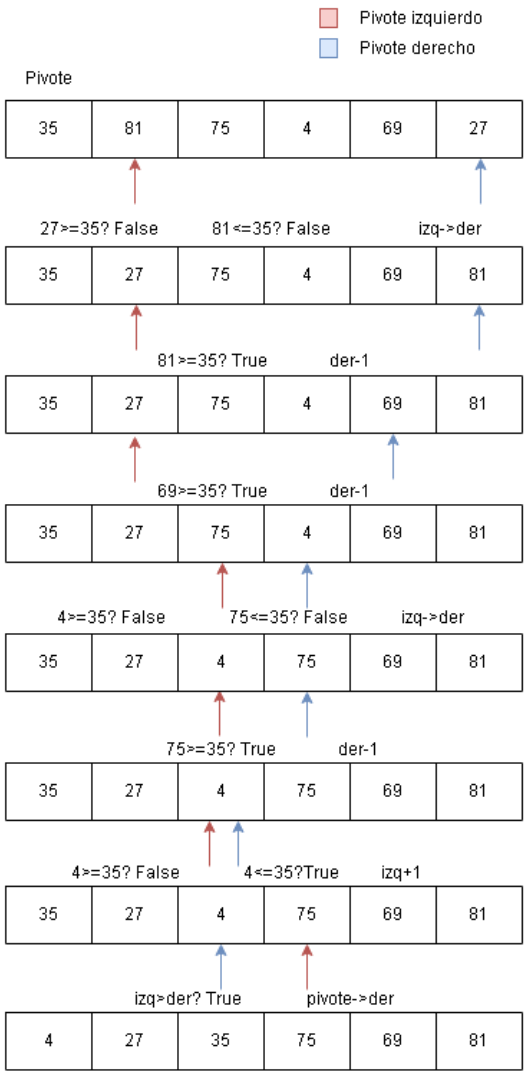

Figura 3.3: Esquema del funcionamiento bidireccional del algoritmo Quicksort.

En esta imagen se puede observar un esquema del método bidireccional del algoritmo *Quicksort*. Los dos índices comienzan cada uno en un extremo

y avanzan en sentidos contrarios. Se compara primero el valor que marca el índice derecho con el valor del pivote; si el valor es más pequeño que el pivote se comprueba el izquierdo y si este es mayor que el pivote se intercambian los valores de ambos índices.

En el caso en el que el valor del índice derecho no sea menor que el pivote se decrementa la posición del índice derecho. Y si el caso de que la condición anterior si se cumpla se comprobará si el índice izquierdo es mayor que el pivote, y si no es así se irá incrementando la posición del índice izquierdo hasta que se cumpla la condición o hasta que finalice el proceso de partición.

Para finalizar la partición, la posición del índice derecho tiene que encontrarse por debajo de la del izquierdo o que alguno de los índices haya llegado al otro extremo. En ese caso se intercambiará la posición del pivote con la del índice derecho.

Algoritmo de ordenación por montículos (Heapsort)

El algoritmo de ordenación por montículos o *Heapsort* a diferencia de los anteriores se trata únicamente de un algoritmo iterativo y no recursivo por lo que los elementos se reestructuran sin necesidad de volver a llamar al algoritmo. Su funcionamiento se basa en la construcción de montículos que se tratan de arboles binarios donde cada nodo tiene un valor mayor o igual que el valor de sus hijos. Por lo tanto el nodo de mayor valor debe ser el nodo raíz.

El método Heapsort tiene dos pasos claros a tener en cuenta: el primero es la construcción de el montículo de máximos donde el objetivo es reordenar el árbol binario que ha formado el vector de tal manera que los valores mayores estén en los nodos superiores. El segundo es la extracción de máximos del montículo para reordenar el vector es decir, intercambiamos el nodo raíz por la hoja del árbol binario que se encuentre más a la derecha y se extrae el valor de esa hoja. Por lo que se reduce el árbol y posteriormente se reordena de nuevo el montículo de máximos.

Este algoritmo tiene una complejidad de $O(n \log n)$ donde 'n' corresponde al tamaño de los datos de entrada y donde '$\log n$' corresponde con la altura del montículo, que es un árbol binario. Cabe destacar que este algoritmo no es muy útil para reordenar conjuntos de datos pequeños por la complejidad de su estructuración en montículo. Tampoco es muy eficiente para trabajar con datos muy complejos.

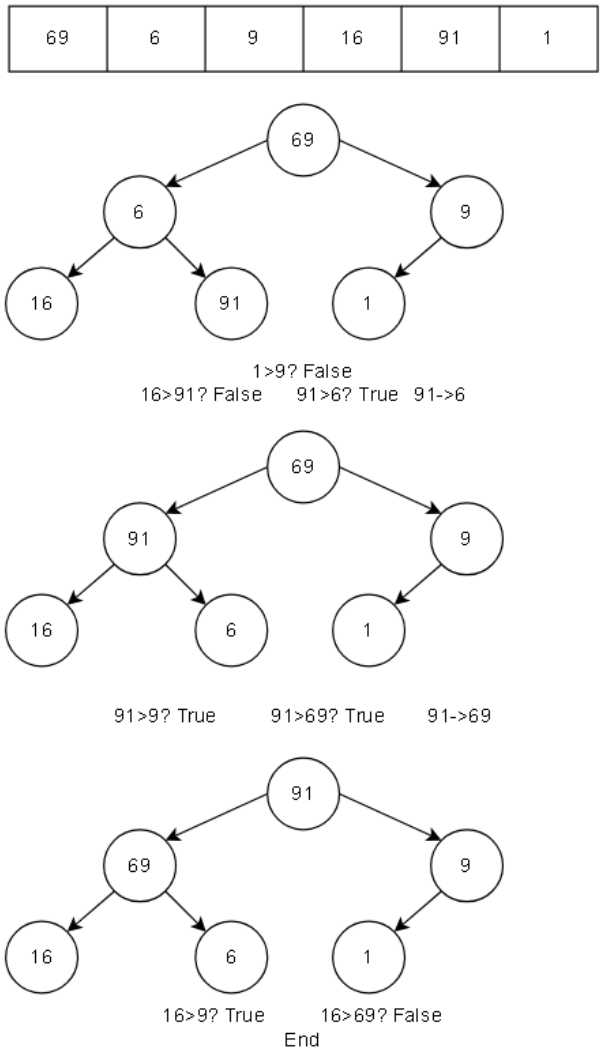

Figura 3.4: Esquema de construcción de un montículo del algoritmo Heapsort.

