**Base de Datos LEGO**

**Martin Polanco Barrero**

# introducción

En el mundo de lego, a veces lo más difícil es 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 ensamblarla 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 “Basededatoslegoinador”, se pretende mejorar la experiencia al momento de clasificar y organizar las fichas de lego que se posean, en donde también se va a encontrar un buscador, el cual encontrara 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 poder 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 lego, dispone de 9000 piezas diferentes, disponibles en 58 colores1. 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 sirve para las instituciones, 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.

# requerimientos funcionales del software

Este sistema debe poder almacenar los datos que se deseen, teniendo la posibilidad de eliminarlos. También debe poder filtrarlos por medio de un id que el usuario debe conocer previamente del fabricante LEGO, con la posibilidad de proyectarlos en pantalla, individual, por grupo o en la totalidad de los datos guardados.

Los datos de ingreso pueden entrar de dos formas, individual o por grupo, cada uno con un formato diferente. Al ser ingresados individual, el programa le pide cada uno de los parámetros que da el software para almacenar los datos, un ID único de la pieza, su nombre y una breve descripción. El programa le va a pedir estos datos uno por uno, indicando claramente en consola cual es cual. Si desea ingresar los datos en grupo debe ir en un formato especial, el cual consiste en [[ID,Nombre,Descripcion],…,[ID,Nombre,Descripcion]]. El software leerá el texto y filtrará los datos ingresados. Cada vez que se ingrese un dato, el programa automáticamente lo ordenara en la lista encadenada en la cual se almacenan los datos, mediante un bubblesort. Este orden se determina por el ID de la pieza ingresada, estando en la cabeza de la lista el ID con menos número, y en la cola, el ID con mayor numero. Estos datos los puede ingresar cualquier usuario.

Funcionalidades:

## Ingresar datos:

A esta función se ingresa oprimiendo la letra I o i, posteriormente va arrojar dos opciones, ingresar los datos individualmente con la letra I o i, o por grupo con la letra L o l. Si se selecciona ingresar individualmente, le indicara por un mensaje que ingrese primero el ID, después el nombre y por ultimo una descripción breve de la pieza. Si se escoge ingresar en grupo, se debe digitar los datos que se deseen en un formato especial, el cual es el siguiente:

[[ID,Nombre,Descripcion],…,[ID,Nombre,Descripcion]]

Una vez si ingresan estos datos, se llama a la función bubblesort, la cual organiza los datos de acuerdo con sus ID’s de manera ascendente.

## Contar el número de piezas:

Cuando se tiene ya unos datos ingresados, ingresando la letra N o n en el menú principal, se llama a la función contar el número de piezas, la cual le dará dos opciones al usuario, mostrarle el número de piezas que tiene en total oprimiendo la letra T o t, o por agrupación de ID’s oprimiendo la letra C o c.

## Imprimir los datos:

Al oprimir la letra D o d se llama a esta función, la cual le da dos opciones al usuario; la primera es imprimir todos los datos contenidos en el programa oprimiendo la letra T o t, y la segunda es imprimir por el ID de un conjunto de piezas oprimiendo la letra U o u.

* *Eliminar:*

Esta función se llama ingresando la letra E o e en el menú principal, dandole 3 opciones al usuario. La primera es eliminar a un elemento por ID mediante la letra U o u, esta elimina el primer elemento que encuentre con el ID ingresado. La segunda opción es eliminar un conjunto de elementos por ID ingresando la letra C o c, por lo tanto si se invoca, todos los elementos que tengan el ID ingresado, se eliminaran. La tercera y última opción es eliminar todos los elementos de la lista oprimiendo la letra T o t.

## Salir del sistema:

La última función es la de salir del programa, el cual al ingresar la letra X o x, acabara el proceso del programa en ejecución.

*Requerimientos funcionales*:

El almacenamiento, creación y eliminación de datos se hace mediante una lista encadenada simple con cola, la cual se ordena mediante un bubble sort cada vez que se ingresa un dato o un conjunto de datos nuevos.

Como los datos están organizados mediante los ID’s de las piezas, los cuales se encuentran de manera ascendente, la eliminación y consulta de datos es más fácil, ya que se pueden eliminar y buscar conjuntos de datos enteros con algoritmos sencillos.

En dado caso de que se ingrese datos incorrectos, como por ejemplo los ID’s que se guardan como enteros, se hace un try catch, el cual detecta si se presenta un error, presentando un mensaje que le indica al usuario que ingreso mal los datos y se repite la función en la que se encontraba este usuario. Y para el ingreso de las letras para acceder a las diferentes funciones del programa, se utilizó un switch case, el cual si no detecta las letras establecidas por el programa, arrojara un mensaje de error y volverá a presentarle la función al usuario para que ingrese los datos correctos.

