**Bases and Bricks**

**José Luis Ávila Guzman**

[Sebastian Felipe Ruiz Espinosa](mailto:seruize@unal.edu.co)

**No. de equipo de trabajo: 7**

# introducción

En el mundo de LEGO, a veces se nos complica más encontrar la pieza indicada en el manual, que la misma construcción. Esto puede generar frustración al momento de construir o crear un proyecto, puesto es más el tiempo que se toma buscar una pieza, que ensamblar y terminar la construcción. Por lo tanto, si se tienen todas las piezas debidamente clasificadas por atributos y ubicadas en un almacenamiento organizado, la experiencia con LEGO se va a disfrutar mucho más. Mediante una base de datos en una aplicación web llamada “Bases and Bricks”, se pretende mejorar la experiencia al momento de clasificar y organizar las fichas de LEGO que se posean, disponiendo también de un buscador, el cual encontrará fácilmente la pieza que se requiera, mediante la introducción de algunos atributos de la ficha.

# descripción del problema a resolver

Cuando se tiene una colección de LEGO, el principal problema es encontrar las piezas adecuadas para la construcción de algún set predeterminado o un proyecto que se tenga en mente. Esto se debe a que la marca dispone de aproximadamente 43000 piezas diferentes, disponibles hasta en 58 colores distintos (en esta primera entrega solo se tendrán en cuenta 11 colores diferentes). Debido a esta gran variedad de piezas, si no se tienen correctamente clasificadas y ubicadas, al momento de construir cualquier proyecto que se tenga en mente, va a ser muy difícil encontrar las piezas que se deseen usar.

# usuarios del producto de software

Este proyecto impacta principalmente a los amantes de los legos, puesto que como se ha mencionado anteriormente, los ayuda para organizar sus inventarios. Pero también puede dársele uso en las instituciones educativas, debido a que los lego son una excelente herramienta de aprendizaje para todas las edades, ya que con estas piezas se pueden lograr proyectos de construcción muy interesantes para una pedagogía más didáctica y atrayente. Otros usuarios de este producto son los constructores profesionales de LEGO, ya que manejan una gran cantidad de fichas para sus construcciones, por lo tanto tienen un inventario muy grande y diverso de piezas, y gracias a esta herramienta podrán organizar mas fácilmente las fichas a su disposición.

# requerimientos funcionales del software

Ingreso de datos:

* Descripción:

Mediante esta funcionalidad se van a ingresar nuevas piezas a la lista encadenada que almacena las piezas del usuario.

* Acciones iniciadoras y comportamiento esperado:

Al momento de ingresar en la página, el usuario dispondrá de una base de datos vacía (la cual en esta entrega está representada como una lista encadenada de objetos Pieza) previamente creada, para acceder a esta se debe de dar click en el recuadro donde indique "Mis piezas" en el menú de la página, esto lo redireccionará a la base de datos, la cual va a tener un recuadro llamado "Agregar", el cual le va a dar la opción de elegir que ficha quiere agregar a su base de datos y en qué cantidad. Una vez seleccione "Agregar", la pieza y la cantidad indicada por el usuario, se agregará a su base de datos personal.

* Requerimientos funcionales:

Esta funcionalidad necesita de una serie de botones que pongan en marcha las instrucciones indicadas anteriormente, los cuales van a acceder y modificar la base de datos del usuario. El primer botón que se va a encontrar es el "Agregar" en la extensión de la página "Mis piezas", al darle click redireccionará al usuario a una página en donde se encuentre la base de datos del programa, la cual contiene todas la piezas y sets disponibles de la marca LEGO. Una vez allí podrá buscar mediante una serie de filtros la pieza que desea agregar a la base de datos personal. Una vez la encuentre, le dará click a la pieza, lo cual llevará al usuario a una página con información adicional de la pieza, y la opción de ingresar la cantidad de piezas que desea agregar a la base de datos personal. Una vez ingrese los datos necesarios, en este caso un número entero positivo hasta el 65635 (esto se debe a la capacidad de la variable unsigned int, la cual va desde 0 hasta 65635), le dará al botón "agregar", el cual ingresará en la base de datos personal del usuario los datos ingresados en los pasos anteriores. En dado caso que el usuario no ingrese ningún número o un número negativo en el recuadro de cantidad, el programa le mostrará un dato de error en donde le indique que los datos ingresados no son válidos.

