Bases and Bricks

Equipo de trabajo 7

José Luis Ávila Guzmán

Problema a resolver

El usar las piezas LEGO es una actividad muy divertida, pero dadas la cantidad de piezas existentes, es sencillo perder la cuenta de cuántas o cuáles se tienen para la realización de un proyecto o un set en específico. Haciendo uso de bases y estructuras de datos, se pretende crear un inventario para apoyar la administración personal de estas piezas.

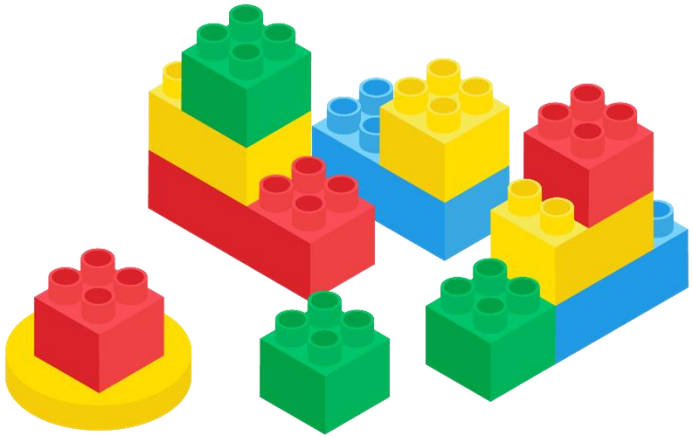


Imagen tomada de pngtree.com

Imagen tomada de pngimage.com

