

Catedráticos: Ing. Bayron López, Ing. Edgar Sabán, Ing. Erick Navarro

Tutores académicos: José Cano, Daniel Álvarez, Mike Gutiérrez y Javier Navarro.

Form-Usac

Primer proyecto de laboratorio

Tabla de contenido

1	Objetivos	5
1.1	Objetivo general	5
1.2	Objetivos específicos	5
2	Descripción	5
2.1	Antecedentes	5
2.2	Propuesta de la solución	5
2.3	Flujo de la aplicación.....	7
3	FORM-USAC	8
3.1	Excel	8
3.2	Componente de interpretación XLS a Lenguaje de alto nivel	8
3.3	Componente para generación de formulario	9
4	Estructura del Archivo XLS	11
4.1	Características del lenguaje	11
4.2	Sintaxis para la elaboración de formularios	12
4.2.1	Archivo XLS	12
4.2.2	Hoja de trabajo de Encuestas.....	12
4.2.3	Hoja de Opciones	16
4.2.4	Hoja de Configuraciones	16
4.2.5	Estilos	17
4.2.6	Tipos de Columnas.....	19
4.2.7	Tipos de Pregunta	24
4.2.8	Agrupaciones de Preguntas	31
4.2.9	Operadores y funciones del formulario	34

4.2.10	Estilos a los textos.....	35
5	Estructura del Lenguaje Alto Nivel	36
5.1	Notación dentro del enunciado.....	36
5.2	Características del lenguaje.....	37
5.3	Case Insensitive.....	37
5.4	Polimorfismo	37
5.5	Recursividad	37
5.6	Tipos de datos.....	37
5.7	Signos de agrupación	39
5.8	Almacenamiento de las respuestas	39
5.9	Sintaxis del lenguaje	39
5.9.1	Comentarios	39
5.9.2	Operaciones Aritméticas.....	40
5.9.3	Operadores.....	40
5.9.4	Condiciones.....	42
5.9.5	Expresiones relacionales.....	42
5.9.6	Expresiones lógicas.....	42
5.9.7	Precedencia de operadores y Asociatividad.....	43
5.9.8	Estructura general	43
5.9.9	Visibilidad	43
5.9.10	Carácter de asignación	44
5.9.11	Importaciones.....	44
5.9.12	Clases	44
5.9.13	Herencia.....	45
5.9.14	Atributos	45
5.9.15	Declaración de variables	46
5.9.16	Asignación de variables	46
5.9.17	Matrices.....	46
5.9.18	Declaración e instanciación de objetos.	48
5.9.19	Métodos y funciones	48
5.9.20	Método Principal.....	49
5.9.21	Constructor.....	50
5.9.22	Acceso a atributos y métodos de un objeto.....	50
5.9.23	Sentencias de control.....	51

5.9.24	Bucles	53
5.9.25	Imprimir	55
5.9.26	Diálogos	55
5.9.27	Funciones Nativas del Lenguaje	55
5.9.28	Funciones con cadenas	55
5.9.29	Funciones Booleanas.....	56
5.9.30	Funciones numéricas	57
5.9.31	Funciones de Fecha y Hora	59
5.9.32	Funciones Multimedia	60
5.9.33	Listas de Opciones.....	61
5.9.34	Formulario y Grupos.....	62
5.9.35	Preguntas.....	63
5.9.36	Procedimiento respuesta.....	65
6	Ejemplos de Entrada y Salida	70
6.1	Tipos de Columnas	70
6.2	Hoja de Configuración.....	82
6.2.1	Estilos.....	82
6.3	Tipos de Preguntas	85
6.4	Agrupaciones	95
7	Ejemplo para generar un formulario.....	99
7.1	Pasos de interpretación del ejemplo	99
7.1.1	Paso 1	99
7.1.2	Paso 2	99
7.1.3	Paso 3	100
7.1.4	Paso 4	101
7.1.5	Paso 5	101
7.1.6	Paso 6	102
7.1.7	Paso 7	102
7.1.8	Paso 6	103
7.1.9	Paso 9	103
7.1.10	Paso 10.....	104
7.1.11	Paso 11.....	105
7.1.12	Paso 12.....	106
7.1.13	Paso 13.....	106
7.1.14	Paso 14.....	107

7.1.15	Paso 15.....	107
7.1.16	Paso 16.....	108
7.1.17	Paso 17.....	108
8	Manejo de errores.....	109
8.1	Manejo de errores del lenguaje de alto nivel.....	109
8.2	Manejo de Errores en la interpretación de XLS	110
9	Requisitos Mínimos.....	111
10	Entregables y Restricciones	113
10.1	Entregables.....	113
10.2	Criterios de Entrega	113
10.3	Criterios de Calificación	113
10.4	Restricciones.....	114

1 Objetivos

1.1 Objetivo general

Aplicar los conocimientos del curso de Organización de Lenguajes y Compiladores 2 en el desarrollo de una aplicación.

1.2 Objetivos específicos

- Utilizar herramientas para la generación de analizadores léxicos y sintácticos.
- Implementar una solución de software que cumpla las funciones de un intérprete.
- Realizar análisis semántico a los diferentes lenguajes a implementar.
- Implementar un lenguaje de tipado estático y fuerte.

2 Descripción

2.1 Antecedentes

Desde inicios del siglo 20, las tareas de recolectar datos estadísticos a través de censos y encuestas se han estado realizando sobre papel en forma de cuestionarios. Hoy se dispone de nuevas tecnologías, pero en gran medida todavía ligadas al papel. El estado actual de las tareas de recolección y procesamiento de datos de encuestas presentan los siguientes problemas:

- Esperas mucho tiempo para el acceso a la información.
- Errores en la entrada de datos.
- Potencial pérdida de datos.
- Costo de procesamiento de los datos recogidos.
- Costos de distribución altos.
- Estructura de datos heterogénea.

Las tecnologías de información resultan una herramienta muy valiosa que podría permitir recolectar, procesar y almacenar datos que son generados por medio de encuestas, formularios u otros medios.

2.2 Propuesta de la solución

Ante esta necesidad los estudiantes del curso Compiladores 2, para el primer proyecto del laboratorio deberán desarrollar una aplicación de escritorio, la cual permitirá crear formularios personalizados y con ellos poder recopilar datos ingresados por los usuarios.

Esta aplicación tendrá el nombre USAC-FORM, estará compuesta por tres componentes principales:

1. Definición de Formulario
2. Construcción de formulario
3. Generación y llenado de formulario

El primer componente permitirá la definición de formularios, el formulario será definido en forma tabular y en un formato estándar para la creación de los mismos, donde cada columna represente atributos de una pregunta y las filas representen el conjunto de preguntas que integraran el formulario, para este fin se utilizará la herramienta de Microsoft Office Excel. Esto permite al usuario construir un formulario de una forma sencilla e intuitiva.

Para proceder con la descripción es necesario definir un formato que represente de una forma no tabular el formulario, este formato será un lenguaje de alto nivel orientado a objetos y llevará por nombre XForm, lo cual permitirá mejor interacción entre las preguntas y sus atributos.

El segundo componente permitirá la construcción de formularios en formato XForm, tras haber definido la estructura de un formulario con ayuda de la herramienta de Microsoft Office Excel, se procederá a interpretar el archivo XLS generado y al finalizar el proceso de interpretación se generará un archivo XForm como salida.

El tercer componente permitirá la visualización del formulario con sus respectivas preguntas, también permitirá el llenado del mismo por parte de los usuarios y la visualización de las respuestas recopiladas.

2.3 Flujo de la aplicación

La imagen en esta página muestra el flujo de los componentes de la aplicación en forma general, en él se observa que todo el proceso inicia con la definición de un formulario escrito en con la ayuda de la herramienta de Excel, seguido a la fase de construcción del formulario y por último en la fase de visualización y llenado del formulario.

Diagrama de Flujo

Definición del formulario



Se procede a abrir la herramienta de Office Excel para crear un nuevo archivo XLS.



A	B	C	D	E
1	2	3	4	5
1	1	Jack	Software Engineering	95
2	2	John	Software Engineering	90
3	3	Michael	Software Engineering	85
4	4	Michael	Software Engineering	80
5	5	Michael	Software Engineering	75
6	6	Michael	Software Engineering	70
7	7	Michael	Software Engineering	65
8	8	Michael	Software Engineering	60
9	9	Michael	Software Engineering	55
10	10	Michael	Software Engineering	50
11	11	Michael	Software Engineering	45
12	12	Michael	Software Engineering	40
13	13	Michael	Software Engineering	35
14	14	Michael	Software Engineering	30
15	15	Michael	Software Engineering	25
16	16	Michael	Software Engineering	20
17	17	Michael	Software Engineering	15
18	18	Michael	Software Engineering	10
19	19	Michael	Software Engineering	5
20	20	Michael	Software Engineering	0

Se define un formulario en el estándar propuesto para su creación.



Se guardan los cambios en el archivo XLS.

Construcción del formulario



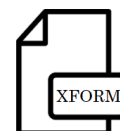
Se recibirá el archivo XLS creado en Excel como entrada,



Tras el proceso de interpretación se generará un archivo XForm.



Visualización y llenado de formulario



Se recibirá un archivo XForm como entrada, luego será posible la visualización del mismo así como su llenado.



EJEMPLO 1

Los campos marcados con * son de inserción obligatoria

Asunto:

* Primer Apellido:

* Segundo Apellido:

* Nombre:

D.N.I. / Pasaporte / N.I.E.:

* Correo electrónico:

Teléfono:

¿Qué va a escribir?:

* Texto del escrito:

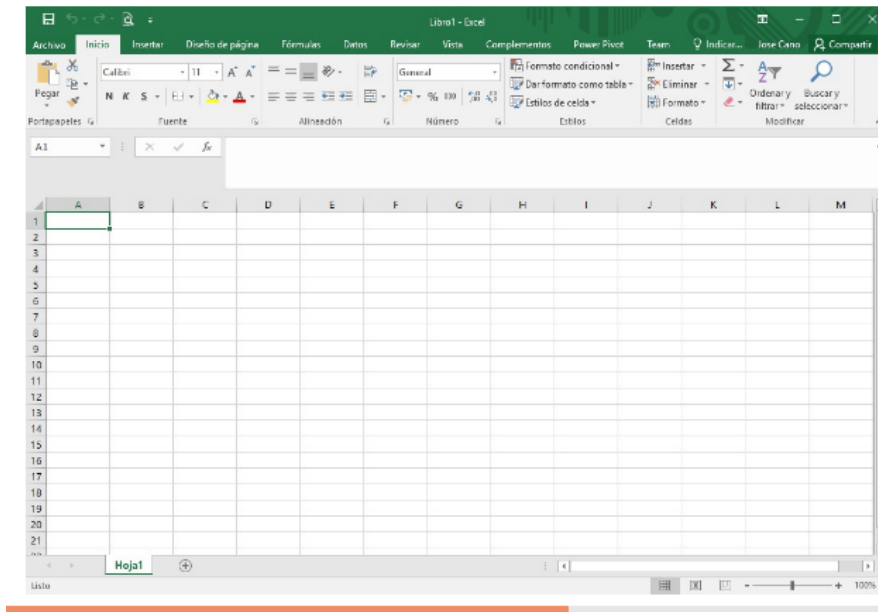
Los datos personales facilitados son susceptibles de cesión cuando ésta sea precisa para tramitar su contestación.

3 FORM-USAC

A continuación, se describen los componentes que conformarán la aplicación.

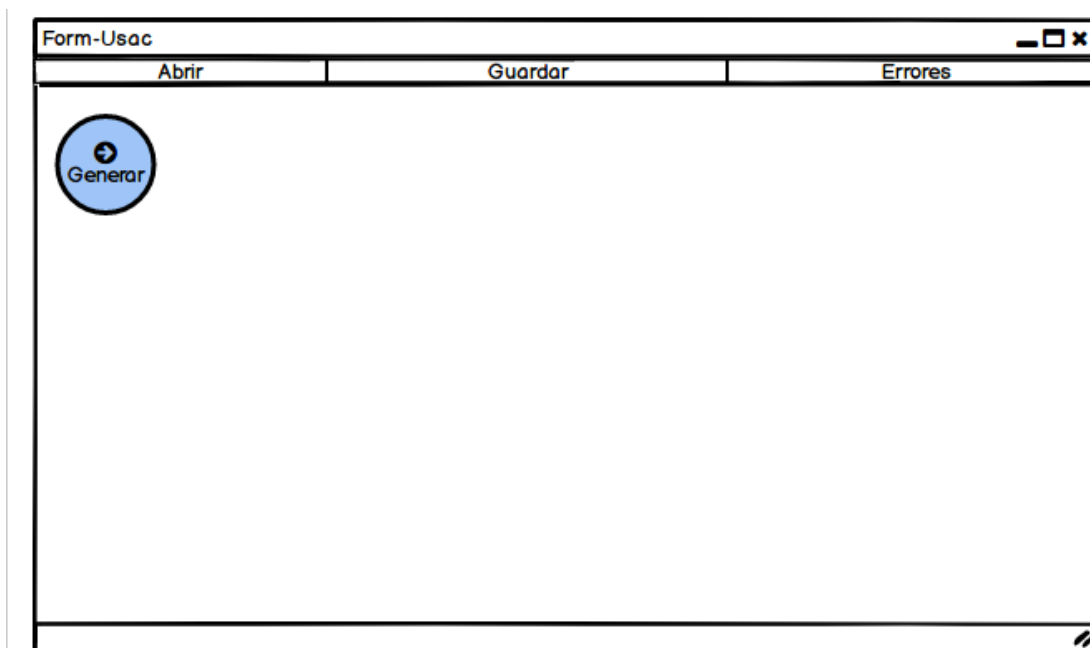
3.1 Componente para la definición del formulario

Este componente estará asociado a la herramienta de Microsoft Office Excel ya que con ayuda de la misma será posible ingresar las preguntas que se quiere mostrar al usuario en forma tabular, donde cada columna representará los atributos de cada pregunta.