Eliminar piezas:

* Descripción:

Mediante esta funcionalidad se podrá eliminar cierta cantidad de piezas de la base de datos personal del usuario.

* Acciones iniciadoras y comportamiento esperado:

Al ingresar en la sección "Mis piezas", si tiene alguna pieza ingresada en su base de datos personal, tendrá la opción de eliminarla, mediante un botón con una "X", la cual, al darle click, lo llevará a una página con la descripción de la pieza, allí podía ingresar la cantidad que desea eliminar de su base de datos. Una vez ingresados los datos le dará al botón "Eliminar", el cual completará la acción de borrado de la cantidad de piezas deseadas en la base de datos personal del usuario.

* Requerimientos funcionales:

Esta funcionalidad necesita de una serie de botones que pongan en marcha las instrucciones indicadas anteriormente, los cuales van a acceder y modificar la base de datos del usuario. El primer botón que se va a encontrar es una "X" en la extensión de la página "Mis piezas" al lado de la pieza que desea eliminar, al darle click redireccionará al usuario a una página con información adicional de la pieza, y la opción de ingresar la cantidad de piezas que desea eliminar a la base de datos personal. Una vez ingrese los datos necesario, en este caso un número entero positivo hasta el 65635 (esto se debe a la capacidad de la variable int, la cual va desde 0 a 65635), le dará al botón "Eliminar", el cual removerá la ficha en la base de datos personal del usuario. En dado caso que el usuario no ingrese ningún número o un número negativo en el recuadro de cantidad, el programa le mostrará un dato de error en donde le indique que los datos ingresados no son válidos.

Ver Base de datos del usuario:

* Descripción:

Mediante esta funcionalidad se podía ver la totalidad de piezas que se encuentran en la base de datos personal del usuario.

* Acciones iniciadoras y comportamiento esperado:

Al ingresar a la página principal, podrá encontrar un botón "Mis piezas" el cual al darle click, llevará al usuario a una página en donde podrá encontrar en formato de tabla la totalidad de piezas que posee. Esta tabla contendrá una serie de columnas que indique el número de referencia de la pieza, color, cantidad y una "X" el cual accionará la opción de eliminar ya descrita anteriormente en el documento.

* Requerimientos funcionales:

Esta funcionalidad necesita de una serie de botones que pongan en marcha las instrucciones indicadas anteriormente, los cuales van a acceder y modificar la base de datos del usuario. Primero, el usuario encontrará en el menú principal, un botón que dice "Mis piezas", el cual lo redirecciona a una página en donde se encuentre la tabla de todas las piezas que posee en su base de datos personal.

Buscar en Base de datos:

* Descripción:

Mediante filtros predeterminados se podrá buscar una pieza que el usuario desee consultar de su base de datos personal.

* Acciones iniciadoras y comportamiento esperado:

Al ingresar a la página principal, podrá encontrar un botón "Mis piezas" el cual al darle click, llevará al usuario a una página en donde podrá encontrar en formato de tabla la totalidad de piezas que posee. Allí encontrará unos filtros predeterminados con 4 características por la cuales podrá buscar la ficha que desee buscar, estos son: "Color", "Set" y "# de pieza". Una vez llene los datos solicitados, le dará al botón "Buscar" el cual filtrará la base de datos con respecto a los datos ingresados por el usuario, y arrojará las piezas encontradas por el programa, con el formato de tabla ya indicado en la funcionalidad de "Ver Base de datos".