Requerimientos funcionales

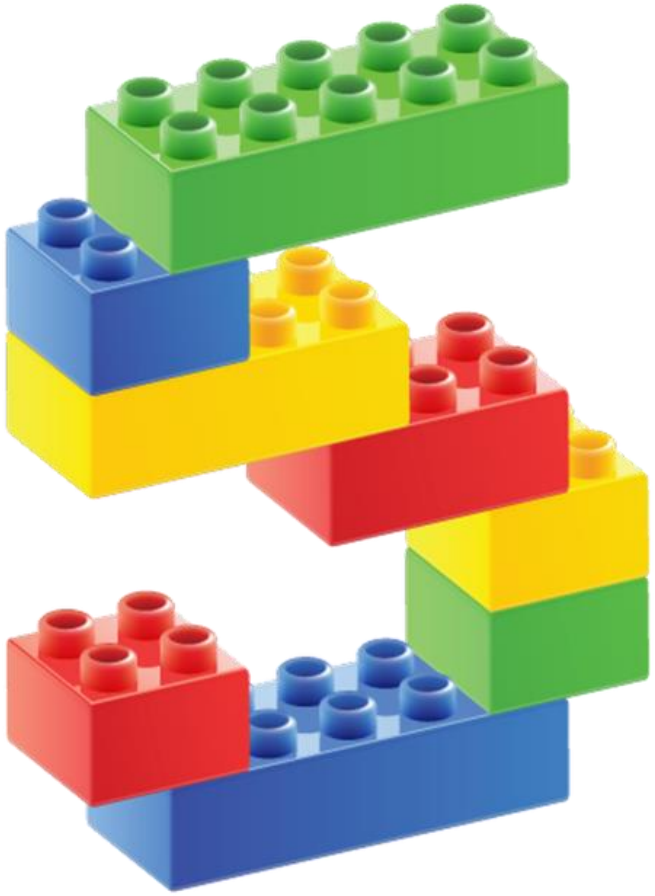


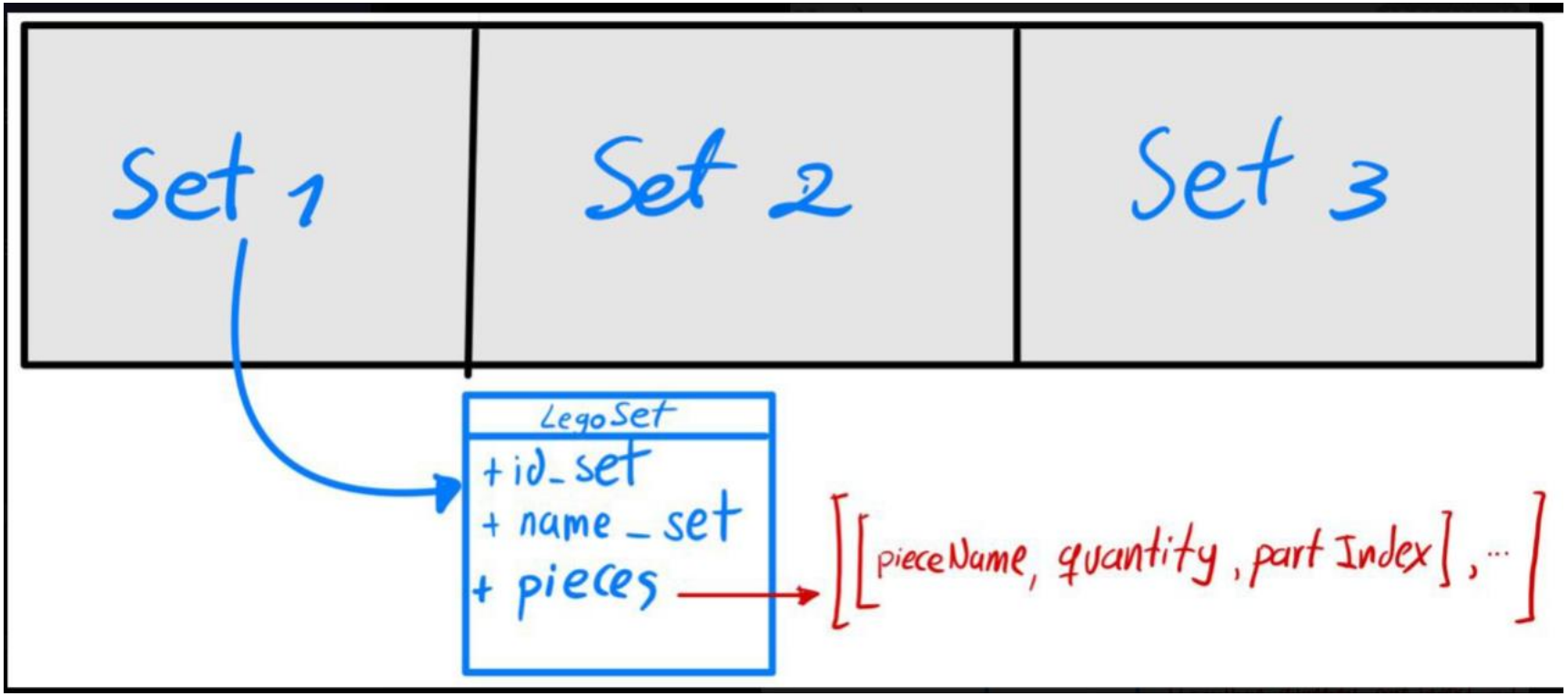
Imagen tomada de pngimage.com

El software requerirá de las siguientes funciones:

- Ingreso de datos
- Eliminación de piezas
- Visualización de inventario (Base de datos personal)
- Búsqueda en el inventario (Base de datos personal)
- Lista de deseos (Construcciones por hacer)

Uso de estructuras de datos

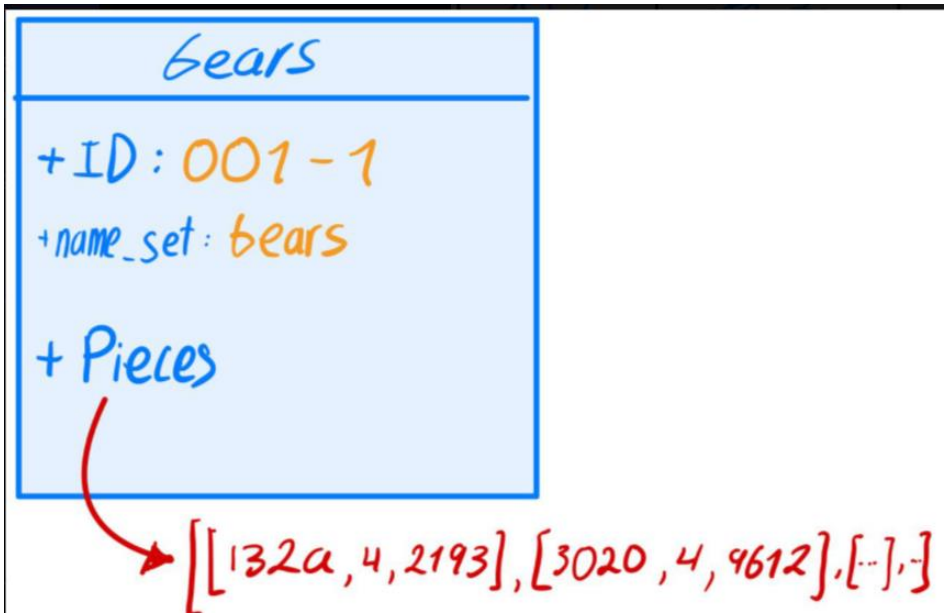
Lista simplemente encadenada y arreglo dinámico