3.2 Componente de construcción de formularios

Este componente servirá para generar archivos en formato XForm, estos archivos XForm se crearán a partir del archivo de Excel que fue explicado anteriormente.



Barra de Herramientas

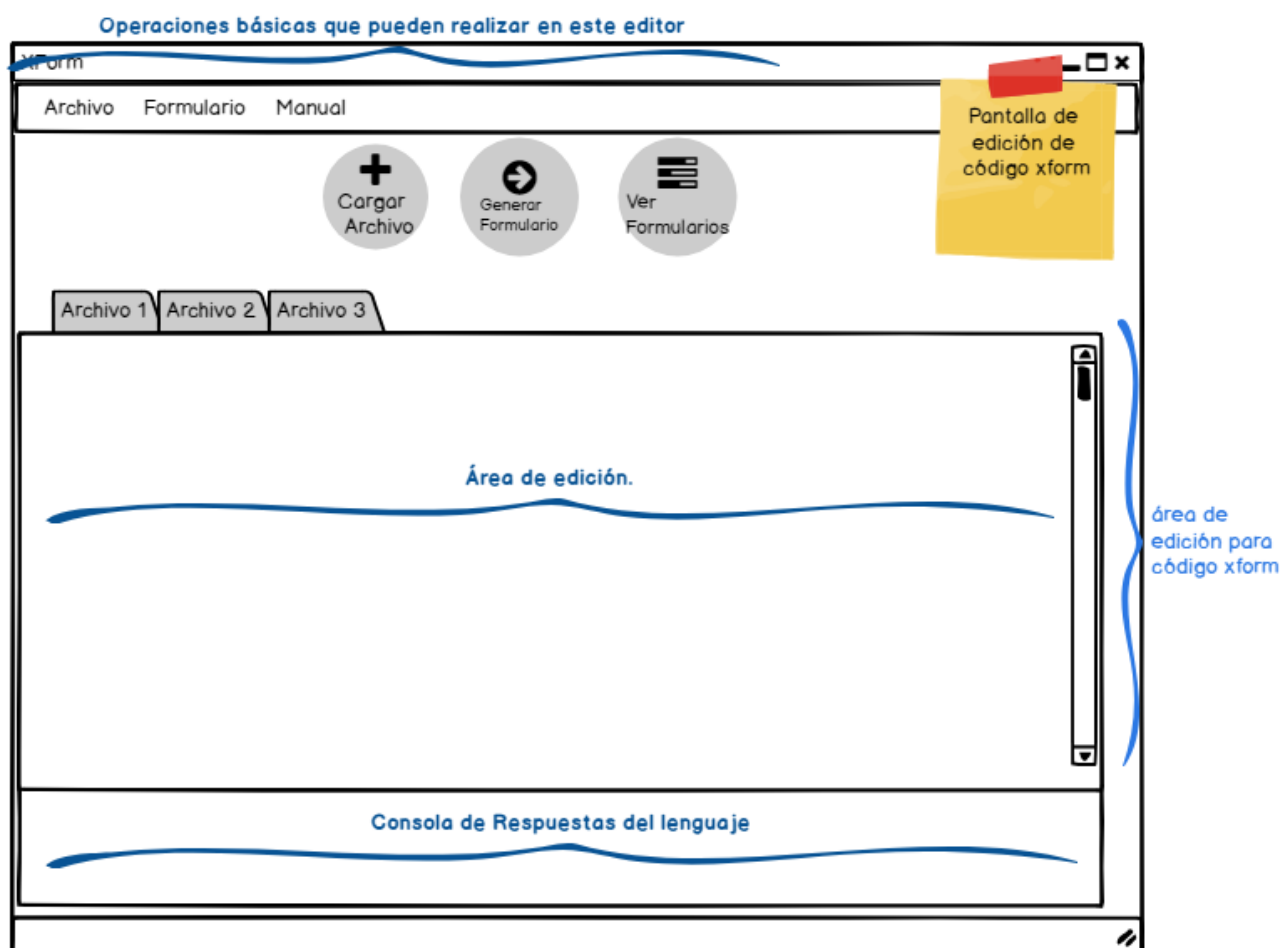
- Abrir: Funcionalidad que permitirá abrir un archivo del lenguaje XForm y lo mantendrá en memoria para un posterior análisis.
- Guardar: Permitirá guardar la información de la pestaña actual con el archivo compilado.
- Errores: Permitirá generar el reporte de errores detectados y deberá mostrarlos en formato HTML o deberá ser mostrado desde la aplicación en forma de tablas.

Botón Generar

Permitirá realizar la interpretación del archivo xls que fue abierto, el cual se tendrá en memoria y generará el archivo con formato Xform.

3.3 Componente para generación de formulario

Este componente servirá para analizar los archivos con formato XForm generados anteriormente y luego de analizarlos deberá mostrar la interfaz del formulario. La interfaz principal deberá ser compuesta por los siguientes componentes:



Barra de Herramientas:

- **Archivo**
 - Abrir: Funcionalidad que permitirá abrir un archivo principal del lenguaje XForm en una nueva pestaña.
 - Guardar: Permitirá guardar los cambios de la pestaña actual en el archivo.
 - Guardar Como: Permitirá crear un nuevo archivo con formato XForm, con la información que se encuentre en la pestaña actual.
- **Formulario**
 - Nueva Pestaña: Permitirá crear una nueva pestaña en el área de edición.
 - Cerrar Pestaña: Permitirá cerrar la pestaña actual del área de edición.
- **Manual**
 - Manual Técnico: Mostrará el manual técnico en formato PDF.
 - Manual de Usuario: Mostrará el manual de usuario en formato PDF.
 - Gramática Utilizada: Mostrará la gramática utilizada en el lenguaje XForm y del estándar de formularios, en formato PDF.

Botones de Funcionalidad Principal

- **Botón cargar archivo:** Este componente permitirá cargar un archivo XForm para su edición, deberá mostrar el código en una pestaña nueva.
- **Botón generar formulario:** Este componente permitirá generar un formulario en base al archivo que se esté analizando, el formulario deberá ser desplegado en una nueva ventana.
- **Botón ver Formularios:** Mostrará la carpeta principal donde se encuentren almacenados los formularios y se permitirá ver las respuestas de los formularios contestados.
- **Editor de texto:** Área de edición donde se permitirá la inserción o modificación de código del lenguaje XForm.
- **Consola de respuestas del lenguaje:** Área de impresión para poder interactuar con el lenguaje y no con el formulario de preguntas.

4 Estructura del Archivo XLS

El lenguaje del archivo XLS será definido como un estándar práctico para elaborar formularios, creado para simplificar la creación de formularios. El lenguaje es bastante simple y fácil de entender, y además también permitirá la creación de formularios complejos por parte de alguien familiarizado con la sintaxis que se describirá a continuación.

4.1 Características del lenguaje

Esta sección describirá las características que contiene el lenguaje estándar para elaborar formulario.

Case sensitive

El lenguaje es case insensitive, esto significa que no existirá diferencia entre mayúsculas y minúsculas, esto aplicará tanto para palabras reservadas propias del lenguaje como para identificadores. Por ejemplo, si se declara un identificador de nombre "Contador" este no será distinto a definir un identificador "contador".

Tipos de dato

Para este lenguaje se utilizará los tipos de dato entero, decimal, cadena, carácter y booleano.

A continuación, se definen los tipos de datos a utilizar:

- Entero: este tipo de dato, como su nombre lo indica, aceptará valores numéricos enteros.
- Decimal: este tipo de dato, como su nombre lo indica, aceptará valores numéricos decimales.
- Booleano: este tipo de dato aceptará valores de verdadero y falso que serán representados con palabras reservadas que serán sus respectivos nombres.
- Cadena: este tipo de dato aceptará cadenas de caracteres, que deberán ser encerradas dentro de comillas dobles ("caracteres") o comillas simples ('Carácter').

Signos de agrupación

Para dar orden y jerarquía a ciertas operaciones aritméticas se utilizarán los signos de agrupación. Para los signos de agrupación se utilizarán los paréntesis.

Ejemplo

$$55 * ((85 - 3) / 8)$$

Referencia hacia una variable en el Formulario

Se podrá hacer la llamada a una variable de las preguntas, para esto se utilizará un numeral y el identificador de la pregunta dentro de corchetes.

Ejemplo

#[id_pregunta]

4.2 Sintaxis para la elaboración de formularios

En este punto se definirá la sintaxis del lenguaje estándar para la elaboración de formularios y la manera correcta del funcionamiento en los archivos XLS.

4.2.1 Archivo XLS

El archivo XLS, es una extensión que corresponde al formato de archivos de Microsoft Excel (es una aplicación de hojas de cálculo que forma parte de la suite de oficina Microsoft Office.).

La idea principal de usar formato XLS, es poder desarrollar el archivo haciendo uso de la herramienta de Excel, por la facilidad del manejo de tablas y la creación de archivos. Cada libro de trabajo de Excel generalmente tiene dos hojas de trabajo:

- **Encuesta:** Esta hoja de trabajo le dará la estructura general y contendrá la mayor parte del contenido del formulario.
- **Opciones:** Esta hoja de trabajo se usará para especificar las opciones de respuestas para preguntas de opción múltiple.

Existirá una tercera hoja de trabajo opcional llamada **configuración**, en esta se podrá agregar especificaciones adicionales a un formulario.

Las hojas de trabajo “encuesta” y “opciones” tienen un conjunto de columnas obligatorias que deben estar presentes para que el formulario funcione. Además, cada hoja de trabajo contiene un conjunto de columnas opcionales que permitirá un mayor control sobre el comportamiento de cada entrada (fila) en el formulario, pero no son esenciales. Cada celda deberá contener valores para cada una de las columnas obligatorias, pero las columnas opcionales pueden dejarse en blanco.

Las columnas que serán agregadas a un archivo de Excel, sean obligatorias u opcionales, podrá aparecer en cualquier orden. Las columnas opcionales se podrán omitir por completo, es decir no hay necesidad de colocarlas. Cualquier cantidad de filas se podrá dejar en blanco. Se ignorará todo formato de archivo .xls, por ejemplo, se podrá dar algún formato a las celdas (ejemplo: color, negrilla, cursiva), sombreado y otro formato fuente al texto para que el formulario sea más legible.

A continuación, se describen las hojas de trabajo del archivo XLS.

4.2.2 Hoja de trabajo de Encuestas

Esta hoja de trabajo le dará a un formulario su estructura general y contendrá la mayor parte del contenido del formulario. Contendrá la lista completa de preguntas e información sobre cómo deberían aparecer en el formulario. Cada fila generalmente representa una pregunta; Sin embargo, hay ciertas otras características que se describirán a continuación que se podrá agregar al formulario para mejorar la experiencia del usuario. El nombre de la hoja del archivo se llamará “encuesta”.

El archivo XLS tiene tres columnas obligatorias: **tipo, idpregunta y etiqueta**.

- La columna **tipo** especifica el tipo de entrada que podrá ingresar el usuario.
- La columna de **idpregunta** especificará el nombre de la variable única para esa entrada, el nombre de la variable es un identificador. No existirá dos entradas que tengan el mismo nombre.
- La columna de **etiqueta** contiene el texto real que se verá en el formulario.

Las columnas descritas no contienen ningún orden en específico, es decir la columna tipo se puede definir de ultimo y la columna etiqueta de primero

Vista Tabla

Tipo	Idpregunta	Etiqueta
Fecha	Hoy	
Seleccionar_uno género	Genero	“¿El género del usuario?”
Entero	Años	“¿Edad del usuario?”