* Requerimientos funcionales:

Esta funcionalidad necesita de una serie de botones que pongan en marcha las instrucciones indicadas anteriormente, los cuales van a acceder y modificar la base de datos del usuario. Primero, el usuario encontrará en el menú principal, un botón que dice "Mis piezas", el cual lo redireccionará a una página en donde se encuentre la tabla de todas las piezas que posee en su base de datos personal. Una vez en esta página, se encontrará con una serie de filtros predeterminados, estos filtros son: "Dimension", "Color", "Set" y "# de pieza". Las opciones de "Dimension" y "Color" tendrán un menú desplegable al darles click, el cual mostrara los diferentes colores y dimensiones predeterminadas que se tiene en el sistema, en cambio las opciones de "# de pieza" y "Set" se ingresarán manualmente por el usuario escribiendo el código individual de la pieza o el set de piezas que busca. Una vez rellene los datos que desea filtrar, le dará al botón "Buscar", el cual respecto a los datos ingresados buscará en la base de datos personal del usuario, y al terminar el proceso de búsqueda, arrojará las piezas que coincidan con los datos ingresados en el formato ya indicado anteriormente de "Mis Piezas". En dado caso que no se encuentre ninguna pieza que coincida con los datos ingresados por el usuario, el programa le indicará mediante un mensaje que ninguna coincide con la descripción dada. También, si se le da al botón "Buscar" sin ingresar ningún dato, el programa arrojará un mensaje de error que indique "Por favor ingresar un dato".

Construcciones deseadas:

* Descripción:

Mediante esta funcionalidad el usuario podrá tener una lista de prioridad con sets de piezas que desee construir.

* Acciones iniciadoras y comportamiento esperado:

Al ingresar a la página principal, podrá encontrar un botón que dice "Sets", el cual lo redireccionará a una página en donde se encuentre la tabla de todas los sets que tiene la base de datos del sistema. Una vez allí podrá darle click a cualquier set, lo cual lo llevará a una página con una descripción más detallada del set. Allí podrá encontrar el botón "Agregar a mi lista de deseos" el cual al darle click, lo agregara a una cola personal, en donde el usuario podrá encontrar en el botón "Mis proyectos" en el menú principal, en donde encontrara en orden de ingreso los set que agregue a la cola.

* Requerimientos funcionales:

Esta funcionalidad necesita de una serie de botones que pongan en marcha las instrucciones indicadas anteriormente, los cuales van a acceder y modificar la base de datos del usuario. Primero, el usuario encontrará en el menú principal, un botón que dice "Sets", el cual lo redireccionará a una página en donde se encuentre la tabla de todas los sets que posee la base de datos del programa. En esta página podrá encontrar en el formato de tabla como fue indicado en la funcionalidad de "Ver Base de datos", pero con todos los sets de piezas disponibles en el sistema. Cuando el usuario encuentre el set que desee agregar a su lista de deseo, deberá darle click al botón con un símbolo "+", el cual lo agregara a una cola que se mostrará por orden de llegada de los datos (FIFO).

# descripción de la interfaz de usuario preliminar

Graphical user interface, text

Description automatically generated with medium confidence

Diagram

Description automatically generated

A picture containing diagram

Description automatically generated

Text

Description automatically generated

# entornos de desarrollo y de operación

Para esta primera entrega el programa será desarrollado en el lenguaje C# con Visual Studio 2019 para Windows. Los datos de todas las piezas de Lego se toman de la base de datos libre Rebrickable [1]. Estos datos luego son procesados con MySQL 8.0 para organizarlos y poder crear archivos .csv que serán usados en la aplicación de C#.

# prototipo de software inicial

El prototipo de software inicial se encuentra en el repositorio de GitHub <https://github.com/avila131/Estructuras-Lego>.

# implementación y aplicación de las estructuras de datos

**Listas**:

Las listas encadenadas serán usadas para el almacenamiento de set existentes tanto en la marca LEGO como en el inventario personal de cada usuario.

