**TRABAJO PRÁCTICO**

**–**

**DISEÑO DE SISTEMAS**

**–**

**Entrega anual parte 2**

**Integrantes:**

. Emiliano Somoza

. Gabiel Freijomil

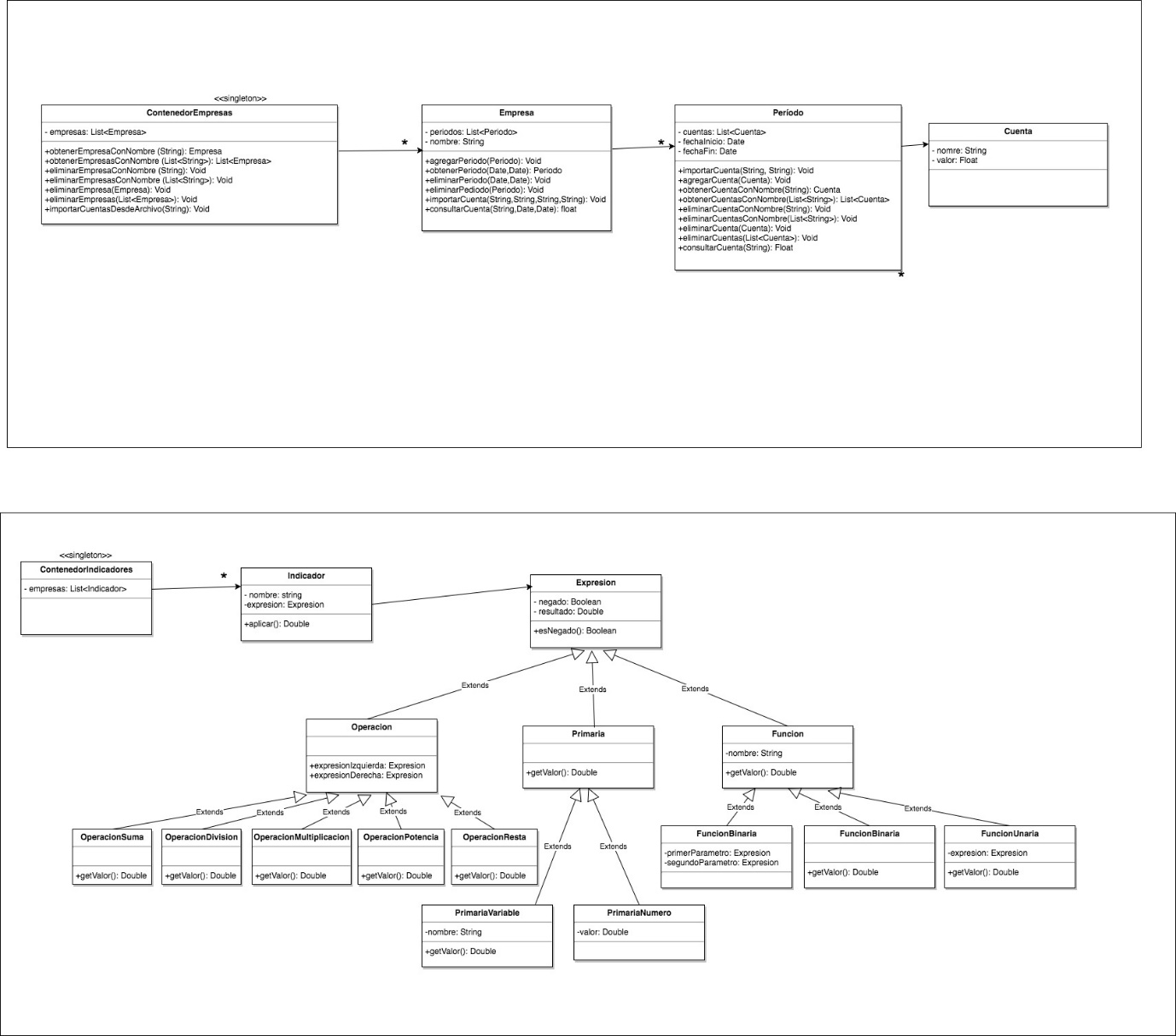
. Alunni Pablo

. Andrés

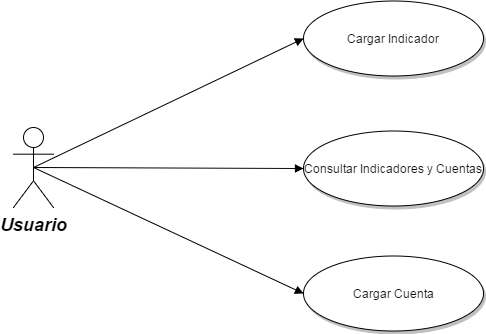
. Rodrigo

**Especialidad:** Ingeniería en Sistemas de Información.

**Diagrama de clases:**



**Diagrama de Casos de Uso:**



**Especificación CU’s**

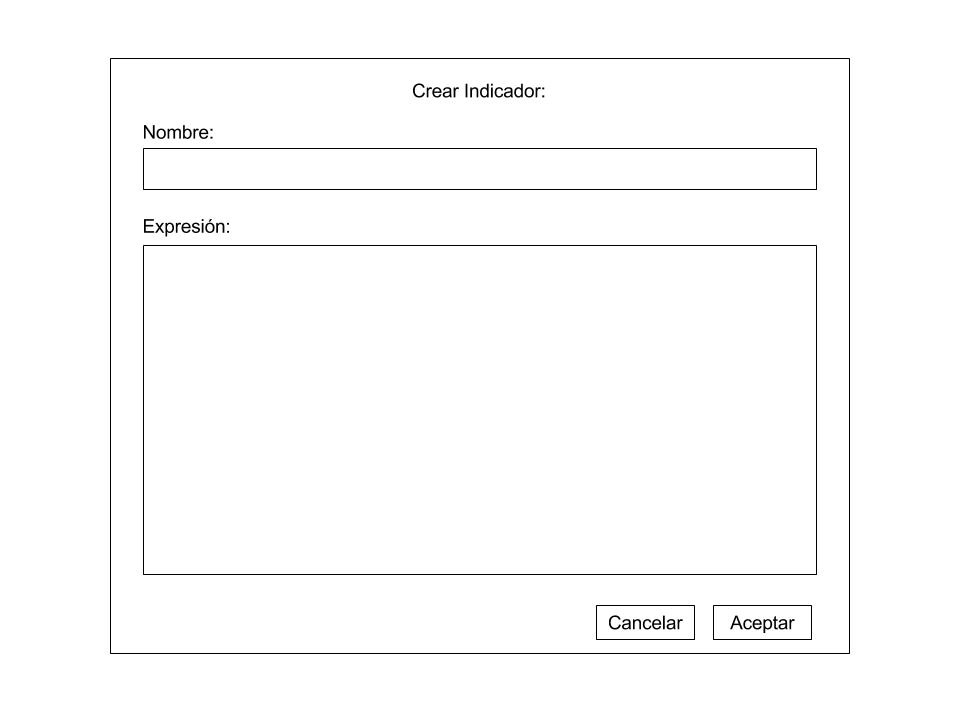
|  |  |
| --- | --- |
| Caso de Uso | Consultar Indicadores y Cuentas |
| Actores | Usuario |
| Precondiciones |  |
| Flujo Principal | 1. El usuario completa los filtros de Nombre de Empresa y/o Periodo.  2. El usuario presiona el botón “Consultar”.  3. La aplicación desplega una grilla donde muestra como resultado las cuentas y los indicadores existentes para las empresas y periodos proporcionados como filtros por el usuario. |
| Flujo Alternativo | 3.1 No hay resultados para mostrar y la grilla se ve vacía |
| Excepciones |  |
| Post-condiciones | Obtener un listado de las cuentas acorde a los filtros ingresados |

|  |  |
| --- | --- |
| Caso de Uso | Cargar Cuenta |
| Actores | Usuario |
| Precondiciones |  |
| Flujo Principal | 1. El usuario presiona el botón Cargar Datos.  2. El sistema abre un seleccionador de archivos para que el usuario indique el origen del archivo a cargar  (un único archivo).  3. El usuario selecciona el archivo TXT a cargar.  4. El sistema guarda la información de la cuenta ingresada.  5. El sistema muestra el resultado de la cuenta recientemente ingresada. |
| Flujo Alternativo |  |
| Excepciones | E1. El archivo no posee el formato esperado. |
| Post-condiciones | La creación de una nueva cuenta en el sistema |

|  |  |
| --- | --- |
| Caso de Uso | Cargar Indicador |
| Actores | Usuario |
| Precondiciones |  |
| Flujo Principal | 1. El usuario elige en el menú la opción “Carga Indicador”.  2. El usuario ingresa nombre del Indicador y la expresión que la compone.  3. El usuario clieckea sobre el botón Guardar. |
| Flujo Alternativo | 3.1 El usuario decide no cargar el indicador y clickea sobre “Cancelar”. |
| Excepciones | E1. La expresión ingresada no se corresponde con la gramática establecida. |
| Post-condiciones | La creación de un nuevo indicador. |