Podrán existir otro tipo de columnas opcionales las cuales son:

Columna	Nombre del Encabezado	Descripción
Sugerencia	Sugerir	Etiqueta de pregunta, esta se mostrará junto a la pregunta, servirá como una sugerencia hacia la misma.
Código Pre	Codigo_Pre	Esta columna permitirá agregar código en alto nivel al formulario, este será agregado antes de la asignación del valor a la pregunta.
Código Post	Codigo_Post	Esta columna permitirá agregar código en alto nivel al formulario, este será agregado después de la asignación del valor a la pregunta.
Restricciones	Restringir	Definirá los valores permitidos para una respuesta.
Mensaje de Restricción	RestringiMsn	Será el mensaje que se mostrará al usuario si la respuesta no fue valida.
Valor necesario	Requerido	Indicará que debe responder una pregunta para que el formulario continúe. Sus valores podrán ser “verdadero” o “falso” y 1 o 0.
Mensaje de valor necesario	requeridoMsn	Será el mensaje que deberá mostrar al momento de omitir una pregunta.
Valor predeterminado	Predeterminado	Será un valor predeterminado que se asignará previamente ante de que el usuario llegue a contestar la pregunta.
Condición aplicable	Aplicable	Característica que permitirá la posibilidad de omitir una pregunta o hacer que aparezca una pregunta adicional dependiendo de la respuesta de la pregunta anterior, para poder mostrar la pregunta, se deberá de cumplir la

		condición que se aplicará a la pregunta actual.
Solo de lectura	Lectura	Indicará si el valor de la respuesta de una pregunta podrá ser modificada. Sus valores permitidos pueden ser “verdadero” o “falso” y 1 o 0.
Cálculo	Cálculo	Será aplicable para realizar cálculos con las respuestas de otras preguntas o realizar cálculos en esa misma pregunta.
Número de repeticiones	Repetición	Se utilizará para el número de repeticiones de un grupo de preguntas de tipo ciclo. Se podrá colocar un número exacto o realizar una operación para asignar el número de repeticiones.
Multimedia	Multimedia	Estas columnas se usarán para definir un archivo multimedia de una pregunta. Los archivos multimedia valido son: imagen, audio y video.
Apariencia	Apariencia	Se utilizará para definir el tipo de entrada de información de la pregunta, cambiando su apariencia, por ejemplo: a una pregunta de tipo entero, se desea mostrar una caja de texto y no una caja de texto numérica. Esta no cambiará el tipo de dato de la pregunta.
Parámetros	Parámetro	Se utilizará para poder definir los parámetros de una pregunta si los tiene y si pueden ser aplicables, caso contrario son ignorados.

Para definir a un formulario se deberá describir su tipo de pregunta. Los tipos de pregunta que se podrán definir en la hoja de opciones serán los siguientes:

Tipo	Palabra reservada	Descripción
Texto	Texto	Este tipo de dato, como su nombre lo indica, aceptará texto. El tipo de respuesta es de texto libre. Permitirá el uso de los siguientes parámetros: Cad_max, cad_min y cad_fila.
Numero de tipo entero	Entero	Este tipo de dato, como su nombre lo indica, aceptará valores numéricos entero. El tipo de respuesta deberá ser un número entero.
Numero de tipo decimal	Decimal	Este tipo de dato, como su nombre lo indica, aceptará valores numéricos entero. El tipo de respuesta deberá ser un número con decimales.
Rango	Rango	Tipo de dato que recibirá de rango un número, entre un rango definido. Se deberá definir el inicio y final del rango. La respuesta de tipo rango, recibe un número

		de tipo entero. Se podrá definir los siguientes parámetros: iniciar y finalizar.
Condicional	Condicion	Este tipo de dato aceptará valores de verdadero y falso o sí y no. La respuesta solo aceptará un valor de tipo booleano. Aceptará como parámetro una opción, este se deberá elegir entre dos valores Si_No o V_F.
Fecha	Fecha	Este tipo de dato aceptará valores de tipo fecha. La respuesta de la fecha deberá de ser dd/mm/yy Donde: <ul style="list-style-type: none"> • dd: días • mm: mes • yy: año
Tiempo	Hora	Este tipo de dato aceptará valores de tipo tiempo. La respuesta de la hora deberá de ser hh:mi:ss o hh:mi. Donde: <ul style="list-style-type: none"> • hh: hora • mi: minutos • ss: segundos
Fecha y Hora	FechaHora	Este tipo de dato aceptará valores de tipo fecha y tiempo. La respuesta de la fecha y hora deberá de ser Dd/mm/yy hh:mm:ss o Dd/mm/yy hh:mm
Respuesta de tipo selección	Selección_uno Selección_múltiples	Tipo de dato con múltiples opciones, existirán dos tipos de selección múltiple Selecciona_uno (se podrá seleccionar una sola respuesta) y Selecciona_múltiple (se podrá seleccionar múltiples respuestas).
Nota	Nota	Pregunta que mostrará una nota o mensaje, este tipo de dato no recibirá ningún valor, es decir mostrará una nota en la pantalla, no tomará ninguna entrada, este tipo de nota podrá ser una caja de texto.
Multimedia	Fichero	Este tipo permitirá la opción de poder subir un archivo multimedia y asignarla como respuesta de una pregunta. El formato del archivo a subir es opcional, si el formato es especificado, se validará que no acepte otro tipo de formato.
Calcular	Calcular	Es una expresión de valores existentes que pueden ser utilizados por otras preguntas (sin la intervención del usuario).
Agrupación	Iniciar agrupación ...	Servirán para poder agrupar preguntas y tener un orden para las preguntas.

	Finalizar agrupación	
Ciclos	Iniciar ciclo ... Finalizar ciclo	Se utilizará para repetir un grupo de preguntas utilizando la sintaxis de inicio de ciclo y la sintaxis de fin del ciclo.

4.2.3 Hoja de Opciones

Esta hoja del archivo XLS se usará para especificar las opciones de las respuestas para las preguntas de opción múltiple. Cada fila representará una elección de respuesta. Las opciones de respuestas con el mismo nombre de la lista, se considerarán parte de un conjunto de opciones relacionadas y aparecerán juntas para una pregunta. Esto también permitirá reutilizar un conjunto de opciones para múltiples preguntas. El nombre de la hoja del archivo será “opciones”.

El archivo tipo opción tiene tres columnas obligatorias: **nombre_lista**, **nombre** y **etiqueta**.

- La columna del **nombre_lista** le permitirá agrupar un conjunto de opciones de respuesta relacionadas, es decir, las opciones de respuesta que deberían aparecer juntas bajo una pregunta.
- La columna de **nombre** especificará el nombre de la variable única para esa elección de respuesta.
- La columna de **etiqueta** mostrará la opción de respuesta exactamente como desea que aparezca en el formulario.
- La columna de **multimedia** se usará para definir un archivo multimedia de una pregunta. Los archivos multimedia valido son: imagen, audio y video. Esta columna tiene la misma funcionalidad que la columna multimedia de la hoja de encuesta.

Vista Tabla

Nombre_lista	nombre	Etiqueta
género	transgénero	Transgénero
género	hembra	Hembra
género	masculino	Masculino
género	otro	Otro

4.2.4 Hoja de Configuraciones

La hoja de configuración es opcional, pero permitirá personalizar aún más un formulario, agregando un tema de estilo general para el formulario, entre otros. Los encabezados de columna en la hoja de configuración son los siguientes:

- **Título_formulario**: el título del formulario que se muestra a los usuarios. El título del formulario se extrae de la columna “idForm”, si “Título_formulario” está en blanco o falta.

- **idForm:** es nombre utilizado para identificar el nombre del almacenamiento del formulario. El identificador del formulario se extrae del nombre del archivo XLS si la columna "idForm" está en blanco o falta
- **Estilo:** Servirá para configurar la vista del formulario o el estilo que se desea, los estilos posibles a utilizar se definirán en el siguiente punto.
- **Importar:** servirá para poder importar otros archivos XForm, al generar el código deberá importar el archivo.
- **Código_menu:** este agregará el código colocado en el lenguaje de alto nivel en el método principal.
- **Código_global:** este agregará el código colocado en el lenguaje de alto nivel, dentro de la clase en área global, se podrá agregar para poder declarar variables globales en el código generado.

Ejemplo

Titulo_formulario	idForm	Estilo	importar	codigo_principal	codigo_global
	impor_arbol		importar Arbol.xform		
	impor_nodo		importar Nodo.xform		
	codigo			abo1 = nuevo Arbol(); abo2 = nuevo Arbol();	Arbol abo1; Arbol abo2;
Arbol Binario	Abb	Grupo			

4.2.5 Estilos

Los estilos válidos para la vista de los formularios serán los siguientes.

Página

Deberá de mostrar las preguntas del formulario una pregunta por página, es decir solo se podrá contestar una pregunta a la vez, para poder contestar la pregunta siguiente se deberá pasar a la siguiente página, en la página deberá de existir un botón de siguiente pregunta y anterior.

Vista Tabla

Titulo_formulario	idForm	Estilo
Precio_Comida	pC	Pagina

Vista

Todo

Mostrará todas las preguntas en una sola página del formulario, se podrán contestar todas las preguntas a la vez. Se deberá de validar las preguntas ocultas o las que se contestan si se cumple una condición deberá de aparecer la pregunta una vez se haya cumplido dicha condición.

Vista Tabla

Titulo_formulario	idForm	Estilo
Precio_Comida	pC	Todo

Vista

Cuadrícula

Mostrará un estilo en forma de cuadros, este estilo permitirá a un formulario imitar el aspecto de las encuestas en papel tradicionales compactando múltiples preguntas en una fila. Este estilo se deberá de configurar en la columna de apariencia el número de celdas a ocupar por pregunta, el número mayor de la cantidad de celdas definirá las columnas. La sintaxis a utilizar en la columna apariencia será: C#numero, donde: C es el referente a celdas y # deberá ser un número, este definirá el número de celdas a ocupar la pregunta.

Vista Tabla de preguntas

tipo	IdPregunta	etiqueta	parametro	Apariencia
iniciar agrupación	AB	
...	A	C4
...	B	C1
...	C	C1
...	D	C1

Finalizar agrupación				
iniciar agrupación	DD	
...	E	C2
...	F	C2
Finalizar agrupación				

Vista Tabla de Configuración

Titulo_formulario	idForm	Estilo
Precio_Comida	pC	Cuadrícula

Vista

Formulario Completo

1.1) Nombre de la instalación.

1.1) Edad difunto.

3

Seleccionar unidad de edad.

☒ Meses ☐ Años
☐ Desconocidas

1.3) Sexo difunto.

Masculino

☐

Hembra

☐

2.1) Fecha de la lesión.

yyyy/mm/dd

2.1) Fecha de la muerte.

yyyy/mm/dd

4.2.6 Tipos de Columnas

El estándar para la elaboración de formularios admite una amplia variedad de columnas. La funcionalidad exacta se explicará a continuación, se describen y se ejemplifican cada tipo de columna a usar en la hoja de trabajo de encuestas:

Columna de Sugerencias (Sugerir)

Etiqueta de pregunta, esta se mostrará junto a la pregunta, servirá como una sugerencia hacia la misma.

Vista Tabla

tipo	IdPregunta	etiqueta	sugerir
texto	capital	Capital de Guatemala?	Inicia con la palabra Ciudad.

Salida

Formulario - Ejemplo

Capital de Guatemala?

Inicia con la palabra Ciudad.

◀

▶

Columnas Código Pre y Post

Estas columnas permitirán poder agregar código de alto nivel, este podrá ser agregado antes de mostrar la pregunta o después de mostrar la pregunta.

Vista Tabla

tipo	IdPregunta	etiqueta	Codigo_pre	Código_post
texto	capital	Capital de Guatemala?	Imprimir ("Consola saldrá Esto");	Imprimir ("Contesto la Capital " + @);

Columna de restricciones (restringir)

Definirá los valores permitidos para una respuesta. Para poder definir una restricción con la respuesta de dicha pregunta se deberá de colocar un punto antes de la condición o expresión a utilizar.

Vista tabla

tipo	IdPregunta	etiqueta	Restringir
entero	edad	Cuántos años tienes?	.<= 18

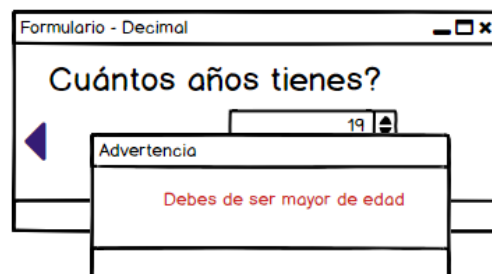
Columna de Mensaje de restricción (RestringirMsn)

Será el mensaje que se mostrará al usuario si la respuesta no fue valida.

Ejemplo vista tabla

tipo	IdPregunta	etiqueta	Restringir	RestringirMsn
entero	edad	Cuanto años tienes?	. <= 18 && .>10	Debes de ser mayor de edad

Salida



Columna valor necesario (requerido)

Indicará si se debe responder u omitir una pregunta, para poder continuar. Sus valores podrán ser verdadero o falso y 1 o 0. Si el valor requerido no es especificado deberá ser falso.

Ejemplo vista tabla

tipo	IdPregunta	etiqueta	restringir	requerido
entero	edad	¿Cuántos años tienes?	. <= 150	verdadero

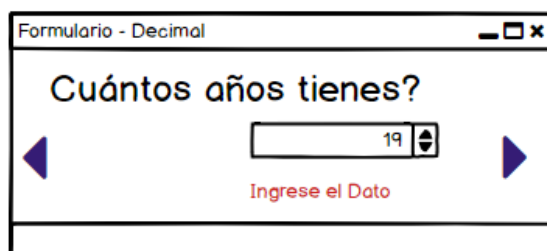
Columna Mensaje de valor necesario (requeridoMsn)

Será el mensaje que deberá mostrar al momento de omitir una pregunta, si no es asignado y la respuesta es necesaria deberá mostrar el mensaje por defecto “Conteste la pregunta”. El mensaje a mostrar deberá de ir entre comillas dobles.