**Arreglos**:

Los arreglos serán implementados en la lista de las piezas de marca LEGO. Esto porque solo se tienen cerca de 45 mil piezas en total y al agregarlas a un arreglo dinámico se puede aprovechar el hecho de que estén indexadas para reducir los tiempos de búsqueda.

**Pilas**:

Se utiliza una pila para organizar todas las piezas que tiene un set:

Table

Description automatically generated

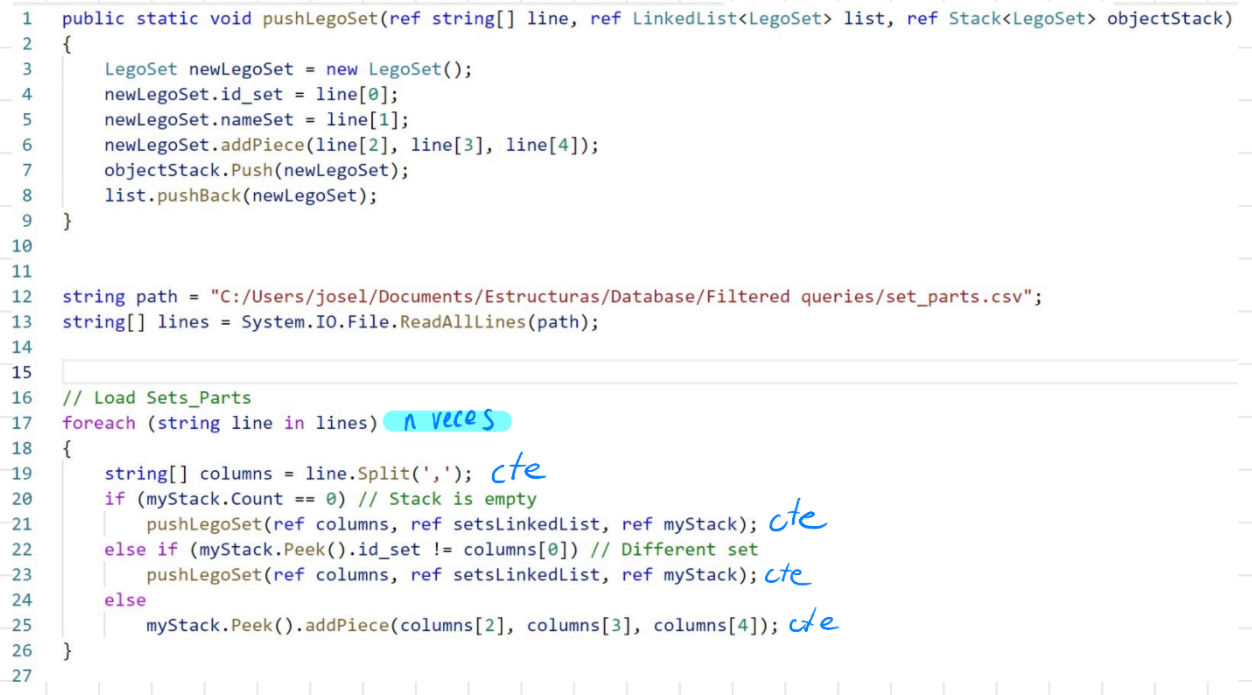
El archivo .csv tiene varias líneas con el mismo set, entonces se utiliza una pila para agrupar las piezas en un solo objeto set: Se ingresa un set a la pila y la siguiente línea tiene el mismo “SET ID” entonces no crea un nuevo objeto sino que agrega las piezas al objeto que está en la parte superior de la pila. Cuando la siguiente lectura de “SET ID” sea diferente de stack.Peek() entonces se crea un nuevo objeto, se inserta al stack y se repite el proceso.

**Colas**:

Las colas se implementarán en la “Lista de deseos” que corresponde a los sets que el usuario desea armar, aprovechando sus características para usarlas como una lista de prioridades.