Como se puede observar en la imagen, para ordenar un montículo de máximos primero se debe comparar los hijos de los elementos mas inferiores del montículo para saber cual de los hijos es el más grande de los dos. Posteriormente se compara el hijo mas grande con su padre y si el hijo es mayor que el padre se intercambian las posiciones. Después se pasan a los hijos de la rama izquierda y se repite el método y posteriormente los valores superiores. En el caso en el que un hijo se intercambia por el padre, si el hijo tenía hijos a su vez se debe volver a comprobar si el nuevo padre es menor

que alguno de sus hijos para poder intercambiarlo. Al final se obtiene un montículo donde los valores mas grandes se encuentran en la parte superior y los más pequeños en la inferior.

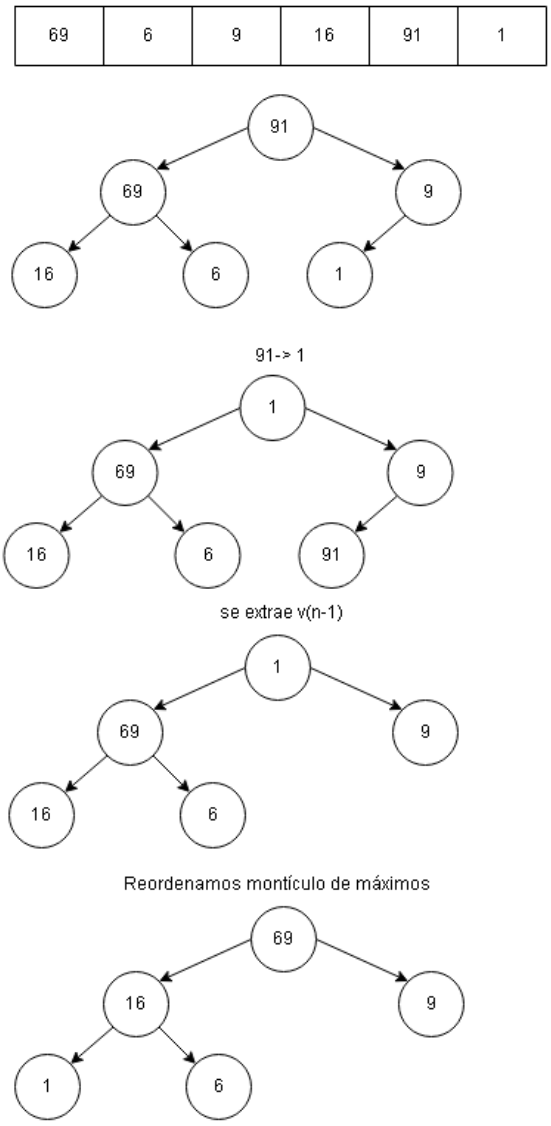

Figura 3.5: Esquema de extracción del valor máximo del montículo del algoritmo Heapsort.

Durante la extracción de los valores máximos se intercambia el valor de la raíz por el último valor en la lista. Luego se reordena el montículo

de máximos para continuar con la extracción de los mismos con el vector reducido a -1 en longitud puesto que el último valor ya ha sido colocado.

En Python este algoritmo está disponible por medio de la librería *heapq* la cual se ha tenido en cuenta para el desarrollo de algunas preguntas. Dicha librería aporta dos métodos interesantes de reordenación mediante el algoritmo *Heapsort*. El método *heappush* el cual consiste en ir formando un montículo de mínimos ya ordenado conforme se van a añadiendo valores al vector. Y el método *heapify* que permite reordenar el montículo cuando los valores estaban ya disponibles en el vector [12].

3.2. Moodle

Plataforma de aprendizaje desarrollada específicamente para educadores y estudiantes. Aporta un sistema personalizable de gestión de contenidos para el aprendizaje y el estudio.

Ofrece multitud de herramientas tanto para el profesorado. Algunas son la implementación de contenido para el estudio, la designación de tareas, el desarrollo de ejercicios que fomenten el trabajo de los alumnos; además de facilitar la corrección de los mismos y el cálculo de puntuaciones.

En cuanto al alumnado ofrece otro tipo de herramientas. Podemos disponer de un calendario donde pueden especificar sus labores de trabajo y especificar las fechas de finalización de los mismos. También disponen de acceso a foros para ayudarse entre los profesores y alumnos para comprender mejor algunos ámbitos de sus estudios. Y principalmente pueden realizar tanto exámenes como ejercicios y entregas de trabajos que ayuden con su comprensión del temario.

En nuestro caso debemos destacar la posibilidad de crear cuestionarios personalizados la cual no solo incluye la implementación manual sino también permite la inclusión de bancos de preguntas por medio de diferentes formatos. Cabe destacar el formato XML ya que se trata del formato en el que nuestro proyecto genera los bancos de preguntas de forma aleatoria.

En los bancos de preguntas nos podemos encontrar con una gran variedad de tipos de preguntas. Vamos a destacar principalmente las utilizadas durante el proyecto:

- **Preguntas tipo cloze:** Se tratan de preguntas donde se representa normalmente un texto en blanco que debe ser rellenado correctamente por el alumno con un valor, palabra o frase correcta.