Vista tabla

tipo	IdPregunta	etiqueta	requeridoMsn	requerido
entero	ages	¿Cuántos años tienes?	Ingrese el dato	verdadero

Salida



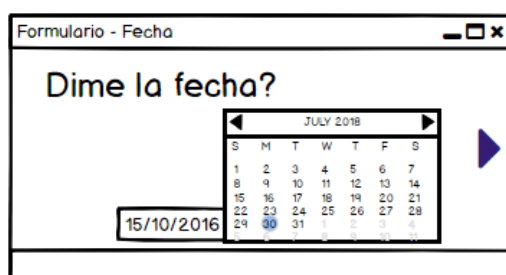
Columna valor por defecto (predeterminado)

Será un valor predeterminado que se asignará previamente ante de que el usuario llegue a contestar la pregunta.

Vista tabla

tipo	IdPregunta	etiqueta	pordefecto
FechaHora	hoy		
Fecha	Nueva_fecha	Preguntas fecha?	15/10/2016

Salida



Columna condición aplicable (Aplicable)

Una característica del estándar para la elaboración de formularios es la posibilidad de omitir una pregunta o hacer que aparezca una pregunta adicional dependiendo de la respuesta de la pregunta anterior, para poder mostrar la pregunta, se deberá de cumplir la condición que se aplicará a la pregunta actual. Esta característica se podrá aplicar en los grupos y ciclos.

Vista tabla

tipo	IdPregunta	etiqueta	aplicable
Texto	Nombre	Cuál es tu nombre?	

Texto	Nombre_comun	Tu nombre es común?	#[Nombre] = "Jose"
Texto	Nombre_no_comun	Tu nombre no es común?	

Columna solo lectura (Lectura)

Permitirá si el valor de la respuesta de una pregunta podrá ser modificada. Sus valores permitidos pueden ser "verdadero" o "falso" y 1 o 0.

Vista tabla

tipo	IdPregunta	etiqueta	pordefecto	lectura
entero	edad	Cuanto años tienes?	18	verdadero

Salida

Columna de cálculo (Calculo)

Es aplicable para realizar calculo con las respuestas de otras preguntas o realizar cálculos en esa misma pregunta. Una expresión matemática a la que se puede hacer referencia en otra parte.

Vista tabla

tipo	IdPregunta	etiqueta	cálculo
decimal	cantidad	¿Cuál fue el precio de la comida?	
calcular	propina		#[cantidad] * 0.18
Nota	monitor	18% de propina para su comida es: #[propina]	

Columna número de repeticiones (repetición)

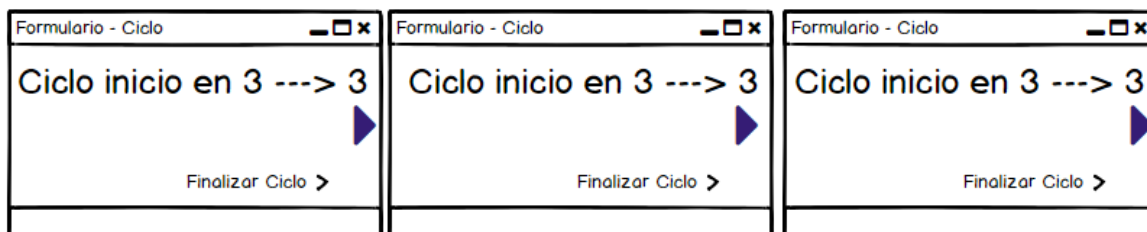
Se utilizará para el numero de repeticiones para un grupo de preguntas. Se podrá colocar un número exacto o realizar una operación para asignar el número de repeticiones.

Vista tabla

tipo	IdPregunta	etiqueta	Repetición
entero	cantidad	¿Cuántas repeticiones desea?	
Iniciar Repetición	Valor		#[cantidad]
Nota	Actual	El valor actual es---> #[valor]	

Finalizar repetición			
-------------------------	--	--	--

Salida



Columna de multimedia (Multimedia)

Estas columnas se usarán para definir un archivo multimedia de una pregunta, si el archivo multimedia se encuentra en la misma carpeta del archivo se definirá ruta relativa, caso contrario se definirá con la ruta completa.

Los tipos de multimedia son:

Media_imagen: Mostrará una imagen junto a una pregunta. La imagen deberá de estar en la misma carpeta donde se encuentre el archivo XForm.

Media_audio: Mostrará un audio junto a una pregunta. El audio deberá de estar en la misma carpeta donde se encuentre el archivo XForm. Se podrá especificar si se desea reproducir al momento de mostrar la pregunta.

Media_video: Mostrará un video junto a una pregunta. El video deberá de estar en la misma carpeta donde se encuentre el archivo XForm. Se podrá especificar si se desea reproducir al momento de mostrar la pregunta.

Sintaxis

```
Media_image "/ruta/relativa/Imagen.jpg"
Media_audio "C://ruta/no/relativa/audio.mp3"
Media_video "/ruta/relativa/video.mp4"
```

Vista tabla

tipo	IdPregunta	etiqueta	multimedia
Nota	Imagen	Mira la imagen	Media_imagen "/ruta/imagen.png"
Nota	video	Mira el Video	Media_video reproduccion = true "/ruta/video.png"
Nota	audio	Escucha el Audio	Media_audio "/ruta/imagen.png" reproduccion = true
Nota	audio2	Escucha el Audio	Media_audio "/ruta/imagen.png"

Salida



Columna de apariencia (apariencia)

Se podrá definir el tipo de entrada de información de la pregunta, cambiando su apariencia, por ejemplo: a una pregunta de tipo entero, se desea mostrar una caja de texto y no una caja de texto numérica. Esta no cambiará el tipo de dato de la pregunta. Los tipos de entrada validos son los mismos tipos de dato de las preguntas.

Ejemplo vista tabla

tipo	IdPregunta	etiqueta	Apariencia
entero	edad	Cuanto años tienes?	Cadena
cadena	valor	Cuanto años tienes?	Entero
cadena	anio	Qué año es?	C4

Salida

Columna de parámetros (Parámetro)

Se utilizará para poder definir los parámetros de una pregunta si los tiene y si pueden se aplicables, caso contrario son ignorados.

Vista Tabla

tipo	idPregunta	etiqueta	parámetro
rango	cuantos	Cuantos hijos tienes	Iniciar=0 finalizar=10

4.2.7 Tipos de Pregunta

El estándar para la elaboración de formularios admite una amplia variedad de tipos de preguntas. La funcionalidad exacta y el estilo de visualización de cada pregunta se especificarán más adelante. El tipo de pregunta se asignará en las celdas de la columna “tipo”.

A continuación, se describen y ejemplifican cada tipo de pregunta a usar en la hoja de trabajo de encuestas:

Tipo Texto (Texto)

Este tipo de dato, como su nombre lo indica, aceptará texto. El tipo de respuesta es de texto libre. En este tipo de pregunta se podrán definir los siguientes parámetros:

- Cad_min: Definirá el número mínimo de caracteres que puede contener la respuesta de una pregunta.
- Cad_max: Definirá el número máximo de caracteres que puede contener la respuesta de una pregunta.
- Cad_fila: Especificará el número mínimo de filas (será la cantidad de saltos de línea) que una puede obtener en un campo de cadena.

Los parámetros pueden ser opcionales.

Vista Tabla

tipo	IdPregunta	etiqueta	parametro
texto	capital	Capital de Guatemala?	cad_max = 20 cad_fila = 10
texto	Capital2	Capital de Guatemala?	cad_max = 20

Salida

Tipo Entero (Entero)

Este tipo de dato, como su nombre lo indica, aceptará valores numéricos entero. El tipo de respuesta debe ser un número entero.

Vista Tabla

tipo	IdPregunta	etiqueta
Entero	cantidad	Cuantos países conoces?

Salida

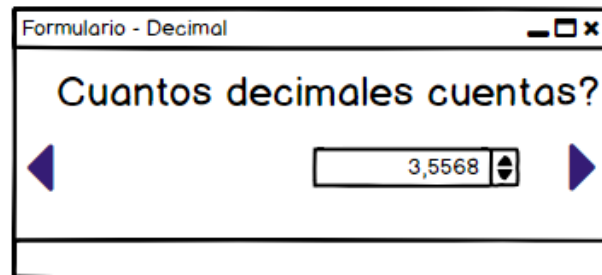
Tipo Decimal (Decimal)

Este tipo de dato, como su nombre lo indica, aceptara valores numéricos con decimales. El tipo de respuesta deberá ser un número con decimales.

Vista Tabla

tipo	IdPregunta	etiqueta
decimal	cantidad	Cuantos decimales cuentas?

Salida



Tipo Rango (Rango)

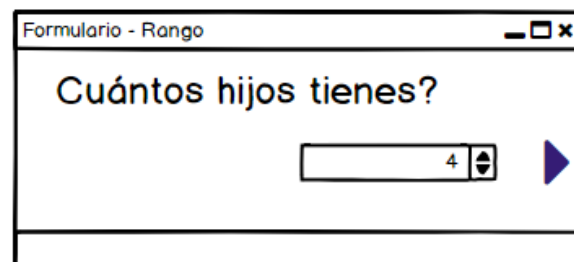
Tipo de dato que recibirá de rango un número, entre un rango definido. Se deberá definir el inicio y final del rango. La respuesta de tipo rango, recibe un número de tipo entero. En este tipo de pregunta se podrán definir los siguientes parámetros:

- Iniciar: como el inicio del rango, si el parámetro no se define iniciará desde 0.
- Finalizar: como el final del rango. Parámetro obligatorio

Ejemplo Vista Tabla

tipo	idPregunta	etiqueta	parametro
rango	cuantos	Cuanto hijos tienes	finalizar=10

Salida



Tipo Condicional (Condición)

Este tipo de dato aceptará valores de verdadero y falso o sí y no. La respuesta solo aceptará un valor de tipo booleano.

En este tipo de pregunta se podrán definir los siguientes parámetros:

- Si_No: mostrará la opción de seleccionar entre los valores de Si o No, como respuesta.

- V_F: mostrará la opción de seleccionar entre los valores de verdadero o falso, como respuesta.

Si no se definen ningún parámetro tomara "V_F" como valor por defecto.

Ejemplo Vista Tabla

tipo	idPregunta	etiqueta	parametro
condicion	mayor	Eres mayor de edad?	Opción=V_F

Salida

Tipo Fecha (Fecha)

Este tipo de dato aceptará valores de tipo fecha. La respuesta de la fecha deberá de ser dd/mm/yy

Donde:

- dd: días
- mm: mes
- yy: año

Ejemplo Vista Tabla

tipo	idPregunta	etiqueta
fecha	mayor	Dime la fecha?

Salida

Tipo Tiempo (Hora)

Este tipo de dato aceptará valores de tipo tiempo. La respuesta de la hora deberá de ser hh:mi:ss o hh:mi.

Donde:

- hh: hora

- mi: minutos
- ss: segundos

Ejemplo Vista Tabla

tipo	idPregunta	etiqueta
hora	mayor	Dime la hora?

Salida

Tipo Fecha y Hora (FechaHora)

Este tipo de dato aceptará valores de tipo fecha y tiempo. La respuesta de la fecha y hora deberá de ser Dd/mm/yy hh:mm:ss o Dd/mm/yy hh:mm

Donde:

- dd: días
- mm: mes
- yy: año
- hh: hora
- mi: minutos
- ss: segundos

Ejemplo Vista Tabla

tipo	idPregunta	etiqueta
hora	mayor	Dime la fecha y hora?

Salida

Tipo Respuesta Selección

Tipo de dato con múltiples opciones, existirán dos tipos de selección múltiple **Selecciona_uno** (se podrá seleccionar una sola respuesta) y **Selecciona_múltiple** (se podrá seleccionar múltiples respuestas).

Escribir una pregunta de opción múltiple requerirá del libro de archivo de XLS llamado **opciones**, seguido del tipo de respuesta de selección múltiple se deberá de especificar el nombre de la lista del archivo de opciones.

Sintaxis

Selecciona_uno lista_nombre

Selecciona_multiples lista_nombre

Vista Tabla

tipo	idPregunta	etiqueta	Parámetro
Selecciona_uno generos	Genero	Dime un género?	
Selecciona_multiples general	numeros	Elige los numeros	

Vista Tabla Listas (Libro de Opciones)

Nombre_lista	nombre	Etiqueta	multimedia
fechas	mes	Meses	
fechas	anio	Año	
fechas	otro	Desconocido	
fruta	man	Manzana	/ruta/relativa/a.jpg
fruta	pe	Pera	/ruta/relativa/b.jpg
fruta	ban	Banano	/ruta/relativa/c.jpg

Salida Selección uno

Pregunta

1) Selecciona una unidad:

☐ Meses ☐ Años


☐ Desconocidas

Aceptar

Salida Selección Múltiple

Pregunta

1) Frutas Favoritas?