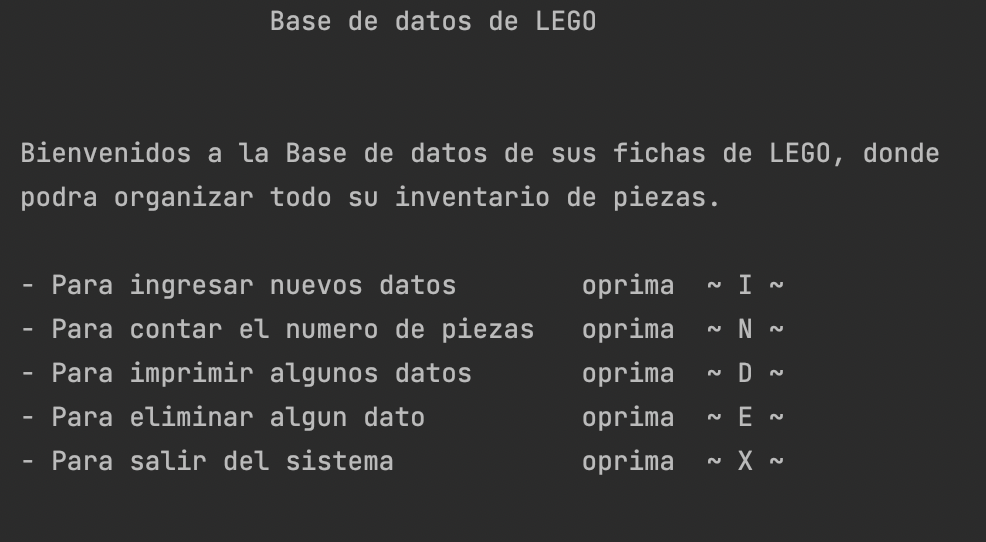
El usuario podrá acabar la ejecución del programa en cualquier momento oprimiendo la letra X o x.

# descripción de la interfaz de usuario preliminar

Este primer prototipo se ejecuta mediante la consola del ide en donde se trabajó el código, el cual es IntelliJ Idea. Mediante un menú básico que se presenta en pantalla, se le indica al usuario las diferentes instrucciones que debe seguir en el programa.

# entornos de desarrollo y de operación

En esta sección se debe describir el entorno en el que se desarrollará el software, así como el entorno en el que funcionará cuando se ponga en operación, para esto último se debe especificar el sistema operativo y el hardware sobre el que operará.



# prototipo de software inicial

Phttps://github.com/mpolancob/LEGO-APP

# diseño, implementación yaplicación de las estructuras de datos

En este prototipo se implementó una lista encadenada simple con cola LIFO, la cual se utiliza como almacenamiento de los datos que se ingresan al sistema, ingresando los datos por la cola de la lista. En cambio, la eliminación solo se hace mediante una posición especial, ya que se le pregunta al usuario que ID de pieza es el que desea eliminar, llegando hasta el nodo donde se encuentre y sacándolo de la lista. Para facilitar la agrupación y búsqueda de los datos, cada vez que se ingresa un dato o un conjunto de datos, se le hace un bubble sort a la lista, mirando si un ID de una pieza es mayor al de la siguiente, pero para reubicar esta información, no se mueven los nodos, si no que se guardan los datos de los nodos en un almacenamiento temporal y se actualizan los datos de los nodos con estos nuevos datos guardados.

# pruebas del prototipo y análisis comparativo

Para inserción en la lista enlazada:

* + 10 mil datos = 51 milisegundos
  + 100 mil datos = 175 milisegundos
  + 1 millón de datos = 188 milisegundos
  + 10 millones de datos = 207 milisegundos
  + 100 millones de datos = 241 milisegundos

# información de acceso al video demostrativo del prototipo de software

<https://www.loom.com/share/feae80ef8c8c47ee8aab4f07568c8775>

# roles y actividades

|  |  |  |
| --- | --- | --- |
| **Integrante** | **Rol** | **Actividad** |
| Martin Polanco | Líder | Desarrollo del prototipo y de los entregables. |

# dificultades y lecciones aprendidas

Tuve problemas al comunicarme con mis compañeros, ya que, aunque el grupo fue creado hace por lo menos 2 semanas, y yo como líder de grupo les escribí al grupo para ver cuando comenzábamos, no hubo respuesta alguna. Faltando 4 días para el proyecto se comunicaron en el grupo, para reunirse, pero solo 2 pudimos (incluyéndome) pudimos asistir a la reunión, en donde establecimos el lenguaje y las herramientas que íbamos a utilizar. Paso el fin de semana y al ver que ninguno decía nada, les escribí a cada uno el domingo, para por lo menos hacer el parcial, la respuesta fue poca. Por lo tanto, decidí hacer y presentar el primer prototipo solo.

# referencias bibliográficas