- **Preguntas tipo multichoice:** Son aquellas conocidas como opción múltiple. Ofrecen varias soluciones posibles a la pregunta indicada y el alumno debe seleccionar la correcta.

4. Técnicas y herramientas

A continuación, se muestran las técnicas y herramientas utilizadas durante el desarrollo del proyecto, documentacion, ejecución y pruebas del mismo. También se incluyen otras herramientas que han sido necesarias durante el trabajo ajenas al desarrollo del software.

4.1. Python

Se trata de un lenguaje de alto nivel de programación cuya utilidad y versatilidad permite realizar aplicaciones de todo tipo de contexto. Es un lenguaje fácil de aprender y muy globalizado por una gran mayoría de programadores. No es necesario realizar una compilación previa y ofrece servicio multiplataforma para diferentes equipos.

Convenientemente, uno de los ámbitos en los que más se recurre a este lenguaje de programación es el desarrollo de aplicaciones y servicios web. Resultando bastante adecuado para el desarrollo del proyecto.

Se ha decidido usar este lenguaje para realizar la parte lógica de la aplicación web. No solo por sus características ya especificadas sino como desafío personal puesto que muchos otros trabajos del mismo ámbito se apoyaban en otros lenguajes de programación. Por otra parte es el lenguaje en el que más se ha estado trabajando durante el grado de ingeniería informática y con el que se siente mayor comodidad a la hora de trabajar.

4.2. Flask

Se trata de un microframework diseñado en Python y desarrollado principalmente para aplicaciones del mismo lenguaje. Permite la creación de aplicaciones web y utiliza el motor de plantillas de *Jinja2* que facilita la implementación de entornos HTML.

Es utilizado normalmente para aplicaciones que siguen el patrón de diseño *Modelo-Vista-Controlador* o MVC. Por tanto nos facilita su implementación y nos permite separar la lógica y el tratamiento de datos de la interfaz de usuario simplificando su complejidad cuando la aplicación sen encuentre en funcionamiento.

Elegido principalmente porque es el framework más adecuado para trabajos cuya lógica esté estructurada en el lenguaje de programación Python. Además permite desarrollar aplicaciones web rápidamente y de forma sencilla lo que ha facilitado bastante la construcción del proyecto.

Jinja2

Se trata de un motor de plantillas diseñado específicamente para software desarrollado en el lenguaje de programación Python. Utiliza un sistema de *templates* o plantillas donde se estructuran las diferentes páginas de la aplicación web en formato de texto.

4.3. HTML

HTML es lenguaje que define todo el contenido que se muestra en las aplicaciones web. Viene estructurado principalmente a partir de etiquetas donde se pueden definir elementos como imágenes, vídeos, enlaces a otras páginas, etc.

Se ha utilizado HTML principalmente para desarrollar las plantillas de las páginas web del proyecto. En ellas están se encuentra contenida la estructura de la web y los textos que se muestran en la misma.

4.4. CSS

CSS se corresponde con las siglas de "hojas de estilo en cascada". Se trata de un lenguaje de hojas de estilo que permite decorar los ficheros HTML para mejorar visualmente la interfaz de usuario [5].

Este lenguaje se ha utilizado principalmente para diseñar el estilo de las plantillas HTML, aportado estilos a las distintas etiquetas del contenido. Con esto se ha conseguido mostrar una interfaz de usuario adecuada con una estructura simple y ligeramente profesional.

4.5. XML

XML es conocido como un lenguaje de marcado extensible que permite intercambiar información entre los diferentes tipos de aplicaciones. Es un lenguaje que no puede operar con datos como lo harían otros, pero a cambio permite definir variedad de elementos para crear un nuevo formato adaptado a sus preferencias. Normalmente se utilizada para la administración de los datos para cualquier otro tipo de lenguaje de programación [3].

En nuestro caso, XML se corresponde con el formato en el que generamos los bancos de preguntas para que estas puedan visualizarse correctamente en la plataforma Moodle.

4.6. JavaScript

JavaScript es un lenguaje de comandos que permite implementar funciones a las aplicaciones web. Normalmente, las funcionalidades que incluye *JavaScript* son usadas para hacer que una aplicación web sea más dinámica y se adapte a los cambios que realizad el usuario cuando navega por la web.

Este lenguaje se ha utilizado principalmente para añadir funcionalidades a la interfaz como cierto control de errores dinámico para los formularios y una barra de navegación desplegable.

4.7. Docker

Docker es un sistema de contenedores que permite crear aplicaciones en módulos y ligeras. Estas aplicaciones se pueden desplegar fácilmente en otros contenedores virtuales ofreciendo una gran portabilidad para el software. Su funcionamiento consiste en generar una imagen o plantilla de *Docker* de nuestro software y posteriormente desplegar con esa imagen un contenedor en el equipo en el cual queramos ejecutar la app [7].

En nuestro caso se ha escogido *Docker* como herramienta de despliegue debido a la facilidad que ofrece en cuanto a su portabilidad. Gracias a esta

herramienta se puede desplegar nuestra aplicación web en cualquier equipo y servidor sin necesidad de pedir requerimientos a otros equipos.

4.8. Metodología SCRUM

Metodología basada principalmente en gestionar una serie de tareas periódicas fomentando el trabajo en grupo.

Facilita a los equipos de personas la estructuración y gestión del trabajo fomentando la experimentación y la capacidad personal para afrontar diferentes problemas. Es utilizada comúnmente por equipos encargados del diseño de software. Cada tarea realizada con esta metodología se relaciona con otras con el fin de coordinar el trabajo que realizan los distintos miembros del equipo [11].

Se ha seccionado esta metodología para gestionar el proyecto ya que es la que más se ha ido utilizando durante el grado. También porque ofrece un sistema de revisión parcial a medida que se desarrolla el software, lo cual resulta muy útil para el desarrollo de un trabajo de fin de grado.