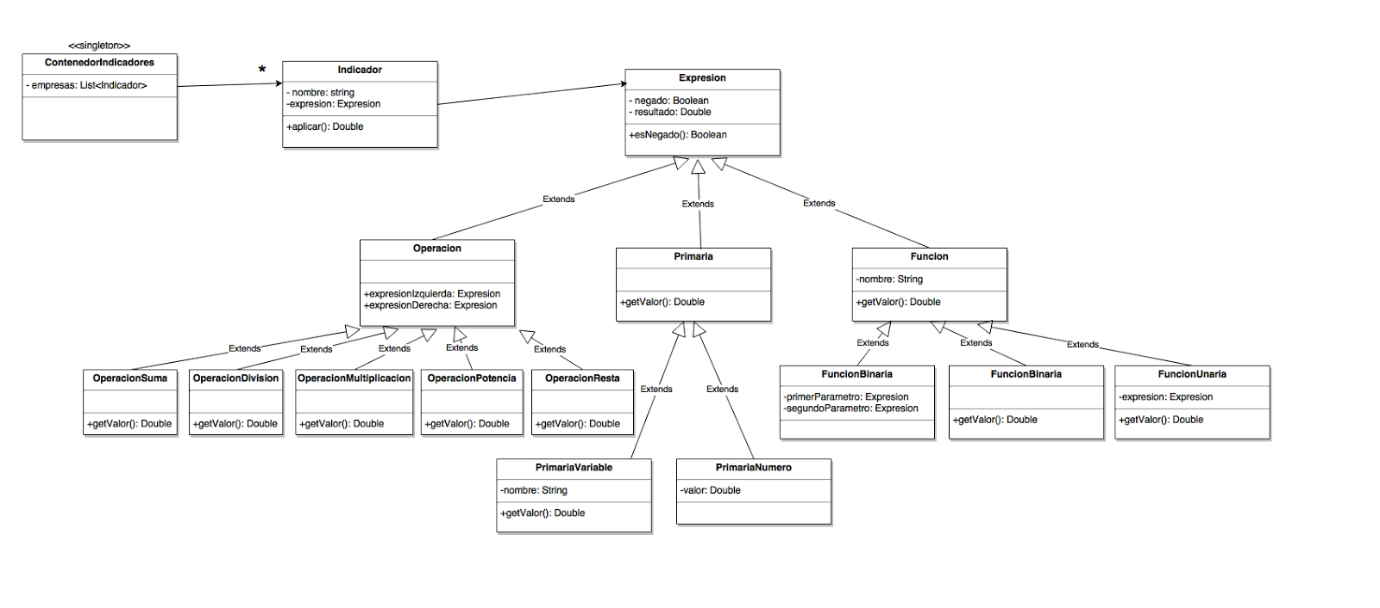
**Indicadores:**

**Interfaz para creación de indicadores:**



Representación:

El indicador está compuesto por un nombre y una expresión, como muestra el siguiente diagrama:



Como se puede ver la expresión es un árbol binario de operaciones que se resuelven con una llamada recursiva desde la raíz hasta las hojas.

Para almacenarlos decidimos utilizar una Base de datos documental (del estilo de MongoDB) ya que es más fácil y más claro serializar la estructura de la expresión que representarla en tablas. Para mantener la performance los indicadores serán cacheados cuando se inicie la aplicación. Por ahora el caché será un singleton ya que no prevemos más de 100 o 200 indicadores generados por el usuario, si este número aumentará considerablemente necesitamos reemplazarlo por una solución como Redis o Elastic Search.

Carga:

Para crear el indicador el usuario ingresara en una pantalla el nombre en un campo y la expresión matemática que representa en otro. La expresión debe respetar el lenguaje generado por la gramática que se explica a continuación.

Reglas:

<expression> -> <termino>

|<expression> MULTIPLICATIVE\_OP <expression>

|<expression> ADDITIVE\_OP <expression>

<between\_parenthesis> -> LPAR <expression> RPAR;

<signed\_termino> -> ADDITIVE\_OP terminal

| ADDITIVE\_OP raise\_operation

<termino> -> <terminal>

| <raise\_operation>

| <signed\_termino>

<raise\_operation> -> <terminal> RAISE\_OP <terminal>

| <terminal> RAISE\_OP <raise\_operation>

| <terminal> RAISE\_OP <expression>

<function> -> <function\_static>

| <function\_unary>

| <function\_binary>

<function\_static> -> FUNC\_STATIC LPAR RPAR

<function\_unary> -> FUNC\_UNARY LPAR <expression> RPAR

<function\_binary> -> FUNC\_BINARY LPAR expression COMMA expression RPAR

<primary> -> VAR

| DOUBLE

Tokens:

FUNC\_STATIC: PI\_FUNC

PI\_FUNC: 'pi'

FUNC\_UNARY: SQRT\_FUNC

SQRT\_FUNC: 'sqrt'

FUNC\_BINARY: ROOT\_FUNC

ROOT\_FUNC: 'root'

VAR: WORD ('\_' WORD)? NUMBER?

DOUBLE: NUMBER ('.' NUMBER)?

NUMBER: DIGIT+

DIGIT: ('0'..'9')

WORD: LETTER+

LETTER: ('a'..'z' | 'A'..'Z')

ASSIGN: '='

LPAR: '('

RPAR: ')'

COMMA: ','

RAISE\_OP: '^'

MULTIPLICATIVE\_OP: '\*' | '/'

ADDITIVE\_OP: '+' | '-'

Procesamiento

Para procesar la expresión decidimos utilizar Antlr ya que lo recomienda la cátedra y la manera de escribir una gramática es similar a lo que se explica en la materia Sintaxis y Semántica de los Lenguajes.

Antlr en base a la gramática especificada genera el parser y el lexer que la interpreta y te provee listeners y visitors para poder hacer el análisis semántico. Nosotros particularmente decidimos utilizar visitors por la naturaleza recursiva de nuestra gramática.

**Diagrama de Arquitectura:**