# pruebas del prototipo y análisis comparativo

Cargar la base de datos O(n)



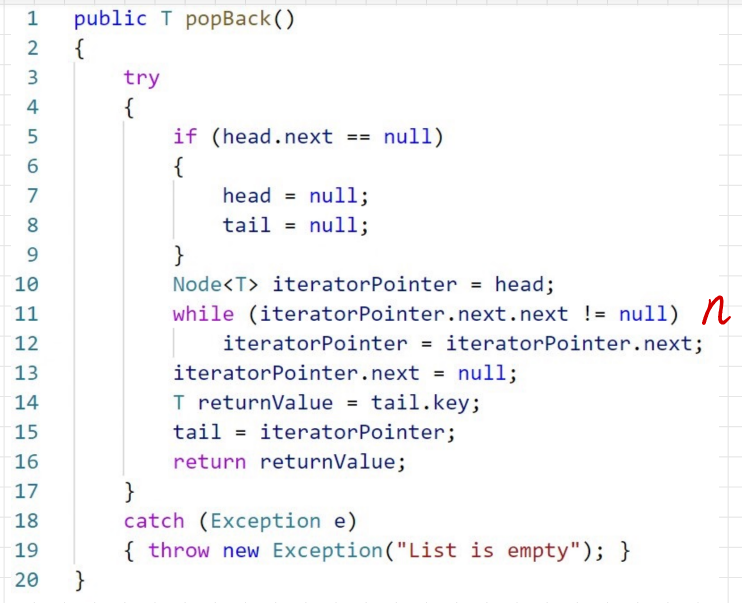
Solo hay un ciclo en esta funcionalidad. Este ciclo recorre cada uno de los registros que se encuentran en un archivo .csv. De esta manera, la notación BigO es O(n).

Eliminar una Pieza dado su ID O(n)



A pesar de que en esta funcionalidad se ven dos ciclos, el recorrido de ambos juntos es n. El procedimiento particular que se desea realizar en este caso es eliminar un elemento de un arreglo dinámico. El primer ciclo encuentra el índice del elemento que se quiere eliminar y el segundo ciclo mueve todos los elementos a la derecha de este índice una casilla a la izquierda. Como el recorrido de ambos ciclos es n, la notación BigO es O(n).

PopBack lista simplemente encadenada O(n)



Otra funcionalidad interesante es eliminar de una lista encadenada. Es importante aclarar que esta es una lista simplemente encadenada, por eso debe recorrer todos los elementos de la lista para encontrar el penúltimo elemento y poder quitar el último elemento. Con este fin se tiene un ciclo while. Como solo hay un ciclo while, la notación Big O es O(n).

**Resultados análisis comparativo**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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* |

El análisis comparativo muestra que la operación de cargar los datos de los sets es más eficiente que cargar los datos a las partes. Esto ocurre porque los sets se organizan en listas simplemente encadenadas (cuya dificultad de agregar un elemento al final es O(1)) mientras que las partes están organizadas en un arreglo dinámico. Este arreglo dinámico, a pesar de que el análisis amortizado sugiere que el costo de agregar elementos es O(1)[2], es comparativamente más lento que agregar a una lista encadenada. Incluso, el dato que se ve de las lecturas de los sets incluye reorganizar la información mediante el uso de una pila, así que el tiempo registrado en la primera columna incluye también estas operaciones de lectura y push sobre la pila. Aún así, el tiempo obtenido es menor que agregar a un arreglo dinámico.

Otra situación interesante que se observa de estas tablas es que no se tiene ningún tiempo para 100 millones de datos. Esto ocurre porque el programa obtiene sus datos a partir de un archivo .csv > 5Gb que leer, donde cada línea es un set diferente, por lo que debe crear un objeto por cada línea que lea e insertarlo a las estructuras de datos.

Además, se observa que no se tienen datos para carga de 10 millones de piezas. Esto ocurre porque se deben cargar 10 millones de piezas en un arreglo dinámico y esta operación tarda mucho. En nuestro diseño consideramos que, al solo existir cerca de 45 mil piezas diferentes en toda la historia de Lego, un arreglo dinámico era adecuado para reducir el tiempo de búsqueda y poder tener las piezas indexadas. Esto se confirma también con la tabla de resultados del análisis comparativo, donde cargar hasta 1 millón de datos se hacía en menos de 1 segundo.