4.9. Microsoft Visual Studio Code

Microsoft Visual Studio Code se trata de un editor de código multiplataforma. Permite la instalación de una gran variedad de extensiones para ayudar en la creación y desarrollo de nuevos proyecto. Además ofrece la posibilidad de trabajar con multitud de lenguajes de programación.

Utilizado principalmente debido a que al ofrecer un sistema multiplataforma ha facilitado el desarrollo del proyecto a la hora de trabajar con distintos lenguajes. Teniendo en cuenta que se han utilizado lenguajes como HTML, Python, JavaScript, XML y CSS. A su vez a facilitado el acceso a librerías y paquetes que han sido imprescindibles en el correcto funcionamiento del software.

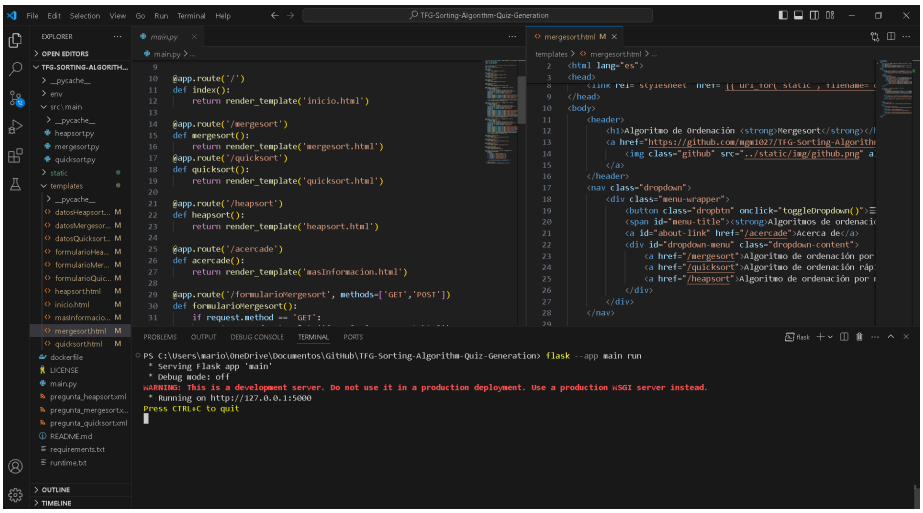

Figura 4.1: Proyecto desplegado en Microsoft Visual Studio Code

4.10. GitHub

GitHub es la plataforma más utilizada para el desarrollo de aplicaciones colaborativas y el control de nuevas versiones de proyectos actualmente. Utiliza el sistema conocido como *Git* para facilitar la creación y edición de repositorios donde los miembros del equipo pueden trabajar y mejorar el proyecto conjuntamente.

Se ha utilizado esta herramienta tanto para el control de versiones como para la implementación del repositorio puesto que ya se trabajó con esta y no resulta complicado utilizarla.

GitHub Desktop

Es una herramienta que permite llevar un control del repositorio trabajando con los archivos hospedados en *GitHub*. Permite insertar cambios y realizar una gran variedad de comandos de *Git* desde el escritorio.

Se ha utilizado esta herramienta principalmente para llevar un control del repositorio e ir actualizándolo conforme se iban desarrollando nuevas implementaciones para el mismo.

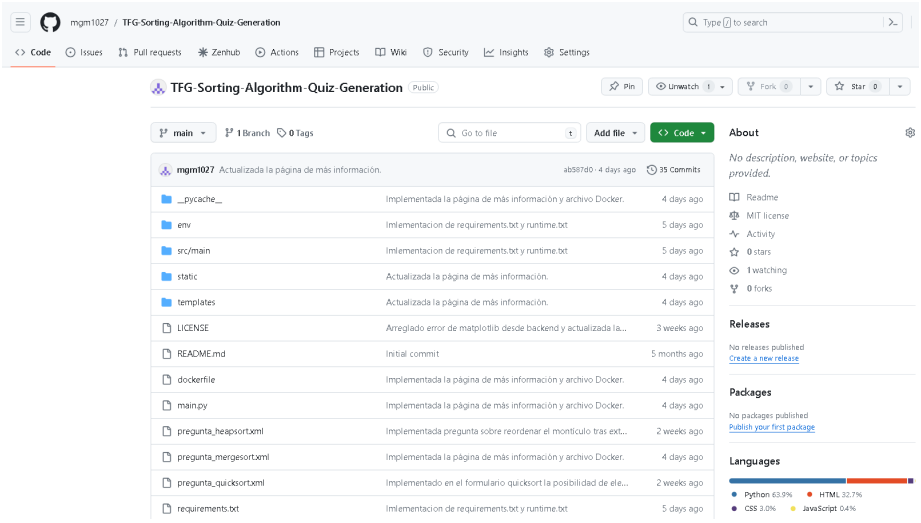

Figura 4.2: Repositorio del proyecto en GitHub

4.11. Koyeb

Koyeb es una plataforma gratuita que permite a los desarrolladores lanzar todo tipo de servicios y aplicaciones a la nube para su correcto despliegue. Esta plataforma facilita el lanzamiento de forma sencilla mediante contenedores *Docker* o desde repositorios de *GitHub*.

Se ha utilizado esta plataforma principalmente por el uso libre que permite a los desarrolladores desplegar aplicaciones sin coste alguno. También se ha elegido por la facilidad que ofrece al trabajar con contenedores de *Docker*, ya que se utilizan durante el despliegue de nuestro proyecto.

4.12. Microsoft Teams

Microsoft Teams se trata de una plataforma desarrollada con el objetivo de facilitar la colaboración entre distintas personas a través videollamadas, chats, envío de ficheros, etc. Su uso esta más extendido en el ámbito académico puesto que ha sido necesario para realizar conferencias y tutorías entre alumnos y profesores, y también para la realización de pruebas online.

