

# Proyecto de Curso

Daniel Rosas, Sebastian Forero, David Hernández, Jose David Ontiveros

*Estructuras de datos*

*Pontificia Universidad Javeriana*

**Abstract**—Este documento describe el desarrollo de un sistema de apoyo interactivo para el juego Scrabble, destinado a asistir a los jugadores en la identificación de palabras con base en las letras disponibles y las condiciones del tablero. Se presenta un enfoque innovador que incluye la inicialización y manipulación de un diccionario de palabras, búsqueda optimizada de palabras con prefijos y sufijos, y un algoritmo de combinaciones de letras. El sistema propuesto no solo busca mejorar la experiencia de juego, sino también servir como herramienta pedagógica en el aprendizaje del lenguaje.

**Index Terms**—Scrabble Sistema de apoyo interactivo, Identificación de palabras, Diccionario de palabras, Búsqueda de palabras, Aprendizaje del lenguaje

## I. INTRODUCTION

Scrabble es un juego de palabras ampliamente reconocido y apreciado tanto por su naturaleza competitiva como educativa. En este juego, la habilidad para construir palabras y la estrategia en el uso del tablero son cruciales para el éxito. Sin embargo, jugadores de todos los niveles pueden enfrentarse al desafío de optimizar sus jugadas. Para abordar esto, hemos desarrollado un sistema de apoyo que utiliza algoritmos avanzados para sugerir las mejores posibles palabras a partir de las letras disponibles. Además, el sistema incluye funcionalidades para evaluar la puntuación de palabras potenciales y admite interacción a través de una consola de comandos, facilitando así una experiencia de usuario fluida y educativa. Este trabajo detalla el diseño, la implementación y las capacidades del sistema, y discute su potencial para transformar el juego de Scrabble en una herramienta educativa más poderosa.

## II. DISEÑO DEL SISTEMA

El diseño del sistema de apoyo interactivo para Scrabble se fundamenta en una arquitectura modular que comprende un procedimiento principal y múltiples operaciones auxiliares. Estas operaciones se implementan mediante comandos específicos que el usuario puede invocar a través de una interfaz de consola.

### A. Procedimiento Principal

El procedimiento principal actúa como el núcleo del sistema, gestionando la inicialización del diccionario, la recepción de las entradas del usuario y la coordinación de las operaciones auxiliares.

#### Entradas:

- Letras disponibles: El usuario introduce las siete letras disponibles en su atril.
- Estado del tablero: Se introducen las letras ya colocadas en el tablero, junto con su posición.

#### Salidas:

- Lista de palabras posibles: Basándose en las entradas, el sistema proporciona una lista de palabras válidas.
- Puntuaciones estimadas: Para cada palabra sugerida, se calcula y muestra una puntuación basada en la ubicación potencial en el tablero y los modificadores de puntuación aplicables.

#### Condiciones:

- Las letras proporcionadas deben conformar palabras válidas dentro del diccionario del juego.
- Las palabras sugeridas deben encajar en el espacio disponible en el tablero, respetando las palabras ya colocadas.

### B. Operaciones Auxiliares

Las operaciones auxiliares son comandos que el usuario puede ejecutar para realizar tareas específicas, como la manipulación de diccionarios y la evaluación de palabras.

1) *Comando: Inicializar Diccionario:* **Entrada:** Ruta al archivo del diccionario. **Salida:** Confirmación de que el diccionario ha sido cargado. **Condiciones:** El archivo del diccionario debe estar presente y ser legible.

2) *Comando: Búsqueda por Prefijo/Sufijo:* **Entrada:** Prefijo/Sufijo especificado por el usuario. **Salida:** Lista de palabras que comienzan/terminan con el prefijo/sufijo dado. **Condiciones:** El prefijo/sufijo debe existir en el diccionario.

3) *Comando: Calcular Puntuación:* **Entrada:** Palabra a evaluar y su posición potencial en el tablero. **Salida:** Puntuación calculada de la palabra basada en su ubicación y letras. **Condiciones:** La palabra debe ser válida y la posición debe estar libre en el tablero.

Cada uno de estos comandos se integra dentro del procedimiento principal para proporcionar una experiencia de usuario fluida y efectiva, asegurando que el sistema no solo asista en la formación de palabras, sino también en el aprendizaje y la estrategia del juego de Scrabble.