En cuanto a las gráficas realizadas, se evidencia que estas corresponden a los análisis hechos a mano al inicio de esta sección de análisis comparativo. Se esperaba que las 4 operaciones propuestas fueran de complejidad O(n) lo cual corresponde con una función lineal. Este comportamiento esperado se reflejó en las gráficas realizadas, donde se evidencia un comportamiento lineal. Este dato puede confirmarse además porque coeficientes de correlación lineal son muy cercanos a .

Por último, en cuanto a factores que pueden afectar los resultados de la tabla de análisis comparativo: El programa se ejecutó a través del depurador del IDE Visual Studio 2019. Es necesario ejecutar el archivo C# de esta manera para poder tomar los tiempos de ejecución. Este IDE requería bastante memoria RAM para ejecutarse, y pudo haber afectado los resultados obtenidos. Además, la máquina en la que se prueban los datos tiene 4GB de memoria RAM, lo cual puede también limitar la cantidad de objetos que se pueden crear en tiempo de ejecución.

# información de acceso al video demostrativo del prototipo de software

[Video Demostrativo](https://www.loom.com/share/2564321aefd44cef92005887184090fc?sharedAppSource=personal_library.)

# roles y actividades

|  |  |
| --- | --- |
| **Rol** | **Actividades fundamentales** |
| Líder(esa) | Consultar a los otros miembros del equipo, atento que la información sea constante para todos. Aportar con la organización y plan de trabajo. |
| Coordinador(a) | Mantener el contacto entre todos,  Programar y agendar reuniones; ser facilitador para el acceso a los recursos. |
| Experto(a) | Líder técnico que propende por coordinar las funciones y actividades operativas. |
| Investigador(a) | Consultar otras fuentes. Propender por resolver inquietudes comunes para todo el equipo. |
| Observador(a) | Siempre está atento en el desarrollo del proyecto y aporta en el momento apropiado cuando se requiera apoyo adicional por parte del equipo. |
| Animador(a) | Energía positiva, motivador en el grupo. |
| Secretario(a) | Se convierte en un facilitador de la comunicación en el grupo. Documenta (actas) de los acuerdos/compromisos realizados en las reuniones del equipo. |
| Técnico(a) | Aporta técnicamente en el desarrollo del proyecto. |

|  |  |  |
| --- | --- | --- |
| INTEGRANTE | ROL(ES) | ACTIVIDADES REALIZADAS (Listado) |
| José Luis Ávila Guzman | Observador | Codificación de las estructuras |
| Creación del repositorio |
| Secretario | Creación de la interfaz |
| Organización de tareas |
| Sebastian Felipe Ruiz Espinosa | Líder | Documento escrito |
| Presentación |
| Investigador | Inquietudes y observaciones |
| Revisión de calidad |

# dificultades y lecciones aprendidas

El hecho de trabajar con un lenguaje de programación con el que no se está tan familiarizado representa una gran dificultad al tener que aprender a estructurarlo y usarlo de forma eficiente para la optimización del proyecto con la cantidad de datos que se busca manejar. La organización de las tareas en equipo y los plazos se vuelven difíciles de acomodar al tener que tomar en cuenta las disponibilidades de cada uno y de todos a la vez. Además, el tener que implementar las estructuras de datos por mano propia genera una capa más de complejidad a la realización del proyecto.

Por otra parte, pensábamos que el tiempo de agregar al final a una lista encadenada con cola y a un arreglo dinámico iba a ser igual porque en clase concluimos que ambas operaciones tenían una dificultad de O(1). Sin embargo, al realizar el análisis comparativo nos dimos cuenta de que la lista encadenada es mucho más eficiente al agregar elementos.

1. Bibliografía

[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].