Esta plataforma se ha utilizado principalmente para realizar las reuniones con el tutor del proyecto.

4.13. LaTeX

LaTeX es un sistema de composición de textos que permite la creación de documentos de texto. Se utiliza normalmente para la creación de artículos de ámbito académico y para la composición de tesis. También permite la creación de plantillas para adaptar la documentación a una serie de estándares.

En nuestro caso se utiliza para el desarrollo de la documentación, principalmente para poder seguir con las reglas de texto establecidas por la Universidad de Burgos.

4.14. BibTeX

BibTeX es una herramienta que facilita el manejo de referencias bibliográficas en *LaTeX*. Ayuda a gestionar las referencias para trabajos académicos como artículos científicos o libros.

Esta herramienta se utiliza principalmente para adaptar las diferentes referencias usadas durante el desarrollo de la documentación al formato de LaTeX ya que se trata del sistema editor de textos utilizado.

4.15. Katalon Recorder

Se trata de una extensión de navegador utilizada para la automatización de pruebas en aplicaciones web. Ofrece compatibilidad para realizar pruebas de *Selenium* y permite grabar las acciones para volver a aplicarlas de forma automática.

Se ha utilizado para realizar las pruebas automáticas sobre la entrada de datos desde la aplicación web.

4.16. SonarQube

Se trata de un software libre multiplataforma que permite analizar el código fuente en busca de fallos de diseño, vulnerabilidades, y código duplicado.

Se ha empleado principalmente para la limpieza de código y la refactorización de alguno de los métodos desarrollados

5. Aspectos relevantes del desarrollo del proyecto

En está sección se detallan los aspectos más destacables del desarrollo del proyecto, explicando todas la toma de decisiones que se ha ido llevando a cabo conforme se realizaba el trabajo. También se explican todos los problemas que han ido surgiendo a lo largo del proyecto.

5.1. Primeros pasos en el proyecto

Me decanté por realizar este proyecto principalmente porque era una de las pocas opciones que me interesaron de las que estaban disponibles para su elección, además de que no era la primera vez que trabaja en algo similar y me ofrecía la posibilidad de poner a prueba los conocimientos que he ido ganando a lo largo de mi paso por el grado. A su vez, el tema principal en el que se basa este proyecto me resultó interesante ya que trataba principalmente sobre el desarrollo de algoritmos con los que ya había trabajado anteriormente y que tienen multitud de usos en el ámbito profesional.

Por lo tanto, decidí reunirme con mi tutor para informarme más acerca de la propuesta y conocer más acerca de los objetivos que se querían completar. Tras finalizar la reunión me convenció la idea de enfocarme en este proyecto y decidí aceptar la propuesta de trabajo, puesto que se adaptaba a la idea de lo que yo buscaba y me resultaba muy interesante.

5.2. Comprensión de los conceptos teóricos

Conceptos básicos sobre los algoritmos de ordenación

Antes de comenzar con el desarrollo del proyecto, la primera semana me centré en revisar los conceptos sobre los algoritmos de ordenación, con el objetivo de poder estructurar correctamente el algoritmo y poder resolver por mi mismo alguna de las cuestiones iniciales que pudiera plantear. Para ello, fue necesario revisar previamente el funcionamiento del algoritmo *Mergesort* en un inicio y posteriormente el funcionamiento del algoritmo Quicksort. Por mi parte desconocía los dos métodos de funcionamiento del algoritmo *Quicksort*. Por lo tanto fue necesario realizar una prueba previa para comprender la diferencia entre ambos, ya que sin un ejemplo práctico me resultaba complicado entenderlos correctamente.

Comprensión de conceptos sobre Python

Con respecto al lenguaje de programación, no era la primera vez que trabaja con *Python*, pero al ser un lenguaje bastante amplio en cuanto a herramientas decidí revisar tanto los conceptos que ya había usado a lo largo de la carrera como algunos nuevos que pudieran resultar útiles para el proyecto. Comencé a revisar algunas de las librerías que ofrecían soporte para generar bancos de preguntas en *Moodle*, pero me centré principalmente en el formato XML puesto ya lo había utilizado en otra ocasión. Finalmente, opté por utilizar el módulo *ElementTree* que implementa una API simple para la creación de archivos XML y al probarla resulto ser bastante intuitiva.

Adaptación a la plataforma Moodle

Acerca de la plataforma *Moodle*, a pesar de haberla utilizado a lo largo de la carrera, era la primera vez que utilizaba su campo de pruebas. Además, para comprobar los bancos de preguntas era necesario entrar al campo de pruebas desde la perspectiva de un profesor, lo que me resultaba algo nuevo puesto que siempre había accedido como alumno. En un primer momento fue necesario aprender cómo crear un cuestionario y como importar un banco de preguntas en el formato XML, lo que resultó ser bastante intuitivo pero algo engorroso para tener que revisar cada banco de preguntas que se vaya a generar. Todo ello debido a que se requieren varios pasos solo para importar y posteriormente añadir las preguntas al cuestionario.

5.3. Desarrollo de la lógica del proyecto

Una vez revisados los conceptos básicos del proyecto, realicé un par de métodos para generar las primeras preguntas. Estas preguntas se basaban en los contenidos utilizados en algunos ejercicios que había realizado durante mi paso por la carrera.

Preguntas Iniciales

Inicialmente desarrollé una pregunta sobre la realización del algoritmo *Mergesort*, teniendo en cuenta que no se completaría el último proceso de mezcla. En esta pregunta el alumno tendría que rellenar una lista de espacios en blanco con el orden de la lista en el momento ya mencionado. No era muy complicado de plantear puesto que solo requería guardar el último cambio que sufría el vector antes de finalizar el algoritmo.