## III. IMPLEMENTACIÓN DE ÁRBOLES

Como parte del sistema de apoyo interactivo para el juego Scrabble, se ha implementado una estructura de datos de árbol para facilitar la búsqueda de palabras con base en prefijos y sufijos. Dicha implementación se realiza a través de dos tipos distintos de árboles: un árbol prefijo y un árbol postfijo.

### A. Arbol Prefijo

El árbol prefijo, también conocido como trie, permite la inserción y búsqueda de palabras a partir de sus prefijos. Cada nodo del árbol representa una letra y la ruta desde la raíz hasta un nodo específico conforma un prefijo de alguna palabra en el diccionario.

### B. Arbol Postfijo

El árbol postfijo se construye de manera similar al árbol prefijo, pero en este caso, las palabras se insertan en orden inverso, facilitando la búsqueda de palabras con base en los sufijos.

**Inicialización:** Para inicializar los árboles, se siguen los pasos detallados a continuación:

- **Árbol Prefijo:** Se carga el diccionario de palabras y se inserta cada palabra en el árbol prefijo.
- **Árbol Postfijo:** Se inserta cada palabra en orden inverso en el árbol postfijo.

Estos árboles se mantienen actualizados a medida que se añaden nuevas palabras al diccionario durante el juego.

### C. Operaciones con Arboles

- **Insertar Palabra:** Permite añadir una nueva palabra al árbol prefijo y al árbol postfijo.
- **Buscar Prefijo/Sufijo:** Realiza una búsqueda en el árbol correspondiente para encontrar todas las palabras que contengan el prefijo o sufijo indicado.
- **Generar Listado Postfijo/Prefijo:** Obtiene todas las palabras posibles a partir de un sufijo o prefijo dado, utilizando la estructura de datos del árbol correspondiente.

## IV. INICIALIZACIÓN DE ARBOLES EN EL SISTEMA

En el momento del lanzamiento del sistema, se realizan dos llamadas fundamentales para la inicialización de los árboles de datos:

- **Inicialización del Árbol Prefijo:** Se llama al método correspondiente para poblar el árbol con los prefijos de las palabras contenidas en el diccionario inicial.
- **Inicialización del Árbol Postfijo:** De manera análoga, se realiza la llamada al método que construirá el árbol de sufijos a partir de las palabras del diccionario.

Ambos árboles se encuentran disponibles inmediatamente después de su inicialización para realizar búsquedas y sugerir palabras durante el juego.

## V. TADS

Dentro del desarrollo del sistema de soporte para el juego Scrabble, se han definido varios Tipos Abstractos de Datos (TADs) para encapsular la lógica y los datos de las entidades principales del juego. A continuación, se detallan los TADs clave y su funcionalidad.

### A. TAD Diccionario

El TAD *Diccionario* representa la colección de todas las palabras válidas que pueden usarse en el juego.

#### Atributos:

- *palabras*: Una lista de objetos *Palabra* que contienen las palabras.

#### Métodos:

- *Diccionario()*: Constructor que inicializa un diccionario vacío.
- *getPalabras()*: Devuelve la lista de palabras del diccionario.
- *agregarPalabra(Palabra palabra)*: Agrega una nueva palabra al diccionario.
- *verificarPalabra(string palabra)*: Verifica si una palabra es válida.

### B. TAD Letra

El TAD *Letra* gestiona la información individual de cada letra utilizada en el juego, incluyendo su puntuación.

#### Atributos:

- *puntaje*: Puntuación asociada a la letra.
- *valor*: Carácter que representa la letra.

#### Métodos:

- *Letra(char v)*: Constructor que inicializa la letra con un valor y puntaje.
- *getPuntaje()*: Retorna el puntaje de la letra.
- *setPuntaje(int puntaje)*: Establece el puntaje de la letra.

### C. TAD ProcesadorComandos

El TAD *ProcesadorComandos* proporciona la lógica para interpretar y ejecutar los comandos ingresados por el usuario.

#### Métodos:

- *inicializarDiccionario(string rutaDiccionario)*: Carga palabras desde un archivo.
- *puntajePalabra(string palabra)*: Calcula el puntaje de una palabra dada.

