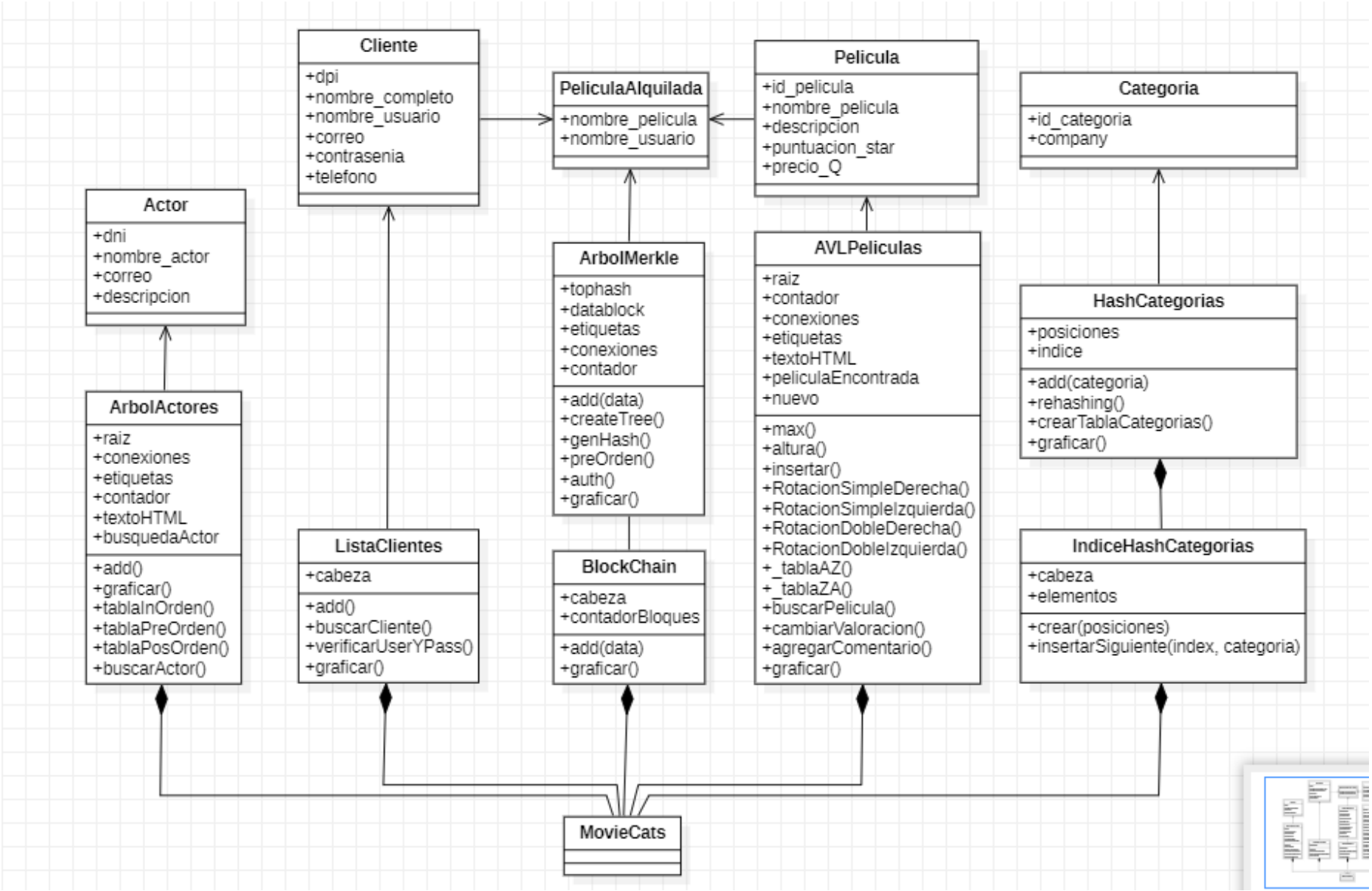
Manual de Técnico

MovieCats

Por Keneth Ramiro Urrutia Diaz - 201940854

Contenido

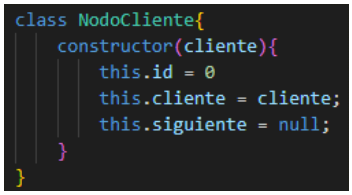
Diagrama de clases



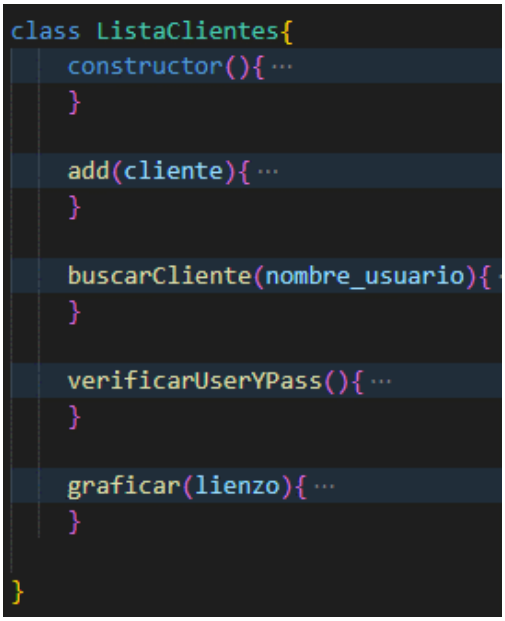
Estructuras de datos

Clientes (lista simplemente enlazada)

Se creo un nodo Cliente el cual formara parte de una lista simplemente enlazada, que representara una lista con todos los clientes de MovieCats.



La estructura es creada a partir de una clase ListaClientes que contiene los metodos necesarios para el funcionamiento de la pagina:



add: agrega un nuevo cliente a la lista.

buscarCliente: enviamos como parámetro el nombre de un cliente y retorna un objeto de tipo Cliente solicitado.

graficar: crea y muestra una vista grafica de la estructura.

Árbol de actores (Árbol binario)

Se creo un nodo Actor el cual formara parte del árbol binario, que representara una tabla con la información de los actores.

```
class NodoActor{
  constructor(actor){
      this.izquierda = null;
      this.derecha = null;
      this.actor = actor;
      this.id = 0;
  }
}
```

```
class ArbolActores{
  constructor(){ ···
  }

  add(actor){ ···
  }

  _graficar(nodo){ ···
  }

  graficar(lienzo){ ···
  }

  _tablaPreOrden(nodo){ ···
  }

  _tablaInOrden(nodo){ ···
  }

  _tablaPosOrden(nodo){ ···
  }

  buscarInOrden(nodo, nombre_actor){ ···
  }

  buscarActor(nombre_actor){ ···
  }

}
```

add: agrega un nuevo actor al árbol.

graficar: método que llama a la función recursiva _graficar.

_graficar: crea y muestra una vista grafica de la estructura.

tablaPreOrden: crea de manera recursiva una tabla donde se muestran todos los actores en pre-orden.

tablaInOrden: crea de manera recursiva una tabla donde se muestran todos los actores en in-orden.

tablaPosOrden: crea de manera recursiva una tabla donde se muestran todos los actores en pos-orden.