Manzana ☐ 

Pera ☐ 

Banano ☐ 

Aceptar

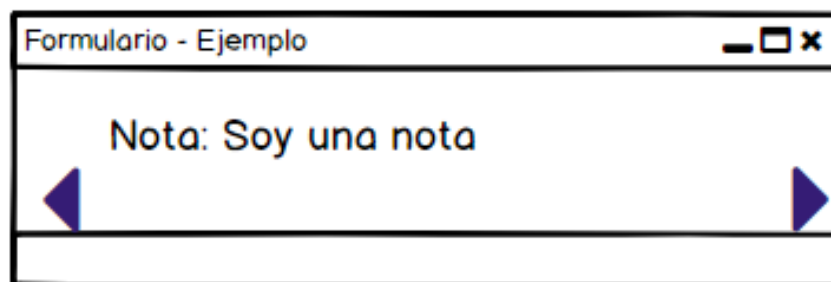
Tipo Nota (Nota)

Pregunta que mostrará una nota o mensaje, este tipo de dato no recibirá ningún valor, es decir mostrará una nota en la pantalla, no toma entrada, este tipo de nota puede ser una caja de texto.

Vista tabla

tipo	IdPregunta	etiqueta	cálculo
Nota	Mi_nota	Soy una Nota	

Salida



Tipo Multimedia

Este tipo permitirá la opción de poder subir un archivo multimedia y asignarla como respuesta a una pregunta.

- Fichero: Subirá y almacenará un archivo. El formato del archivo a subir es opcional, si el formato es especificado, se validará que no acepte otro tipo de formato. También se podrá definir uno o varios formatos en específico.

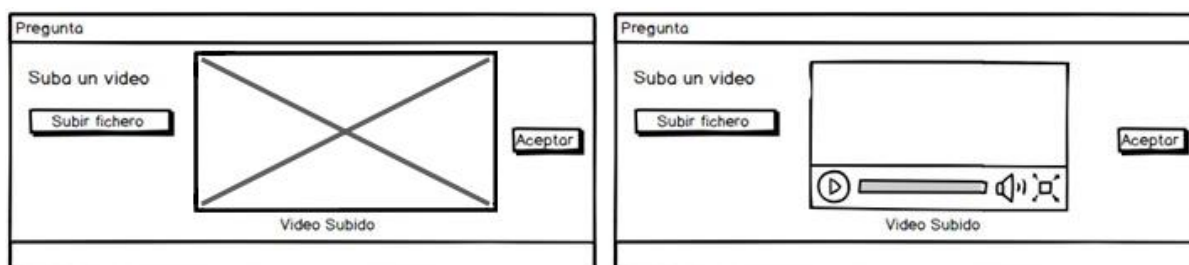
Sintaxis

Fichero “.txt, .csv”
Fichero “.doc”
\$\$si no se especifica se deberá tomar como un archivo de texto
Fichero

Vista tabla

tipo	IdPregunta	etiqueta
Fichero “.png, .jpg, .gif”	Mi_imagen	Sube una Imagen
Fichero “.mp3”	Mi_Audio	Sube un audio
Fichero	Mi_fichero	Sube cualquier archivo

Salida



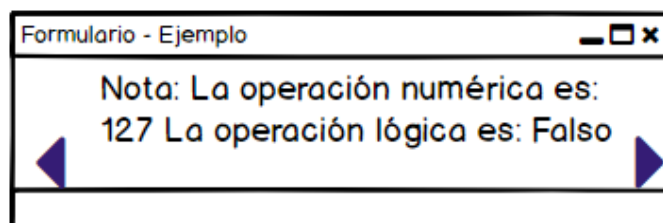
Tipo Cálculo (Calcular)

Es una expresión de valores existentes que pueden ser utilizados por otras preguntas (sin la intervención del usuario). Creará una variable de instancia sin elemento en el cuerpo del archivo XForm, este se establecerá mediante una expresión en la ruta, se especificará en la columna de cálculo.

Vista tabla

tipo	IdPregunta	etiqueta	cálculo
calcular	numeros		$150 * 0.18 + 100$
calcular	logicos		Verdadero && Falso && !(falso)
Nota	Ver_resultado	La operación numérica es: #[numeros] La operación lógica es: #[lógicos]	

Salida



4.2.8 Agrupaciones de Preguntas

Servirá para poder agrupar preguntas y tener un orden para las preguntas. Los grupos de preguntas se podrá aplicar valores en la columna “aplicable”, esto servirá para verificar si el grupo de preguntas deberán ser contestadas.

Dentro de las agrupaciones podrán existir otras agrupaciones, ciclos, ciclos anidados o cualquier tipo de preguntas. Deberá de mostrar todas las preguntas de un grupo del formulario por página, es decir solo se podrá contestar las preguntas de un grupo, para poder contestar el siguiente grupo de preguntas se deberá pasar a la siguiente página, en la página deberá de existir un botón de siguiente grupo y anterior. Al definir un grupo se deberá dar un identificador y para finalizar el grupo se deberá especificar con el identificador el nombre del grupo a finalizar.

Sintaxis

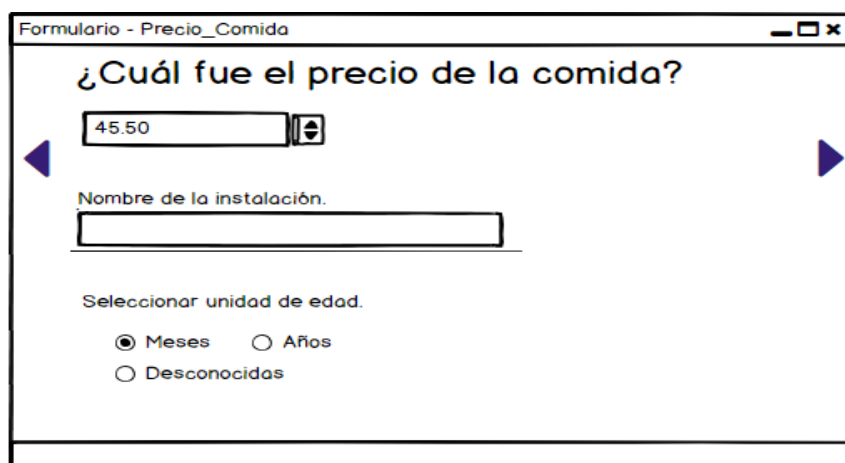
Iniciar agrupación

...
Finalizar agrupación

Vista tabla

tipo	IdPregunta	etiqueta	cálculo
Iniciar agrupación	operarGrupo		
Entero	Precio	¿Cuál fue el precio de la comida?	
Texto	Nombre	Nombre de la instalación	
Seleccionar_uno unidad	Tipo_fecha	Seleccionar unidad de edad	
Finalizar agrupación	operarGrupo		

Salida



Agrupación anidada de Preguntas

Son agrupaciones dentro otras agrupaciones de preguntas.

Vista tabla

tipo	IdPregunta	etiqueta	repetir
Iniciar agrupación	operarGrupo		
Entero	Cuántas	¿Cuántas veces repite?	
Iniciar agrupación	grupo_1		
Entero	Precio	¿Cuál fue el precio de la comida?	
Texto	Nombre	Nombre de la instalación	
Seleccionar_uno unidad	Tipo_fecha	Seleccionar unidad de edad	
Iniciar agrupación	Grupo_2		
Entero	Precio	¿Cuál fue el precio de la comida?	

Texto	Nombre	Nombre de la instalación	
Seleccionar_uno unidad	Tipo_fecha	Seleccionar unidad de edad	
Finalizar agrupación	Grupo_2		
Finalizar agrupación	grupo_1		
Finalizar agrupación	operarGrupo		

Agrupación de tipo Ciclo

Un usuario podrá repetir un grupo de preguntas utilizando la sintaxis de inicio de ciclo y la sintaxis de fin del ciclo. En un ciclo se podrá asignar la cantidad de veces a repetir el grupo de preguntas o dejar que el usuario finalice el grupo de preguntas mostrando un botón para finalizar el ciclo, este botón solo se mostrara cuando no se asigna la cantidad de veces a repetir.

Dentro de los ciclos podrán existir otras agrupaciones, ciclos, ciclos anidados o cualquier tipo de preguntas. Al definir un ciclo se deberá dar un identificador y para finalizar el ciclo se deberá especificar con el identificador el nombre del ciclo a finalizar.

Sintaxis

Iniciar ciclo
...
Finalizar ciclo

Vista tabla

tipo	IdPregunta	etiqueta	repetir
Iniciar agrupación	operarGrupo		
Entero	Cuantas	¿Cuántas veces repite?	
Iniciar ciclo	Ciclo_1		#[cuantas]
Entero	Precio	¿Cuál fue el precio de la comida?	
Texto	Nombre	Nombre de la instalación	
Seleccionar_uno unidad	Tipo_fecha	Seleccionar unidad de edad	
Finalizar ciclo	Ciclo_1		
Finalizar agrupación	operarGrupo		
Iniciar ciclo	Ciclo_Infinito		
Entero	Precio	¿Cuál fue el precio de la comida?	

Texto	Nombre	Nombre de la instalación	
Seleccionar_uno unidad	Tipo_fecha	Seleccionar unidad de edad	
Finalizar agrupación	Ciclo_infinito		

4.2.9 Operadores y funciones del formulario

Las expresiones utilizadas en las columnas calculo, restricciones y aplicable podrán contener operadores y funciones.

Operadores Matemáticos

	Explicación	Ejemplo
+	suma	#[salario] + #[entradas]
-	resta	#[ingreso] - [gastos]
*	multiplicación	#[bill] * 1.18
div	división	#[porcentaje] div 100
mod	módulo (residuo de la división)	(#[numero] mod 2) = 0

Los operadores matemáticos solo trabajarán con números.

- El operador de adición se puede usar para concatenar cadenas, solo en la columna de cálculo.
- Los valores vacíos (es decir, las variables que hacen referencia a preguntas y no contienen respuesta) son cadenas vacías (Nulos).

Operadores de comparación

Los operadores de comparación se usarán para comparar valores. El resultado de una comparación siempre será 'verdadero' o 'Falso'.

	Explicación	Ejemplo	Nota
=	igual a	#[enrolled] = "sí"	Se podrá comparar números o cadenas.
!=	diferente	#[enrolled] != "no"	Se podrá comparar números o cadenas.
>	Mayor que	#[edad] > 17	
>=	Mayor que o igual	#[age] >= 18	
<	menor que	#[age] < 65	
<=	Menor que o igual	#[age] <= 64	

Operadores booleanos

	Explicación	Ejemplo
&&	Operador and	#[edad] > -1 && #[edad] < 120
	Operador Or	#[edad] < 19 #[edad] > 64
!	Operador Not	!([#(edad) < 19])

Operadores de ruta

Explicación	Ejemplo	Notas
.	valor actual de la pregunta. Se deberá colocar haciendo referencia hacia su misma pregunta.	. > = 18 Usado en la columna de restricciones, calculo y aplicable.
..	Grupo o ciclo padre de la pregunta actual	posicion(..) Haciendo uso de la función posicion() para obtener el índice de iteración.
@	Se colocará haciendo referencia hacia su misma pregunta, es decir su valor actual de la pregunta.	Abb.insertar(@); Este solo se podrá usar en las columnas Codigo_pre y código_posr

Funciones

El lenguaje estándar para elaborar formularios podrá hacer uso de todas las funciones nativas descritas en el lenguaje de alto nivel, estas podrán ser utilizadas en las columnas calculo, restricciones y aplicable.

4.2.10 Estilos a los textos

El estándar para la elaboración de formularios permitirá dar estilos a las columnas de la hoja de preguntas de tipo etiqueta, RestringirMsn, sugerir y a las columnas de hoja de opciones a la etiqueta.

Sintaxis

```
@{estilos1:parámetro, estilos2, ..... }:{ Texto }
```

Los estilos aplicables son:

Estilo	Descripción	Ejemplo
negrilla	Colocará el texto en negrilla	\$\$ el texto "Guatemala?" estará en negrilla Capital de @{negrilla}: {Guatemala?}@
Color: nombre_color	Dará color al texto, los colores a definir serán dado por el código de color hexadecimal.	\$\$ el texto "Guatemala?" estará de color amarillo Capital de @{color: #FFFF00 }: {Guatemala?}@
Tam: #numero	Definirá el tamaño del texto.	\$\$ el texto "Guatemala?" estará de color amarillo y tamaño 20 Capital de @{color: #FFFF00 , tam:20 }: {Guatemala?}@
Cursiva	Colocará al texto en cursiva.	\$\$ el texto "Guatemala?" estará en negrilla Capital de @{cursiva}: {Guatemala?} @

subrayado	Colocará subrayado en el texto.	\$\$ el texto "Guatemala?" estará en negrilla Capital de @{subrayado}: {Guatemala?} @
-----------	---------------------------------	--

5 Estructura del Lenguaje Alto Nivel

Este formato representará a un lenguaje de alto nivel orientado a objetos y llevará por nombre XForm, lo cual permitirá poder pasar del código generado anteriormente a una forma visual y agradable al usuario también poder proporcionar una mejor interacción entre las preguntas y sus atributos. El lenguaje de alto nivel, permitirá poder relacionar de una forma más fácil, las preguntas con sus respectivas respuestas. La representación de un formulario tendrá una o más preguntas y una pregunta podrá contener una o más respuestas, las cuales serán interpretadas como atributos de un objeto.