Posteriormente, se planteó lo mismo para el algoritmo *Quicksort* con la diferencia de que en este se revisaría el proceso tras dividir por primera vez el algoritmo. Además se tuvo que incluir los dos métodos de los algoritmos para que el alumno pudiera completar tanto el método unidireccional como bidireccional.

La implementación de los algoritmos era necesaria realizarla. Esto es debido a que se necesita un vector que estuviera correctamente ordenado para comparar los datos que los alumnos rellenen a la hora de corregir la pregunta.

Preguntas de selección múltiple

Tras realizar las primeras preguntas y revisar que el fichero XML se importara correctamente en *Moodle*, al hablar con mi tutor decidimos que teníamos que probar otro tipo de preguntas. Para ello, se optó por incluir el formato de preguntas de selección múltiple que admitía *Moodle*. Resultaba bastante complicado puesto que no bastaba con generar un vector aleatorio por cada pregunta creada. Para la creación de una pregunta de selección múltiple se tuvo que tener en cuenta estos puntos:

- No solo el vector de entrada tiene que ser aleatorio, también lo tiene que ser las respuestas que no sean correctas.
- La respuesta correcta no debería ser siempre la misma opción entre diferentes preguntas de selección múltiple.

- Las opciones que no son correctas deben ser al menos parecidas al vector de entrada, de manera que sea coherente como posible opción.

Teniendo todo esto en cuenta, no solo se tuvo que hacer aleatorio el vector de entrada, si no que tuve que ordenar de forma aleatoria las opciones que no eran correctas y también el orden de las opciones.

Implementación de un nuevo algoritmo

Una vez finalizadas las preguntas de selección múltiple, planteamos la posibilidad de incluir un nuevo algoritmo de ordenación para el proyecto. En este caso se optó por añadir al algoritmo de ordenación por montículos o *Heapsort*. En un primer momento, tuve que revisar tanto la documentación como el funcionamiento sobre el algoritmo *Heapsort* porque no recordaba haber trabajado con él. Me resultó bastante difícil ya no solo comprender su funcionamiento, sino también diseñar nuevas preguntas para el algoritmo *Heapsort*.

Inicialmente, revisé algunos ejemplos de implementación pero al final se optó por incluir la implementación de la librería *heapq* de *Python*. Revisando la documentación del repositorio de *Python* sobre *heapq* aprendí sobre dos métodos de uso para el algoritmo conocidos como *heapify* y *heappush*. El primero reordenaba el vector en un montículo de mínimos y el segundo iba reordenando el vector a un montículo de mínimos conforme se iban añadiendo valores al vector.

Finalmente, se rediseñó el método *Heapsort* dentro de nuestro proyecto para realizar montículos de máximos.Se plantearon nuevas preguntas sobre los dos métodos que destacan al algoritmo *Heapsort*: la ordenación por montículos de máximos y la extracción de máximos. Pero para ello vi necesario implementar preguntas más visuales.

Nuevo tipo de preguntas

Tras iniciar el desarrollo de las cuestiones del nuevo algoritmo, empezamos a plantear preguntas con enunciados algo mas diferentes. Por tanto comencé a revisar algunos ejemplos de cursos sobre algoritmos de ordenación para poder incluir nuevas preguntas en este proyecto [1].

En este caso, decidí incluir una pregunta sobre el número de comparaciones que se producen cuando se ejecuta el algoritmo *Mergesort* sobre un vector concreto. Gracias a la pregunta anterior, también se me ocurrió implementar

una pregunta sobre que posición adquiere el pivote del algoritmo *Quicksort* cuando se termina una partición sobre otro vector en concreto.

Finalmente, se planteó también la posibilidad de implementar preguntas con imágenes para que se pueda visualizar bien los valores de los vectores y resulte mas sencillo responder a las preguntas. Esto resultaba bastante complicado puesto que, a pesar de que no era la primera vez que generaba imágenes a partir de datos, no sabía muy bien cómo mostrar las imágenes en los cuestionarios sino disponía de ningún lugar donde guardarlas. Todo esto se debía a que posteriormente los usuarios solo se descargarían los ficheros ningún sitio donde guardar las imágenes.

Tras revisar algunas opciones, me decante por codificar la imagen a *Base64*. Este tipo de codificación permite a cualquier tipo de dato transformarse en una cadena de texto para cuando tienen que ser transmitidos por ficheros que sean únicamente de texto. Este caso se adaptaba a los ficheros en formato XML puesto que son solo ficheros de texto [4]. Gracias a este método se pudo incluir las imágenes en los ficheros XML sin necesidad de almacenarlas en cualquier otro formato.

5.4. Desarrollo de la aplicación web

Tras haber desarrollado parte de la lógica del proyecto, comencé a desarrollar un entorno web para disponer de una interfaz de usuario sencilla y comprensible. Por esta misma razón decidí basarme el patrón de diseño de fachada puesto que oculta el funcionamiento de la lógica al usuario, facilitando el manejo del software al mismo.

Al principio, tuve que crear un par de páginas HTML con algunas funcionalidades para recordar cómo se usaban y cuáles eran las etiquetas más comunes. Comencé por realizar una página de inicio para presentar el proyecto y otra página para generar un banco de preguntas de uno de los algoritmos. Más tarde, comprendí que era necesario disponer de un entorno y del framework para poder navegar entre páginas.

Creación del entorno con Flask

Tras implementar las plantillas, comencé a realizar pequeñas pruebas con el framework de *Flask* puesto que era el más recomendado para aplicaciones web desarrolladas en *Python*. Empecé por desplegar un entorno para poder probar *Flask* desde mi equipo y, más tarde, comencé a probar distintas

herramientas que ofrecía la plataforma. Principalmente diseñé un par de métodos para poder navegar entre las dos páginas que había creado.