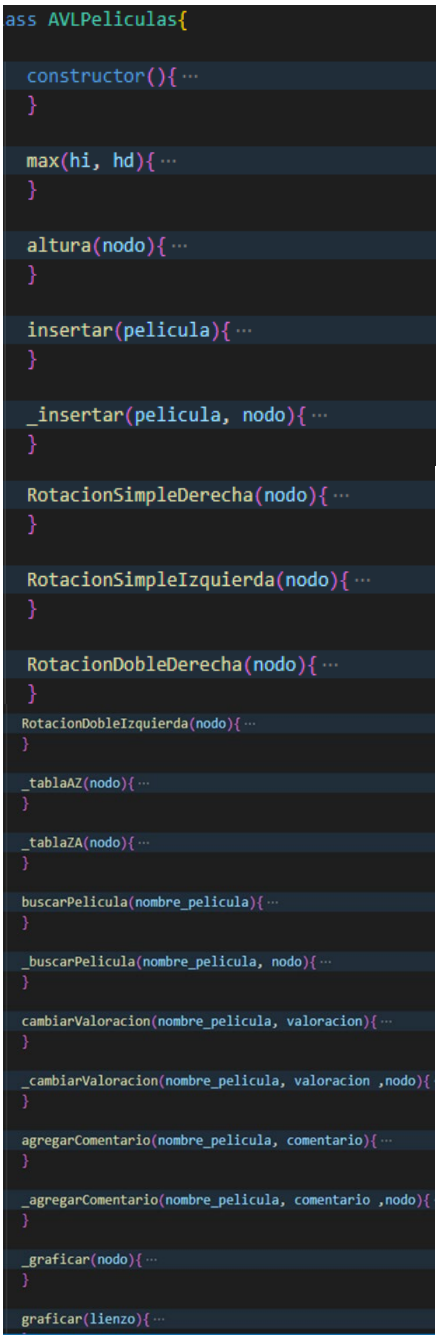
buscarInOrden: retorna un objeto tipo Actor que buscamos.

buscarActor: método que llama a la función recursiva buscarInOrden.

Árbol de películas (Árbol AVL)

Se creo un nodo Película el cual formara parte del árbol AVL, que representara una tabla con la información de las películas.

```
class NodoPelicula{
  constructor(pelicula){
      this.izquierda = null;
      this.derecha = null;
      this.pelicula = pelicula;
      this.altura = 0;
      this.id = 0;
      this.comentarios = new Array();
  }
}
```



max: se envia de parametro la altura de dos nodos y retorna el mas grande de los dos.

altura: retorna la altura de el nodo que enviamos de parametro

insertar: método que llama a la función recursiva _insertar.

_insertar: inserta una nueva película al árbol.

RotacionSimpleDerecha: realiza una rotación simple a la derecha del nodo que enviamos de parámetro.

RotacionSimpleIzquierda: realiza una rotación simple a la izquierda del nodo que enviamos de parámetro.

RotacionDobleDerecha: realiza una rotación doble a la derecha del nodo que enviamos de parámetro.

RotacionDobleIzquierda: realiza una rotación doble a la izquierda del nodo que enviamos de parámetro.

_tablaAZ: crea una tabla con todas las películas ordenadas de menor a mayor.

_tablaZA: crea una tabla con todas las películas ordenadas de mayor a menor.

_buscarPelicula: retorna un objeto tipo Pelicula que buscamos.

buscarPelicula: método que llama a la función recursiva _ buscarPelicula.

cambiarValoracion: método que llama a la función recursiva _cambiarValoracion.

_cambiarValoracion: cambia la valoración de la película que ingresamos de parametro.

agregarComentario: método que llama a la función recursiva _agregarComentario.

_agregarComentario: agrega un comentario a la película que enviamos de parametro.

graficar: método que llama a la función recursiva _graficar.

_graficar: crea y muestra una vista grafica de la estructura.