### D. TAD Palabras

El TAD *Palabras* encapsula las operaciones relacionadas con la manipulación de palabras en el juego.

#### Atributos:

- *letras*: Lista de *Letra* que componen la palabra.
- *puntaje*: Puntuación total de la palabra.

#### Métodos:

- *Palabra(string cadena)*: Construye una palabra a partir de una cadena de caracteres.
- *getLetras()*: Retorna las letras que componen la palabra.

### E. TAD Consola

El TAD *Consola* maneja la interacción con el usuario a través de la línea de comandos.

#### Métodos:

- *recibirComando(char\* comando)*: Procesa el comando ingresado por el usuario.
- *ayuda()*: Muestra los comandos disponibles.

Diccionario
Atributos: palabras: Una lista de objetos Palabra que representan las palabras almacenadas en el diccionario.
Métodos: Diccionario(): Constructor por defecto que inicializa un diccionario vacío. Diccionario(std::list<Palabra> palabras): Constructor que inicializa el diccionario con una lista de palabras. getPalabras(): Método para obtener la lista de palabras del diccionario. setPalabras(std::list<Palabra> palabras): Método para establecer la lista de palabras del diccionario. agregarPalabra(Palabra palabra): Método para agregar una palabra al diccionario. verificarPalabra(std::string palabra): Método para verificar si una palabra contiene únicamente caracteres válidos.

Fig. 1. TAD Diccionario

Letra
Atributos: puntaje: Un entero que representa el puntaje asociado a la letra. valor: Un carácter que representa el valor de la letra.
Métodos: Letra(): Constructor por defecto que inicializa el puntaje en 0. Letra(char v): Constructor que inicializa el puntaje y el valor de la letra. getPuntaje(): Método para obtener el puntaje de la letra. setPuntaje(int puntaje): Método para establecer el puntaje de la letra. getValor(): Método para obtener el valor de la letra. setValor(char v): Método para establecer el valor de la letra. calcularPuntaje(char v): Método para calcular el puntaje de una letra específica.

Fig. 2. TAD Letra

ProcesadorComandos
Descripción: Proporciona métodos para procesar comandos relacionados con la manipulación del diccionario y operaciones de búsqueda y combinación de palabras.
Métodos: inicializarDiccionario(std::string rutaDiccionario): Inicializa el diccionario a partir de un archivo de texto. inicializarDiccionarioInverso(std::string rutaDiccionario): Inicializa el diccionario a partir de un archivo de texto y almacena las palabras en sentido inverso. puntajePalabra(std::string palabra): Calcula el puntaje de una palabra. verificarPalabra(std::string palabra): Verifica si una palabra es válida. calcularPuntaje(std::string palabra): Calcula el puntaje de una palabra. enMinuscula(std::string palabra): Convierte una palabra a minúsculas. Métodos relacionados con la búsqueda de palabras por prefijo, sufijo y combinaciones.

Fig. 3. TAD ProcesadorComandos

Palabras
Atributos: letras: Una lista de objetos Letra que representan las letras que componen la palabra. puntaje: Un entero que representa el puntaje total de la palabra.
Métodos: Palabra(): Constructor por defecto que inicializa una palabra vacía. Palabra(std::list<Letra> _letras): Constructor que inicializa la palabra con una lista de letras. Palabra(std::string cadena): Constructor que toma una cadena de caracteres y la convierte en una lista de letras para inicializar la palabra. setLetras(std::list<Letra> _letras): Método para establecer las letras de la palabra. getLetras(): Método para obtener las letras de la palabra. getPuntaje(): Método para obtener el puntaje de la palabra. setPuntaje(int puntaje): Método para establecer el puntaje de la palabra. palabraInversa(): Método para obtener la palabra invertida. calcularPuntaje(): Método para calcular el puntaje de la palabra. agregarLetra(Letra l): Método para agregar una letra a la palabra.

Fig. 4. TAD Palabra

Consola
Métodos: recibirComando(char* comando): Función que recibe un comando ingresado por el usuario desde la consola y lo procesa. ayuda(): Función que muestra la lista de comandos disponibles. ayudaComando(const char* comandoAyuda): Función que muestra la descripción de un comando específico.