Uso de estructuras de datos

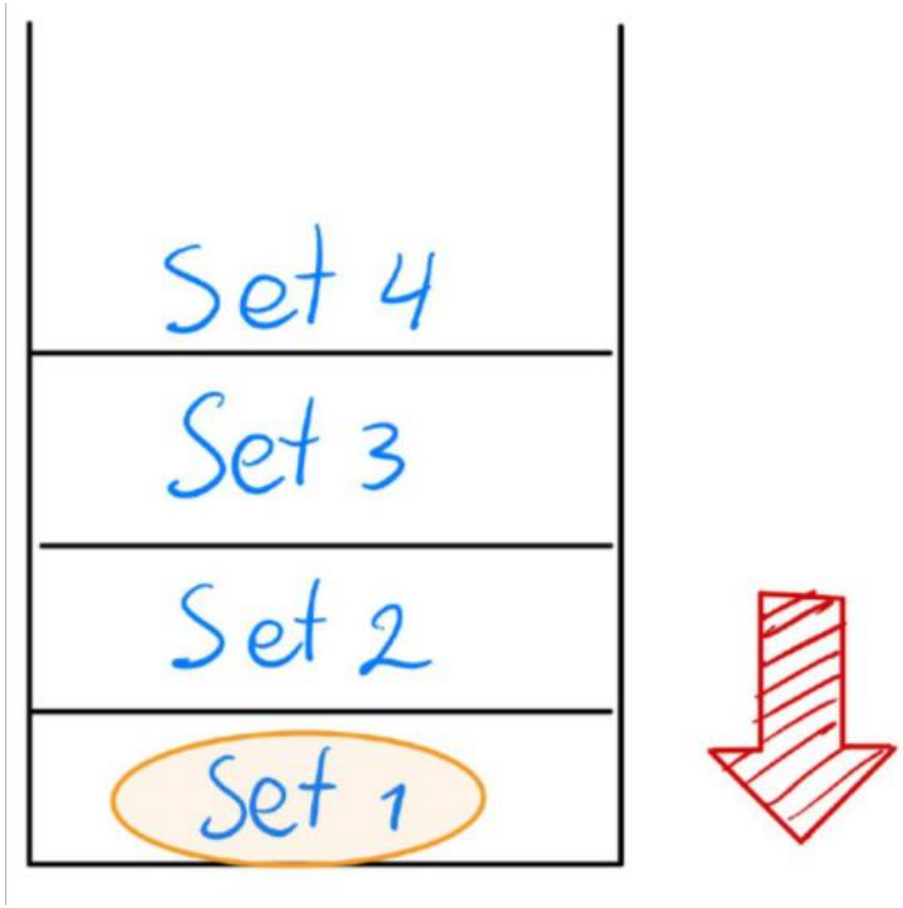
Stack

Set ID	Set Name	Piece ID	Quantity	Piece index
001-1	Gears	132a	4	2193
001-1	Gears	3020	4	9612
001-1	Gears	3062c	1	10504
001-1	Gears	3404bc01	4	13538
001-1	Gears	36	4	14068
001-1	Gears	7039	6	27684
001-1	Gears	7049b	4	27695
001-1	Gears	715	4	27801
001-1	Gears	741	4	28188
001-1	Gears	742	4	28198
001-1	Gears	743	1	28205
001-1	Gears	744	3	28206



Uso de estructuras de datos

Queue



Pruebas y análisis comparativo del uso de las estructuras de datos

Cargar la base de datos O(n)

```
1  public static void pushLegoSet(ref string[] line, ref LinkedList<LegoSet> list, ref Stack<LegoSet> objectStack)
2  {
3      LegoSet newLegoSet = new LegoSet();
4      newLegoSet.id_set = line[0];
5      newLegoSet.nameSet = line[1];
6      newLegoSet.addPiece(line[2], line[3], line[4]);
7      objectStack.Push(newLegoSet);
8      list.pushBack(newLegoSet);
9  }
10
11
12 string path = "C:/Users/josel/Documents/Estructuras/Database/Filtered queries/set_parts.csv";
13 string[] lines = System.IO.File.ReadAllLines(path);
14
15
16 // Load Sets_Parts
17 foreach (string line in lines) n veces
18 {
19     string[] columns = line.Split(','); cte
20     if (myStack.Count == 0) // Stack is empty
21         pushLegoSet(ref columns, ref setsLinkedList, ref myStack); cte
22     else if (myStack.Peek().id_set != columns[0]) // Different set
23         pushLegoSet(ref columns, ref setsLinkedList, ref myStack); cte
24     else
25         myStack.Peek().addPiece(columns[2], columns[3], columns[4]); cte
26 }
27
```