Categorías (Tabla Hash)

Se creo un nodo Categoría el cual formara parte de la tabla hash, también se usó un nodo IndiceHash el cual formara parte de una lista que será el índice de la tabla hash.

```
class NodoIndiceHash{
  constructor(index){
    this.index = index;
    this.siguiente = null;
    this.abajo  = null;
  }
}
```

```
class NodoCategoria{
    constructor(categoria){
      this.id = 0;
      this.categoria = categoria
      this.siguiente = null;
    }
}
```

```
class IndiceHashCategorias{
  constructor(posiciones){···
  }

  crear(posiciones){···
  }

  insertarSiguiente(index, categoria){···
  }

}

class HashCategorias{
  constructor(posiciones){···
  }

  add(categoria){···
  }

  rehashing(){···
  }

  crearTablaCategorias(){···
  }

  graficar(lienzo){···
  }
}
```

Se utilizo una clase IndiceHashCategorias para que haga de indice de la tabla hash.

crear: crea el indice de la tabla hash con la cantidad de posiciones que enviamos de parámetro.

insertarSiguiente: inserta una nueva categoría en el indice que enviamos de parámetro.

Se utilizo una clase HashCategorias que es la estructura principal.

add: agrega una nueva categoría a la tabla.

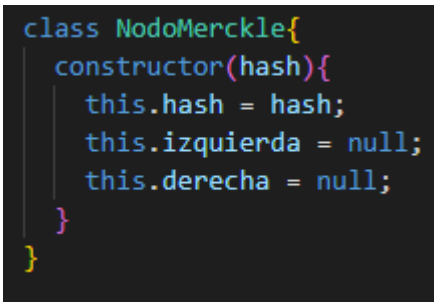
rehashing: hace el rehashing de la tabla.

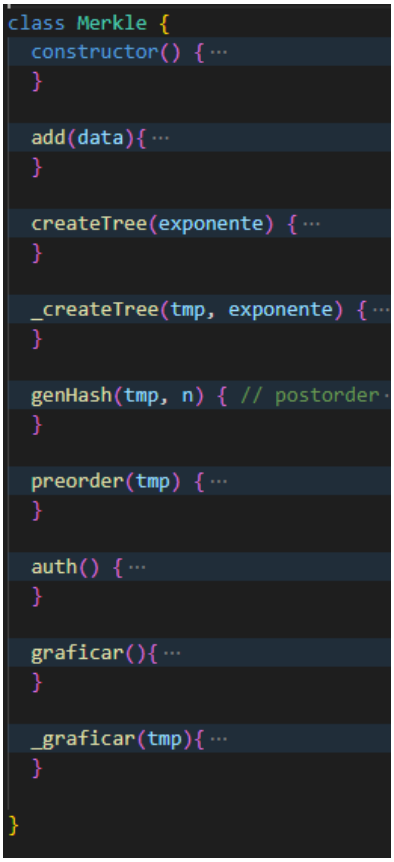
crearTablaCategorias: crea una lista con todas las categorías de la tabla hash.

graficar: crea y muestra una vista grafica de la estructura.

Películas Alquiladas (Árbol de Merkle y Blockchain)

Se creo un nodo Merkle el cual formara parte del árbol de Merkle, también se usó una clase Bloque el cual formara parte la blockchain.





Árbol de Merkle:

add: agrega un nuevo nodo al árbol.

createTree: método que llama a la función recursiva _createTree.

_createTree: crea los nodos necesarios para que el árbol este completo según la cantidad de data ingresada.

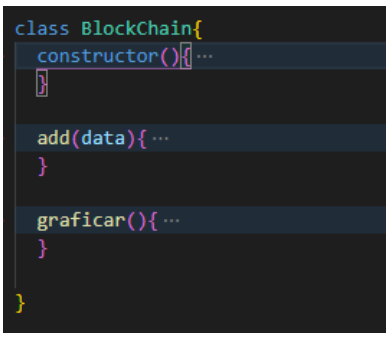
genHash: llena la data de los nodos creados con _createTree.

preorder: hace un recorrido en pre-orden del árbol.

auth: ejecuta los métodos createTree, genHash, y preorder.

graficar: método que llama a la función recursiva _graficar.

_graficar: crea y muestra una vista grafica de la estructura.



BlockChain:

add: agrega un nuevo bloque a la blockchain;

graficar: crea y muestra una vista grafica de la estructura.