Fig. 5. TAD Consola

## VI. PLAN DE PRUEBAS

La siguiente tabla presenta el plan de pruebas diseñado para evaluar el sistema. Cada operación se somete a una serie de casos de prueba, comparando los resultados obtenidos con los esperados y proporcionando un análisis en caso de discrepancias.

### A. Descripción general:

El comando "puntaje" calcula el puntaje de una palabra en el juego Scrabble. Este plan de pruebas evaluará el correcto funcionamiento del comando "puntaje" bajo varias condiciones, incluyendo palabras válidas e inválidas, y verificará si el diccionario se inicializa correctamente.

### B. Procedimiento de pruebas:

Se compilará y ejecutará el programa Scrabble ('main') en un entorno de desarrollo utilizando el compilador g++ con el estándar de C++11. Se ejecutarán pruebas individuales mediante la introducción de comandos "puntaje" con palabras específicas en el diccionario. Se verificarán los resultados obtenidos según las especificaciones del juego Scrabble y se compararán con los resultados esperados.

### C. Pruebas individuales:

- Prueba 1:** Inicialización del diccionario Comando: '\$inicializar diccionario.txt' Verificar si el diccionario se inicializa correctamente desde el archivo 'diccionario.txt'. Resultado esperado: Mensaje indicando que el diccionario se ha inicializado correctamente desde el archivo especificado.
- Prueba 2:** Palabra válida Comando: '\$puntaje zebra' Verificar el puntaje de la palabra "zebra". Resultado esperado: Un puntaje de 16, que es el puntaje válido según las reglas del juego Scrabble para la palabra "zebra".
- Prueba 3:** Palabra válida Comando: '\$puntaje quilt' Verificar el puntaje de la palabra "quilt". Resultado esperado: Un puntaje de 14, que es el puntaje válido según las reglas del juego Scrabble para la palabra "quilt".
- Prueba 4:** Palabra no existente en el diccionario Comando: '\$puntaje palabra' Verificar el manejo adecuado de una palabra que no existe en el diccionario. Resultado esperado: Un mensaje indicando que la palabra no existe en el diccionario.
- Prueba 5:** Palabra con caracteres inválidos Comando: '\$puntaje p;lop' Verificar el manejo adecuado de una palabra que contiene caracteres inválidos. Resultado esperado: Un mensaje indicando que la palabra contiene símbolos inválidos.
- Prueba 6:** Palabra no existente en el diccionario Comando: '\$puntaje spider' Verificar el manejo adecuado de una palabra que no existe en el diccionario. Resultado esperado: Un mensaje indicando que la palabra no existe en el diccionario.

## VII. RESULTADOS OBTENIDOS

Este plan de pruebas establece un marco detallado para evaluar la funcionalidad del comando "puntaje" en el juego Scrabble, asegurando que el programa maneje correctamente una variedad de casos de uso y proporcione resultados precisos según las reglas del juego.

```
PS C:\Users\david\OneDrive\Documents\ProyectoScrabble\Scrabble\src> g++ -std=c++11 *.cpp -o main
PS C:\Users\david\OneDrive\Documents\ProyectoScrabble\Scrabble\src> ./main
$inicializar diccionario.txt
Diccionario inicializado correctamente desde
$puntaje zebra
La palabra tiene un puntaje de 16
$puntaje quilt
La palabra tiene un puntaje de 14
$puntaje palabra
La palabra no existe en el diccionario
$puntaje p;lop
La palabra contiene símbolos inválidos.
$puntaje spider
La palabra no existe en el diccionario
$
```

## VIII. ENTORNO DE PRUEBA:

Sistema operativo: Windows PowerShell Compilador: g++  
Archivos: 'main.cpp', 'diccionario.txt' (contiene las palabras válidas para el juego Scrabble) Comandos ejecutados directamente desde el entorno de Windows PowerShell.

## REFERENCES

- [1] J. Autor, *Título del libro*, 1ra ed. Ciudad: Editorial, Año.
- [2] A. Autor y B. Coautor, "Título del artículo," *Nombre de la Revista*, vol. xx, no. xx, pp. xx-xx, mes, año.
- [3] C. Autor, "Título del artículo de conferencia," en *Nombre de la Conferencia*, Ciudad, País, año, pp. xx-xx.