5.1 Notación dentro del enunciado

Dentro del enunciado, se utilizarán las siguientes notaciones cuando se haga referencia al código de alto nivel.

Sintaxis y ejemplos

Para definir la sintaxis y ejemplos de sentencias de código de alto nivel se utilizará un rectángulo gris.



Asignación de colores

Dentro del enunciado y en el editor de texto de la aplicación, para el código de alto nivel, seguirá el formato de colores establecido en la siguiente tabla. Estos colores también deberán de ser implementados en el editor de texto.

Color	Token
Azul	Palabras reservadas
Naranja	Cadenas, caracteres
Morado	Números
Gris	Comentario
Negro	Otro

Palabras reservadas

Son palabras que no podrán ser utilizadas como identificadores ya que representan una instrucción específica y necesaria dentro del lenguaje.

Expresiones

Cuando se haga referencia a una 'expresión', se hará referencia a cualquier sentencia que devuelve un valor, ya sea una operación aritmética, una operación relacional, una operación lógica, un atributo, variable, parámetro, función, objeto o matriz.

Los tipos de datos para cada expresión vendrán dados por los sistemas de tipos definidos para las operaciones y del tipo de datos asociados al resto de elementos del lenguaje.

5.2 Características del lenguaje

Esta sección describirá las características que contiene el lenguaje de alto nivel.

5.3 Case Insensitive

El lenguaje es case insensitive, esto significa que no existirá diferencia entre mayúsculas y minúsculas, esto aplicará tanto para palabras reservadas propias del lenguaje como para identificadores. Por ejemplo, si se declara un identificador de nombre "Contador" este no será distinto a definir un identificador "contador".

5.4 Polimorfismo

Los procedimientos y funciones deberán tener la propiedad de polimorfismo. Esto quiere decir que podrá haber una o varias funciones o procedimientos con el mismo nombre, pero que tengan diferente tipo de retorno o diferentes parámetros.

5.5 Recursividad

Los procedimientos y funciones deberán soportar llamadas recursivas.

5.6 Tipos de datos

La declaración de tipos de variables será explícita en el lenguaje alto nivel, los tipos de variable soportados serán los descritos en la siguiente tabla.

Tipo	Descripción	Formato
Cadena	Las variables que se le asigne los valores de tipos cadena, deberán ir entre comillas dobles " ".	<code>cadena mivariable = "Hola mundo";</code>
Booleano	Son valores de verdadero y falso, las palabras reservadas para estos serán 'verdadero' y 'falso'	<code>booleano mivariable = verdadero;</code> <code>booleano mivariable = falso;</code>
Entero	Estas son los valores numéricos que puede tomar las variables, solo pueden ser enteros.	<code>entero mivariable : 100;</code> <code>entero mivariable : 90.90;</code>
decimal	Estas son los valores numéricos que puede tomar las variables, solo pueden ser enteros con decimales.	<code>decimal mivariable = 100.01;</code> <code>decimal mivariable = 90.90;</code>

hora	Estas deberán ir dentro de ‘ ‘ o “ ” para realizar la asignación de una fecha. Estos se ingresarán con el formato de tiempo llevará como hh:mm:ss en donde hh denota horas la cual será en formato de 24 horas, mm denota minutos y ss denota segundo.	fechaHora miHora = '23:59:59 '
fecha	Estas deberán ir dentro de ‘ ‘ o “ ” para realizar la asignación de una fecha. Estos se ingresarán con el formato dd/mm/yyyy. Donde dd denota el día, mm denota el mes y yyyy denota el año.	fecha miFecha = '01/01/2000';
fechaHora	Estas deberán ir dentro de ‘ ‘ para realizar la asignación de una fecha. Estos se ingresarán con el formato dd/mm/yyyy. Donde dd denota el día, mm denota el mes y yyyy denota el año. El formato de tiempo llevará como hh:mm:ss en donde hh denota horas la cual será en formato de 24 horas, mm denota minutos y ss denota segundo.	fechaHora miFecha = '31/12/1999 23:59:59 '
Respuesta	Este tipo de dato se utilizará para obtener las respuestas de los formularios y únicamente podrá ser usado dentro de la plantilla Formulario. Al momento de realizar la obtención de los datos se deberá de especificar el tipo de dato a obtener y solo podrá ser utilizado como parámetro de las Plantillas de Respuestas y Grupos.	Respuesta miRes ; miRes. Cadena miRes.Entero miRes.Decimal miRes.Fecha miRes.Hora miRes.Booleano

Cadenas

La cadena a mostrar en la pregunta podrá tener un estilo definido, es decir se podrá dar un color, tamaño diferente, negrilla, entre otros. Esto solo funcionará con las cadenas que pertenezcan dentro de “Formulario”, “Grupos” o “Preguntas”

Sintaxis

```
@{estilos1:parámetro, estilos2, ..... }:{ Texto }
```

Los estilos aplicables son:

Estilo	Descripción	Ejemplo
negrilla	Colocará el texto en negrilla	\$\$ el texto “Guatemala?” estará en negrilla Cadena etiqueta = “Capital de @{negrilla}: {Guatemala?}@”;

Color: nombre_color	Dará color al texto, los colores a definir serán dado por el código de color hexadecimal.	\$\$ el texto "Guatemala?" estará de color amarillo Cadena etiqueta = "Capital de @ {color: #FFFF00 } : { Guatemala?} @";
Tam: #numero	Definirá el tamaño del texto.	\$\$ el texto "Guatemala?" estará de color amarillo y tamaño 20 Cadena etiqueta = "Capital de @ {color: #FFFF00 , tam:20 } : { Guatemala?} @";
Cursiva	Colocará al texto en cursiva.	\$\$ el texto "Guatemala?" estará en negrilla Cadena etiqueta = "Capital de @ {cursiva}: {Guatemala?} @";
subrayado	Colocará subrayado en el texto.	\$\$ el texto "Guatemala?" estará en negrilla Cadena etiqueta = "Capital de @ {subrayado}: {Guatemala?} @";

5.7 Signos de agrupación

Para dar orden y jerarquía a ciertas operaciones aritméticas se utilizarán los signos de agrupación. Para los signos de agrupación se utilizarán los paréntesis.

Ejemplo

```
55 * ((85 - 3) / 8)
```

5.8 Almacenamiento de las respuestas

Las respuestas obtenidas del usuario deberán ser almacenadas en archivos físicos, por cada formulario contestado existirá una carpeta, que contendrá toda la información ingresada por el usuario, todas las carpetas del formulario contestadas deberán estar dentro de la carpeta principal del formulario, la estructura de los archivos de respuestas queda a discreción del estudiante. No es permitido el uso de bases de datos.

5.9 Sintaxis del lenguaje

En este punto se definirá la sintaxis del lenguaje de alto nivel y la manera correcta de funcionamiento.

5.9.1 Comentarios

Existirán dos tipos de comentarios. Los comentarios de una línea que empezarán con los símbolos "\$\$" y los comentarios con múltiples líneas que empezarán con los símbolos "\$#" y terminarán con los símbolos "#\$".

Ejemplo

```
$$este es un ejemplo de un lenguaje de una sola línea
$#
este es un ejemplo de un lenguaje de múltiples líneas.
# $
```

5.9.2 Operaciones Aritméticas

Una operación aritmética es un conjunto de reglas que permiten obtener otras cantidades o expresiones a partir de datos específicos. A continuación, se definen las operaciones aritméticas soportadas por el lenguaje.

5.9.3 Operadores

Las operaciones que soportará el lenguaje, se encuentran en la tabla siguiente.

Operador	Nombre	Función	Ejemplo
+	Suma	Operación aritmética que consiste en sumar varias cantidades en una sola.	5+5+10+6+5 10.5+10.5 95+0.5+1.5 +105
-	Resta	Operación aritmética que consiste en restar o quitar cantidades y obtener la diferencia	100-0.5-95 50-1.5-5
*	Multiplicación	Operación aritmética que consiste en multiplicar es decir sumar las cantidades las veces indicadas a otro número.	10*15*10 30*0.5*2 -105
/	División	Operación aritmética que devolverá el resultado de una división.	10/2 200/5
^	Potencia	Operación aritmética de potencia, toma la primera expresión como base y la segunda como exponente.	a : b^2;
%	Modulo	Operación aritmética que devolver el residuo de una división.	a : b%2;
++	Adición	Suma 1 a la expresión.	a++;
--	Sustracción	Resta 1 a la expresión.	(b+2)--

Los operadores aritméticos deberán de cumplir con las reglas de operatoria que se encuentran en la tabla siguiente.

Suma(+)	Booleano	Numérico	Cadena	Fecha/ Hora/ FechaHora
Booleano	OR	Sumar	Concatenar	Error
Numérico	Sumar	Sumar	Concatenar	Error
Cadena	Concatenar	Concatenar	Concatenar	Error
Fecha/ Hora/ FechaHora	Error	Error	Concatenar	Error

Resta(-)	Booleano	Numérico	Cadena	Fecha/ Hora/ FechaHora
Booleano	Error	Restar	Error	Error
Numérico	Restar	Restar	Error	Error
Cadena	Error	Error	Error	Error
Fecha/ Hora/ FechaHora	Error	Error	Error	Error
Multiplicación(*)	Booleano	Numérico	Cadena	Fecha/ Hora/ FechaHora
Booleano	AND	Multiplicar	Error	Error
Numérico	Multiplicar	Multiplicar	Error	Error
Cadena	Error	Error	Error	Error
Fecha/ Hora/ FechaHora	Error	Error	Error	Error
División (/)	Booleano	Numérico	Cadena	Fecha/ Hora/ FechaHora
Booleano	Error	Dividir	Error	Error
Numérico	Dividir	Dividir	Error	Error
Cadena	Error	Error	Error	Error
Date/Datetime	Error	Error	Error	Error
Potencia(^)	Booleano	Numérico	Cadena	Fecha/ Hora/ FechaHora
Booleano	Error	Elevar	Error	Error
Numérico	Elevar	Elevar	Error	Error
Cadena	Error	Error	Error	Error
Fecha/ Hora/ FechaHora	Error	Error	Error	Error
Modulo(%)	Booleano	Numérico	Cadena	Fecha/ Hora/ FechaHora
Booleano	Error	Error	Error	Error
Numérico	Resto de división	Resto de división	Error	Error
Cadena	Error	Error	Error	Error
Fecha/ Hora/ FechaHora	Error	Error	Error	Error

5.9.4 Condiciones

Las condiciones son operaciones que servirán para poder determinar la veracidad o falsedad de la misma. Para ello se hace uso de operaciones aritméticas, relacionales y lógicas.

5.9.5 Expresiones relacionales

Las operaciones relacionales servirán para comparar dos expresiones, el sistema deberá de manejar las operaciones de igualdad, diferente, menor, mayor, menor o igual y mayor o igual.

Especificaciones sobre las operaciones relacionales:

- Será válido comparar datos numéricos (`entero`, `decimal`) entre sí, la comparación se realizará utilizando el valor numérico con signo de cada dato.
- Será válido comparar datos numéricos con datos de tipo carácter en este caso se hará la comparación del valor numérico con signo del primero con el valor del código ascii del segundo.
- No será válido comparar valores lógicos (`booleano`) entre sí.

En la tabla siguiente se describe el uso de las expresiones relacionales.

Nombre	Símbolo	Descripción
Igual	==	Esta operación comprobará si dos expresiones tienen el mismo valor, de ser así retorna 1 (verdadero) de lo contrario retorna 0(falso).
Diferente	!=	Esta operación comprobará si dos expresiones tienen distinto valor, de ser así retorna 1 (verdadero) de lo contrario retorna 0(falso).
Menor que	<	Esta operación comprobará si la primera expresión es menor a la segunda expresión, de ser así retorna 1 (verdadero) de lo contrario retorna 0(falso).
Mayor que	>	Esta operación comprobará si la primera expresión es mayor a la segunda expresión, de ser así retorna 1 (verdadero) de lo contrario retorna 0(falso).
Menor o igual	<=	Esta operación comprobará si la primera expresión es menor o igual a la segunda expresión, de ser así retorna 1 (verdadero) de lo contrario retorna 0(falso).
Mayor o igual	>=	Esta operación comprobará si la primera expresión es mayor o igual a la segunda expresión, de ser así retorna 1 (verdadero) de lo contrario retorna 0 (falso).