Con las plantillas HTML hechas y los vínculos correspondientes de *Flask* para poder navegar entre páginas, empecé a desarrollar un método para poder descargar ficheros por medio del framework. Para ello, era necesario importar el paquete de los archivos de *Python* que había creado para generar preguntas de todos los algoritmos. Luego, sólo se necesitaba llamar al método correspondiente para generar preguntas y descargar en el equipo del usuario el archivo generado dándole un nombre.

Implementación de formularios

Tras implementar un funcionamiento básico de la aplicación y comprobar que funcionara, comencé a desarrollar otras plantillas HTML para los demás algoritmos. También, comencé a desarrollar un formulario en las páginas para recoger los datos y usarlos desde el framework de *Flask*. Todo ello para recoger la información necesaria del usuario, y con ello poder generar un cuestionario a su medida.

Diseño de estilos

Una vez establecida las estructura de plantillas HTML, opté por modificar el estilo de las páginas con ayuda de un archivo CSS. Decidí optar por realizar un diseño sencillo para no sobrecargar la interfaz y complicar su comprensión al usuario. También, decidí añadir algo de dinamismo a la página con ayuda de un fichero de *JavaScript*. Con esto, pude implementar una barra de navegación desplegable y realizar parte de control de fallos en los formularios.

5.5. Despliegue

Una vez desarrollada la aplicación, solo quedaba realizar el despliegue y comprobar que funcionara todo correctamente. Para ello, estuve revisando distintas plataformas que ofrecieran servidores gratuitos. La gran mayoría de ellos no ofrecían un despliegue libre de costes y otros no me convencían del todo ya que exigían muchos requisitos. Finalmente opté por desplegar la aplicación web desde *Koyeb*, que aunque no fuera libre de costes ofrecía un plan gratuito.

Koyeb ofrece dos formas para desplegar la app: por medio del repositorio de *GitHub*, y por medio de una imagen de *Docker*. En un inicio, opté por desplegarlo desde *Docker*, y por lo tanto cree un archivo *Dockerfile* para crear una imagen de *Docker*. Por algunas complicaciones con uno de los servidores gratuitos de *Koyeb*, decidí volver a realizar el despliegue desde el repositorio de *GitHub* desde otro de los servidores. Por suerte, esta opción también te permitía de desplegar la aplicación por medio de un archivo *Dockerfile*, resultando conveniente para el proyecto. Finalmente, la aplicación se pudo desplegar sin fallos desde *Koyeb*, y con ello se había conseguido realizar correctamente el despliegue.

6. Trabajos relacionados

En este apartado se concretará algún trabajo relacionado con este proyecto. En este caso, muchos de estos trabajos con temática similar no habían sido desarrollados en *Python*.

6.1. MQGenerator

Álvaro Hoyuelos Martín, graduado en Ingeniería Informática por la Universidad de Burgos en el año 2023, realizó el proyecto sobre 'Desarrollo de una aplicación web para generar cuestionarios automáticos para la plataforma *Moodle* sobre el seguimiento y control de proyectos'.

En este proyecto también se realizó el cambio al lenguaje de programación *Python* para la generación de archivos XML. Se desarrolló una aplicación web que permite generar cuestionarios sobre el seguimiento y control de proyectos, realizando tres tipos de preguntas. En la primera, a partir de un conjunto de datos sobre un plan de proyecto se debe analizar el valor planificado, el coste actual, el valor ganado, etc. En la segunda, a partir de una gráfica sobre el seguimiento del proyecto se debe analizar el coste, tiempo y la estimación del mismo. En la tercera, a partir de una gráfica sobre el seguimiento del proyecto se debe calcular la variación del coste y determinar como va el proyecto [8].

6.2. PLQUIZ

Roberto Izquierdo Amo, graduado en Ingeniería Informática por la Universidad de Burgos en el año 2014, realizó el proyecto sobre "Generación

de cuestionarios para la plataforma *Moodle* sobre algoritmos de análisis léxico".

Este proyecto permite generar bancos de preguntas por medio del lenguaje de programación *Java*. También permite imprimir los cuestionarios en formato XML, donde también se codifican las imágenes mostradas a través de *Base64* [9].

6.3. TFGII-Quiz-Grafos

Jorge Alonso Márquez, graduado en Ingeniería Informática por la Universidad de Burgos en el año 2015, realizó el proyecto sobre 'Generación de cuestionarios sobre los algoritmos de *Prim*, *Dijkstra*, recorrido en anchura y recorrido en profundidad, etc'.

Este proyecto permite generar bancos de preguntas por medio del lenguaje de programación *Java*. También se incluye la creación de grafos visuales de manera dinámica para las diferentes preguntas [2].

7. Conclusiones y Líneas de trabajo futuras

En este apartado contiene las conclusiones obtenidas tras el desarrollo del proyecto junto con algunas posibles líneas de trabajo futuras para continuar mejorando el proyecto.

7.1. Conclusiones

Personalmente, me he sentido bastante cómodo a medida que he ido desarrollando el proyecto. Era la primera vez que desarrollaba completamente una aplicación desde sus inicios hasta su completo despliegue. A nivel personal, este proyecto me ha dado la oportunidad de utilizar prácticamente todos los conocimientos que he ido adquiriendo a lo largo de mi paso por el grado de ingeniería informática. Gracias a esto, ha mejorado mi confianza y seguridad a la hora de programar de diferentes formas que nunca había utilizado antes.

Hablando de los conceptos teóricos planteados para este proyecto, era una de las cosas que más me llamó la atención puesto que me resultaba bastante curioso la multitud de formas con las que se puede ordenar una lista de elementos. Además, otra de las razones por las que me apasionaban los algoritmos de ordenación es la multitud de usos que tienen, tanto en el ámbito profesional como empresarial.