Pruebas y análisis comparativo del uso de las estructuras de datos

Eliminar una Pieza dado su ID O(n)

```
1  public static void removePieceByID(ref DynamicArray<Piece> legoPiecesList, string givenID)
2  {
3      int removeIndex = legoPiecesList.arraySize - 1;
4      for (int i = 0; i < legoPiecesList.arraySize; i++) n - removeIndex
5          if (legoPiecesList.data[i].lego_id == givenID)
6              removeIndex = i;
7      for (int i = removeIndex; i < legoPiecesList.arraySize - 1; i++) remove Index
8          legoPiecesList.data[i] = legoPiecesList.data[i + 1];
9  }
```

n

Pruebas y análisis comparativo del uso de las estructuras de datos

PopBack lista simplemente encadenada O(n)

```
1   public T popBack()
2   {
3       try
4       {
5           if (head.next == null)
6           {
7               head = null;
8               tail = null;
9           }
10          Node<T> iteratorPointer = head;
11          while (iteratorPointer.next.next != null)
12              iteratorPointer = iteratorPointer.next;
13          iteratorPointer.next = null;
14          T returnValue = tail.key;
15          tail = iteratorPointer;
16          return returnValue;
17      }
18      catch (Exception e)
19      { throw new Exception("List is empty"); }
20  }
```

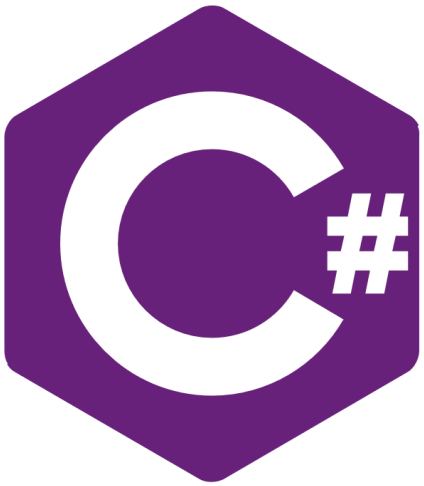
n

Pruebas y análisis comparativo del uso de las estructuras de datos

	Load Sets data	Load Parts Data	Remove piece by id	Pop Back sets linked list
10000	0.009942	0.0118441	0.000596	0.000035
100000	0.134635	0.1568757	0.005274	0.0001387
1000000	1.28967	1.2076322	0.099151	0.0005387
10000000	13.72169	No data	No data	0.0042013
100000000	No data	No data	No data	No data

Lenguajes de programación y herramientas de software usados

El lenguaje de programación usado es C#, y para su usarlo se aprovecha la herramienta Visual Studio con las extensiones y paquetes correspondientes. Además se usa MySQL para organizar los datos de la base de datos gratuita de lego Rebrickable.



Referencias

[1] "Rebrickable Help Guide: Frequently Asked Questions | Rebrickable - Build with LEGO", *Rebrickable.com*. [Online]. Available: https://rebrickable.com/help/faq/. [Accessed: 28- Nov- 2021].

[2] N. Rhodes, "Basic Data Structures: Dynamic Arrays and Amortized Analysis", *Coursera*. [Online]. Available: https://d3c33hcgiwev3.cloudfront.net/_72d29db2f2280185e66f0a77ada6d61f_05_4_dynamic_arrays_and_amortized_analysis.pdf?Expires=1638230400&Signature=OCtloBnmfH2~jBlSkkE~VeYAPj8dnEOxRTZvOffbieC8sohBruCdtWozEIXyR4~8VyomwX8Wq~iTtF2-OeKF2TcnlXpro6SM4kZCA8GGbuYcFGv66gR3bKfgnKw4X5SBK13L444ZKRK~2dT6QwvuIe90ht~BbRwrHYBnqaoz7sM_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A. [Accessed: 28- Nov- 2021].