5.9.6 Expresiones lógicas

Estas expresiones utilizarán los conectores lógicos para unir dos condiciones. Sus operandos tienen que ser de tipo booleano, de lo contrario se deberá reportar error. El resultado de la operación será de tipo booleano. En la tabla siguiente se describen los operadores lógicos a implementar.

Nombre	Símbolo	Descripción
--------	---------	-------------

AND	&&	Este operador comprobará que tanto la primera expresión como la segunda expresión tengan valor verdadero. Si esto se cumple retorna 1 (verdadero) de lo contrario retorna 0(falso).
OR	 	Este operador comprobará que la primera expresión o la segunda expresión tenga valor verdadero. Si esto se cumple retorna 1 (verdadero) de lo contrario retorna 0(falso).
NOT	!	Este operador negará el valor de la expresión con la que viene acompañada.

5.9.7 Precedencia de operadores y Asociatividad

En la tabla que se presenta a continuación se define la precedencia y asociatividad de cada uno de los operadores definidos anteriormente, ordenados de mayor precedencia a menor precedencia. Los paréntesis son los únicos símbolos de agrupación a utilizar.

Símbolo	Precedencia	Asociatividad
()	9	Izquierda a derecha
++ -- ! not +(unario) -(unario) *(unario)	8	Derecha a izquierda
/ * %	7	Izquierda a derecha
+ -	6	Izquierda a derecha
< <= > >=	5	Izquierda a derecha
== !=	4	Izquierda a derecha
&&	3	Izquierda a derecha
 	2	Izquierda a derecha

5.9.8 Estructura general

La estructura general del lenguaje se dividirá en dos áreas: área de importaciones y el área de clases

```
<<Importaciones>>
<<clases>>
```

5.9.9 Visibilidad

La visibilidad se refiere al nivel de encapsulamiento que se desee aplicar para una clase y sus miembros, de esta manera se podrá limitar el acceso a ellos. Los tipos de visibilidad aceptados por el lenguaje son:

- **Público:** se podrá acceder a los miembros de una clase desde cualquier lugar.
- **Protegido:** se podrá acceder a los miembros de una clase desde la misma clase o desde otra que herede de ella.
- **Privado:** se podrá acceder a los miembros de una clase sólo desde la misma clase.

Especificaciones sobre visibilidad

- Las variables locales y los parámetros de un método no tienen visibilidad puesto que sólo pueden ser accedidos desde el propio método.

- Si una clase, atributo o método no tienen declarado visibilidad se tomará como público siempre.

5.9.10 Carácter de asignación

Dentro del lenguaje se podrá asignar valor a un atributo, variable, arreglo, objeto, etc. se utilizará "=" este será el único carácter de asignación dentro de todo el lenguaje sin excepciones.

5.9.11 Importaciones

Cuando se necesite hacer referencia a clases que no se encuentran definidas dentro del mismo archivo se realizará una importación de otro archivo que contendrá la definición de otras clases. En esta definición se llamará 'archivo origen' al archivo que realiza la importación y 'archivo importado' al archivo que es llamado desde la importación.

Sintaxis de importación

```
Importar (nombreArchivo.xform);
```

Especificaciones sobre la importación de archivos

- Cuando se importe un archivo solo podrá utilizar sus clases públicas.
- Cuando se importen clases se podrá acceder a todos sus métodos y atributos públicos.
- Cuando se realice una importación y se encuentra una clase que tiene el mismo nombre que otra que ya ha sido definida en el archivo origen, se ignorará la clase importada y se utilizará la clase del archivo origen.
- Cuando se importe un archivo que contiene una clase con método Principal y en el archivo origen se encuentra definida una clase con método Principal se ignorará la clase del archivo importado.
- Cuando se importe un archivo que contiene una clase con un método Principal y no existe una clase con método Principal en el archivo origen utilizará el método Principal de la clase con el archivo importado como método Principal.
- Dentro de los archivos importados existirán varias clases con el mismo nombre, existirá ambigüedad al hacer referencia. Para resolverlo se dará prioridad a los archivos importados según el orden en que aparezcan en la lista de importaciones.
- Las importaciones sólo podrán venir al inicio de un archivo xform.
- Solo se podrá importar archivos xform que se encuentren dentro del mismo proyecto.
- Se podrá importar la cantidad de archivos xform que deseemos.

5.9.12 Clases

Una clase será una plantilla para la creación de objetos con un modelo predefinido, utilizamos una clase para representar entidades.

Sintaxis

```

class nombreClase visibilidad {
...
    Principal () {
        ...
    }
...
}

```

Especificaciones sobre clases

- Un archivo xform podrá contener una o más clases.
- Los atributos y métodos de una clase podrán venir en cualquier orden.
- Las clases privadas sólo podrán ser accedidas desde otras clases que se definirán en el mismo archivo xform.

5.9.13 Herencia

La herencia será un mecanismo que por medio una clase se deriva de otra de manera que extiende su funcionalidad. En este lenguaje para definir la herencia se usará la palabra reservada “**Padre**” seguida del nombre de la clase de la cual heredará.

Especificaciones sobre Herencia

- Cuando una clase hereda de otra se obtendrán todos sus métodos y atributos públicos y protegidos
- Cuando una clase hereda de otra no se obtendrán sus métodos y atributos privados.
- Una clase podrá tener una y sólo una clase padre.
- Una clase podrá ser heredada por muchas clases.
- Cuando una clase hereda de otra no hereda el método constructor.

Sintaxis

```

class nombreClase visibilidad padre nombreClaseHerencia {
...
...
}

```

Constructor de la clase padre:

Cuando se necesite llamar al método constructor de la clase padre se utilizará la palabra reservada “**super**” la cual puede venir seguida de parámetros o no según el tipo de constructor que se haya definido.

```

super(parametro1,...,parametroN );

```

5.9.14 Atributos

Un atributo será una característica de una clase, se entenderá como una ‘variable global’ de la clase a la que pertenece. Los atributos de una clase serán todas las

variables que se encuentran dentro del cuerpo de la clase y fuera de los métodos. Un atributo podrá tener un tipo de dato asociado y una visibilidad asociada.

Sintaxis de atributos

```
tipoDeDato visibilidad ID = expresión;
```

Ejemplos válidos de un atributo

```
decimal numero;  
decimal public numero1;  
decimal public numero2 = 10;  
decimal numero3 = 10;
```

5.9.15 Declaración de variables

Para la declaración de variables será necesario empezar con el tipo de dato de la variable seguido del identificador que esta tendrá. Las variables que se declaren afuera de los métodos serán variables globales.

Sintaxis

```
Tipo Identificador;  
//si se desea inicializar una variable  
Tipo Identificador = Expresión;
```

Ejemplo

```
Entero Identificador;  
//si se desea inicializar una variable  
Entero numero = 10;
```

5.9.16 Asignación de variables

Una asignación de variable consiste en asignar un valor a una variable. La asignación será llevada a cabo con el signo igual (=).

Sintaxis

```
identificador = Expresión;
```

Ejemplo

```
resultado_op = 5 - (9 + variable)*14;
```

5.9.17 Matrices

Una matriz será una estructura de datos, o más técnicamente, un espacio de memoria que permitirá almacenar una colección de elementos.

Especificaciones de las matrices

- Una matriz siempre tendrá el mismo tipo de datos para todos sus elementos.
- Una matriz podrá ser definida de una o varias dimensiones.
- Una matriz podrá contener elementos con tipos de datos primitivos o no primitivos, es decir que una matriz también podrá contener objetos.
- Una matriz tendrá un tamaño estático, es decir no va a variar su tamaño una vez este haya sido definido.

Declaración de matrices

Una matriz podrá ser declarada vacía y sin el tamaño de sus dimensiones, puede ser declarada con los datos que contendrá o podrá ser declarada vacía, pero con el tamaño de sus dimensiones.

Sintaxis

```
decimal numeros [][].....[] = asignacion matriz;  
decimal public numeros [][].....[] = asignacion matriz;  
decimal numeros [][].....[];  
decimal public numeros [][].....[];
```

Asignación de matrices

Una matriz podrá ser asignada al momento de ser declarada o posteriormente. Si una matriz es declarada y asignada en la misma sentencia el tamaño de la matriz será definido por el total de datos que le sean asignados. Si una matriz es asignada luego de su declaración los datos asignados deben coincidir con el tamaño declarado.

Sintaxis

```
decimal numeros [][] = nuevo decimal[2][2];  
Decimal numeros [][] = {{1,2},{3,4}};
```

Tamaño de dimensiones

El tamaño de una dimensión podrá ser definido de dos maneras, con un dato entero. Cuando se vaya a definir una dimensión solo con un dato entero entonces el número ingresado representa el límite superior de la matriz y el límite inferior será el número 0.

Sintaxis de tamaño de dimensiones

```
TipoDeDato ID [][][] = nuevo TipoDeDato[2][3][3];
```

Ejemplos válidos de tamaño de dimensiones

```
Decimal numeros [] = nuevo decimal [5];
```

```
Decimal numeros [][][] = {{3,3,3},{2,2,2},{1,1,1},  
                             {4,4,4},{5,5,5},{6,6,6}}
```

Acceso y asignación de una posición de la matriz

Luego de ser declarada una matriz se podrá asignarle un valor a una posición en específico, también se podrá acceder a un valor que se encuentre guardado en una posición específica.

Sintaxis de asignación a una posición

```
ID[expresión][expresión][expresión] = expresión;  
Números[2*1][0][0] = 25*2/8;
```

Si se intenta acceder a una posición con un dato entero negativo o mayor a los límites definidos en el tamaño de la matriz se deberá mostrar un error semántico.

Para acceder a una posición de una matriz basta con especificar la matriz y la posición en cada una de las dimensiones correspondientes.

Sintaxis de acceso a una posición de una matriz

```
ID[expresión][expresión][expresión];
```

Ejemplo

```
Entero numero = matriz [5][0];
```

5.9.18 Declaración e instanciación de objetos.

Un objeto será la instancia de una clase, es un conjunto de estados (atributos) y de comportamientos (métodos y funciones). Un objeto será un dato cuyo tipo es la clase de la que fue instanciado.

Sintaxis de declaración de un objeto.

```
ClaseA objeto1 = nuevo ClaseA();  
ClaseA public objeto1;  
ClaseA objeto1 = Nulo;  
Objeto1 = otroObjeto;
```

Instanciar un objeto será el acto de reservar memoria para sus atributos. Es decir que si declaramos un objeto y no lo instanciamos no declaramos memoria para este objeto y su referencia sería "Nulo". Cuando se inicie un objeto implícitamente se llamará a su método constructor.

5.9.19 Métodos y funciones

Un método será una subrutina con instrucciones, una función es un método que retornará un valor, los métodos serán contenidos únicamente dentro de las clases, una clase puede contener uno o más métodos.

Sintaxis de un método

```
Visibilidad tipoRetorno nombreMetodo (param1, param2, .... ,paramN) {  
    $$ Instrucciones  
}
```

Si la visibilidad de un método no es definida, se tomará como default la visibilidad de la clase que lo contiene. Dentro de una clase podrán existir más de un método con el mismo nombre, pero deberán diferenciarse por la cantidad o tipo de parámetros.

Variables locales

Una variable local es un espacio de almacenamiento de valores que únicamente podrá ser llamado dentro del método o función donde sea declarado, la declaración de una variable local tendrá la misma sintaxis que un atributo, exceptuando la visibilidad:

```
Tipo nombreVariable = expresion;
```

Sentencia de retorno

La sentencia de retorno tendrá dos funciones diferentes:

- Dentro de los métodos su función será alterar el flujo de ejecución del mismo, si dentro de un método se encuentra la sentencia de retorno se deberá terminar la ejecución del método, sintaxis:

```
Retorno;  
Retorno expresion;
```

Ejemplo de la declaración de un método o función

```
Clase claseA {  
    Publico vacio método (entero a) {  
        Entero b = a;  
        Retorno;  
    }  
    Publico entero método (entero a) {  
        Entero b = a;  
        Retorno b;  
    }  
}
```

5.9.20 Método Principal

Es el punto de entrada para que nuestra aplicación cobre vida dentro de lenguaje, en este método se iniciará la ejecución de todas las sentencias que definirán el comportamiento de la aplicación que se ha escrito, tendrá la siguiente sintaxis:

Sintaxis

```
Principal () {  
    $$Instrucciones  
}
```

5.9.21 Constructor

Un constructor es un método especial que se utiliza para definir el comportamiento inicial de un objeto al ser instanciado, su objetivo es poder definir valores válidos para el objeto, un constructor tiene las siguientes características:

- Su nombre deberá ser el mismo que la clase que lo contiene.
- No debe devolver ningún valor.
- Su visibilidad es pública.
- No se hereda.
- Puede contener parámetros

```
Clase claseA {  
    claseA(entero a) {  
        Entero b = a;  
    }  
    Publico entero método (entero a){  
        Entero b = a;  
        Retorno b;  
    }  
}
```

5.9.22 Acceso a atributos y métodos de un objeto.

Dentro de las sentencias del lenguaje de alto nivel se encuentran el acceso de atributos y métodos, para definirlos los dividiremos en dos grupos.