Respecto a las herramientas utilizadas, no era la primera vez que trabajaba con *Python* ni con muchas de las librerías utilizadas. Por esa razón, resultó ser bastante reconfortante poder aplicar mi experiencia con estos elementos, sobre todo con la generación de imágenes. En cuanto a *Flask*, fue

un reto aprender a manejarme con el framework, sobretodo para manejarme entre las plantillas HTML y los métodos para generar los cuestionarios. La razón se debía que no tenía mucha experiencia a la hora de utilizar los métodos GET y POST para manejar la información entre la aplicación web y la lógica del proyecto.

En cuanto a gestión del trabajo, al final no me resultó difícil seguir las metodología SCRUM para la realización de proyecto. Por supuesto, me hubiera gustado poder llevar un mejor control de las tareas que me iban surgiendo a medida que iba desarrollando el código. Principalmente, debido a que al comienzo del proyecto me costó bastante pensar acerca de como estructurar el desarrollo del proyecto.

Como conclusión final, este proyecto ha resultado ser una premisa ideal para los proyectos que me voy a encontrar en un futuro en el mundo laboral. También, me ha proporcionado más confianza para desenvolverme en los diferentes problemas que me vayan surgiendo de aquí a un futuro próximo, además de hacerme sentir bastante orgulloso por ser capaz de finalizar un proyecto de principio a fin. Espero que mi experiencia sirva de ayuda para los compañeros que me sucedan cuando tengan que desarrollar su propio proyecto.

7.2. Líneas de trabajo futuras

A lo largo del desarrollo y durante las reuniones con el tutor, hemos detectado una serie de posibles implementaciones que podrían mejorar la experiencia de uso de la aplicación y serían interesantes aplicarlas.

Internacionalización de la aplicación

Actualmente, la aplicación web sólo esta disponible en español. Resultaría interesante implementar al menos un nuevo lenguaje como el inglés para que la aplicación pueda ser utilizada por más usuarios y se adapte a sus condiciones.

Mejora de la interfaz para otros dispositivos

La interfaz actual mantiene un diseño simple para facilitar la experiencia de usuario. No obstante, estaría interesante implementar un diseño mas claro y elaborado que no sea muy sobrecargado y que siga resultando fácilmente accesible para el usuario.

Implementación de nuevos algoritmos de ordenación

En cuanto a la temática principal del proyecto, todavía se disponen de muchos algoritmos de ordenación que no han sido incluidos para este proyecto. Incluir alguno de los múltiples algoritmos mejoraría ampliamente la variedad que ofrecen los bancos de preguntas.

Ampliar la variedad de preguntas

Durante este proyecto se han utilizado únicamente solo dos tipos de preguntas que admite la plataforma *Moodle*. Incluir alguna otra opción para plantear nuevas preguntas también favorecerían la variedad de preguntas del proyecto.

Implementar nuevos formatos de ficheros

Moodle ofrece mas formatos para la implementación de bancos de preguntas en sus cuestionarios aparte de XML. Incluir la generación de cuestionarios en estos formatos ofrecería la posibilidad de elegir a los usuarios aquel que mejor se adapte a sus preferencias.

Bibliografía

[1] Course | edX — learning.edx.org. https://learning.edx.org/course/course-v1:GTx+CS1332xIII+1T2024/home. [Accedido 04-06-2024].

[2] Jorge Alonso Márquez. GitHub - jorgealonsomar/TFGII-Quiz-Grafos: Trabajo de Fin de Grado. Grafos. — github.com. https://github.com/jorgealonsomar/TFGII-Quiz-Grafos, 2015. [Accedido 05-06-2024].

[3] AWS Amazon. ¿Qué es XML? - Explicación del lenguaje de marcado extensible (XML) - AWS — aws.amazon.com. https://aws.amazon.com/es/what-is/xml/. [Accedido 04-06-2024].

[4] MDN contributors. Base64 codificando y decodificando - Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web | MDN — developer.mozilla.org. https://developer.mozilla.org/es/docs/Glossary/Base64. [Accedido 04-06-2024].

[5] MDN contributors. CSS básico - Aprende desarrollo web | MDN — developer.mozilla.org. https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/CSS_basics. [Accedido 04-06-2024].

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 3 edition, 2009.

[7] Docker. What is a Container? — docker.com. https://www.docker.com/resources/what-container/. [Accedido 04-06-2024].

[8] Alvaro Hoyuelos Martín. GitHub - xhm1001/MQGenerator: La herramienta tiene como objetivo la generación automática y personalizada de problemas de control integrado de proyectos, fundamentalmente mediante técnicas de valor ganado y programación. — github.com. https://github.com/xhm1001/MQGenerator/tree/main, 2023. [Accedido 05-06-2024].

[9] Roberto Izquierdo Amo. GitHub - RobertoIA/PLQuiz — github.com. https://github.com/RobertoIA/PLQuiz, 2014. [Accedido 05-06-2024].

[10] Donald E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley Professional, Reading, MA, 2 edition, 1998.

[11] Ikujiro Nonaka and Hirotaka Takeuchi. The New New Product Development Game — hbr.org. https://hbr.org/1986/01/the-new-new-product-development-game, 1986. [Accedido 04-06-2024].

[12] Kevin O'Connor, Tim Peters, and Raymond Hettinger. cpython/Lib/heapq.py at main · python/cpython — github.com. https://github.com/python/cpython/blob/main/Lib/heapq.py. [Accedido 03-06-2024].

[13] Wikipedia. Ordenamiento por mezcla - Wikipedia, la enciclopedia libre — es.wikipedia.org. https://es.wikipedia.org/wiki/Ordenamiento_por_mezcla. [Accedido 03-06-2024].

[14] Wikipedia. Quicksort - Wikipedia, la enciclopedia libre — es.wikipedia.org. https://es.wikipedia.org/wiki/Quicksort. [Accedido 03-06-2024].