Atributos y métodos de la misma clase

Cuando deseamos acceder a los atributos y métodos de la clase en la que nos encontramos basta con llamarlos por su nombre propio, o bien utilizar la palabra reservada “**Este**” seguida del atributo o método al que deseamos acceder.

Sintaxis de acceso a atributos y métodos de la misma clase

```
METODO (param, param1, ..., paramN);  
ID = expresion;  
Este.Id = expresion;  
este.Metodo();
```

El lenguaje soporta llamadas a métodos y funciones de forma recursiva.

Atributos y métodos de un objeto

Para poder acceder a los atributos y métodos de un objeto que se ha realizado una instancia se deberá utilizar el nombre del objeto seguido por punto "." y el nombre del atributo o método al que se desea acceder.

Sintaxis de acceso a atributos y métodos de un objeto

```
ID = ID.ID(param1).ID;  
ID = ID.ID.ID();  
:  
ID.ID.ID = ID.ID();
```

5.9.23 Sentencias de control

Sentencia SI

Sentencia cuya función determinará el flujo que el programa deberá seguir entre una acción u otra dependiendo de una o varias condiciones establecidas por el programador.

Sintaxis:

```
SI (expresion_logica){  
    $$ instrucciones  
}
```

Ejemplo

```
SI(a < b) {  
    $$ instrucciones  
}
```

Sentencia Si Sino

Se deberá utilizar cuando se desea tener varias instrucciones en el caso que la condición sea verdadera y otras instrucciones cuando la condición sea falsa.

Sintaxis

```
SI (a < b) {  
    $$ instrucciones  
} SINO {  
    $$ instrucciones  
}
```

Sentencia Si Sino Si

Cuando se necesitan diferentes acciones para diferentes casos (condiciones), será posible utilizar esta sentencia, esta sentencia se podrá agregar varias instrucciones que se ejecuten en el caso que no se cumpla ninguna condición.

Sintaxis

```
SI(a < b) {  
    $$ instrucciones  
} SINO SI (a < b) {  
    $$ instrucciones
```

```
} SINO SI (a < b) {  
  $$ instrucciones  
} SINO {  
  $$ instrucciones  
}
```

Sentencia Caso De

Es una sentencia de anidamiento de múltiples instrucciones si...sino si, sobre un caso en particular siendo aplicado sobre una expresión, cuando una condición se cumpla se deberá ejecutar el grupo de instrucciones contenidas dentro de las condiciones, en caso no se cumpla ninguna condición el “caso” puede incluir un set de instrucciones que será ejecutadas por defecto, estas se deberán ejecutar hasta encontrar una sentencia de que indique su salida.

- **Sentencia romper:** Indicará cuándo se terminará de ejecutar sentencias dentro del case de. Puede venir en cualquier parte, no necesariamente solo al final. Si no aparece la sentencia de salida dentro del “caso”, entonces se deberá seguir comparando los resultados siguientes.

Sintaxis

```
romper;
```

Sintaxis

```
Caso (expresión) de  
{  
  Valor1: { $$ instrucciones1  
            romper;  
  } Valor2: {  
            $$instrucciones2  
            romper;  
  } Valor3: {  
            $$instrucciones3  
            romper;  
  } Defecto: {  
            $$instrucciones Defecto  
  }  
}
```

Sentencia Si Sino simplificada

Al realizar una asignación u obtener un único valor que dependerá de una condición, permitiendo el uso de un condicional “Si Sino” simplificado.

Sintaxis

```
Condicion ?valor_si_verdadero: valor_Si_falso;
```

Ejemplo

```
A < B ?25: 50;  
C = A < B ?25: 50;  
ID.ID = A < B ?25: 50;  
Este. Id = A < B ?25: 50;
```

5.9.24 Bucles

Un bucle o ciclo, es una instrucción que deberá ejecutar cada sentencia contenida en ella repetidas veces hasta que una condición asignada deje de cumplirse.

- **Sentencia Romper:** Esta sentencia se utilizará para salir de cualquier bucle, romper solo puede ser utilizada siempre que se encuentre dentro de un bucle.

Sintaxis

```
Romper;
```

- **Sentencia Continuar:** Esta sentencia se utilizará para indicarle al bucle que termine el ciclo que se encuentra y que evalúe la condición si se sigue cumpliendo la condición debe continuar la ejecución del ciclo.

Sintaxis

```
Continuar;
```

Sentencia Mientras

Es un ciclo que se ejecutará mientras una condición se siga cumpliendo.

Sintaxis

```
Mientras (condicion)  
{  
    $$instrucciones  
}
```

Ejemplo

```
Mientras (a < b)  
{  
    a = a + 1;  
}
```

Sentencia Hacer Mientras

Es un ciclo que deberá ejecutarse al menos una vez, luego debe seguir la ejecución mientras la condición se siga cumpliendo.

Sintaxis

```
Hacer {  
    $$instrucciones  
} mientras (condicion);
```

Ejemplo

```
Hacer {  
    A = A + 1;  
} mientras (A < 10);
```

Sentencia Repetir Hasta

Es un ciclo que deberá ejecutarse al menos una vez y continuará su ejecución mientras la condición NO se cumpla.

Sintaxis

```
Repetir {  
    $$instrucciones  
} hasta (condicion);
```

Ejemplo

```
a = 1000;  
Repetir {  
    a = a - 1;  
} hasta (a < b);
```

Sentencia Para

Es un bucle que permitirá inicializar o establecer una variable como variable de control, el ciclo contendrá una condición que se verificará en cada iteración, luego se deberá definir una operación que afecte directamente a la variable de control cada vez que se ejecuta un ciclo para volver a verificar si se cumple la condición. Su uso es ideal para los casos en donde se conocerá el número de repeticiones o se necesitará hacer un barrido a vectores o matrices.

Sintaxis

```
Para (variable de control; condicion; operacion) {  
    $$instrucciones  
}
```

Ejemplo

```
Para (entero a = 0; a < 10; a++) {  
    $$instrucciones  
}
```

```
Entero a;  
Para (a = 0; a < 10; a++) {  
  $$instrucciones  
}
```

5.9.25 Imprimir

Esta sentencia recibirá un valor de cualquier tipo de dato y lo mostrará en la consola de salida.

Sintaxis

```
Imprimir( Valor );
```

Ejemplo

```
Imprimir ( "el texto que quiero mostrar");  
Imprimir ('a');  
Imprimir (5);  
Imprimir (Variable);  
Imprimir (Verdadero);
```

5.9.26 Diálogos

Los diálogos serán cajas de mensajes, se utilizarán para desplegar mensajes importantes para el usuario al momento de hacer uso los formularios. Esto solo se podrán utilizar dentro de las plantillas "Formulario", "Grupo" y "Pregunta".

Sintaxis

```
Mensajes (argumentos);
```

Ejemplo

```
Mensaje ("Hola mundo");
```

5.9.27 Funciones Nativas del Lenguaje

El lenguaje de alto nivel admite un subconjunto de funciones las cuales realizarán la acción al momento de hacer una referencia o un llamado hacia la instancia. Para mayor facilidad, las funciones se clasificarán según su uso principal.

5.9.28 Funciones con cadenas

Convertir a cadena

La función 'Cadena' convierte cualquier dato en una cadena. Retorno: cadena

Sintaxis

```
Cadena (Valor)
```

Ejemplo

```
Cadena id = Cadena(105);
```

Sub-Cadena

Devuelve la sub-cadena que comienza en el índice de inicio basado en 0 especificado y se extiende al carácter en el índice final menos una posición. Retorno: cadena

Sintaxis

```
subCad(cadena, número de inicio, numero fin)
```

Ejemplo

```
Cadena hola = sudCad ("Hola mundo", 0, 4); $$ retorno "Hola"
```

Obtener Carácter de una Posición

Devolverá el carácter que se encuentre en una posición especificada, la posición inicial de la cadena es 0. Si la posición es un valor fuera de rango devolverá un error semántico. Retorno: Carácter.

Sintaxis

```
PosCad(Cadena, posición)
```

Ejemplo

```
Caracter hola = poscad("Hola mundo", 3); $$ retorno 'a'
```

5.9.29 Funciones Booleanas

Convertidor a booleano

La función booleana convierte su argumento en booleano de la siguiente manera:

- Un número es verdadero si y solo si es positivo, un número mayor a 0.
- Un objeto es verdadero si y solo si no se encuentra vacío (Nulo).
- Una cadena es verdadera si y solo si su longitud es distinta de cero.

Retorno: booleano

Sintaxis

```
Booleano (Argumento)
```

Ejemplo

```
Booleano calculate= booleano ('false')  
Booleano calculate= booleano (0)  
Booleano calculate= booleano (1)  
Booleano calculate= booleano ('true')
```


5.9.30 Funciones numéricas

Convertir a número

Esta función convertirá un argumento en un número, si su parámetro es:

- Una cadena, devolverá la suma del ascci de todos los caracteres,
- Si el parámetro es falso devolverá un 0 y si su parámetro es verdadero devolverá 1.
- Convertirá cadenas con formato fecha, hora y fechaHora a varios días, tomando como día y hora inicial 01/01/2000 00:00:00.
- Si el parámetro es un número con decimales, devolverá el número más cercano al parámetro realizando una aproximación.

Retorno: entero

Sintaxis

Entero (argumentos)

Ejemplo

<pre>Entero calculate= Entero ("100") Entero calcular = Entero("02/01/2000 12:00:00"); \$\$retorno 1.5 Entero calcular = Entero("02/01/2000"); \$\$retorno 1 Entero calcular = Entero("12:00:00"); \$\$retorno 0.5 dias</pre>

Tamaño de un Objeto

Devolverá el tamaño de un objeto. Retorno: entero.

Sintaxis

Tam(Objeto)

Ejemplo

<pre>Entero hola = Tam("Hola mundo"); \$\$ retorno "10" Entero hola1 = Tam(hola); \$\$ retorno "10"</pre>

Aleatorio

Devolverá un número decimal aleatorio entre 0 y 1, si no contiene ningún parámetro, caso contrario devolverá un valor aleatorio de los parámetros asignados. Retorno: decimal, entero, cadena

Sintaxis

```
random (* agumentos)
random ()
```

Ejemplo

```
Carácter calculate= random ('a', 'b', 'c', 'd', 'e', 'f') $$devolverá cualquier letra
Decimal calculate= random () $$devolverá un valor decimal entre 0 y 1
Entero calculate= random () * 100
```

Función mínima

La función mínima devolverá el valor mínimo o más pequeño de los parámetros, si el parámetro incluye cadenas, deberá de convertirse primero en un número. Si el parámetro es un conjunto de objetos vacío, o si alguno de los objetos es evaluado como Nulo, el valor de retorno deberá ser 0. Retorno: Entero o decimal, 0

Sintaxis

```
min (* agumentos )
```

Ejemplo

```
Decimal calculate= min ('a', 9.45, 15.5, 15.5, 'b'); $$devolvería 15.5
Entero calculate= min ('a', 'b', 'c', 6); $$devolvería 6
Entero calculate= min (miArreglo); $$sí miArreglo viene vacío, devolvería 0
Entero calculate= min ('a', 10, 15.5, 15.5, 'b', Objeto().valor); $$devolvería 10
```

Función Máxima

La función máxima devolverá el valor máximo o valor más alto de los parámetros, si el parámetro incluye cadenas, deberá de convertirse primero en un número. Si el parámetro es un conjunto de objetos vacío, o si alguno de los objetos es evaluado como Nulo, el valor de retorno deberá ser 0.

Retorno: Entero o Decimal

Sintaxis

```
min (* agumentos )
```

Ejemplo

```
Entero calculate= max ('a', 9.45, 15.5, 15.5, 'b') ; $$devolvería 98
Entero calculate= max ('a', 'b', 'c', 6); $$devolvería 99
Entero calculate= max (miArreglo); $$devolvería 0, si miArreglo fuera un objeto nulo
Entero calculate= max ('a', 10, 15.5, 15.5, 'b', Objeto().valor); $$devolvería 98
```

Operaciones matemáticas

Las siguientes funciones, realizará la operación descripta.

Nombre	Descripción	Sintaxis	Retorno
Potencia	Devolverá la potencia del primer parámetro elevado con el segundo parámetro.	pow(expresión, numero potencia)	Decimal
Logaritmo	Devolverá el logaritmo del número de parámetro.	Log(expresión)	Decimal
Logaritmo base 10	Devolverá el logaritmo del número de parámetro con base 10.	Log10(expresión)	Decimal
Absoluto	Devolverá el valor absoluto del número de parámetro.	Abs(expresión)	Entero, decimal
Seno	Retornará el seno del parámetro, expresado en radianes	Sin(expresión)	Decimal
Coseno	Retornará el coseno del parámetro, expresado en radianes	Cos(expresión)	Decimal
Tangente	Retornará la tangente del parámetro, expresado en radianes	tan(expresión)	Decimal
Raíz cuadrada	Retornará la raíz cuadrada no negativa del parámetro.	Sqrt(expresión)	Decimal
Función Pi	Retornara el valor del número Pi, con un máximo de 5 decimales.	Pi()	Decimal

Ejemplo

```
$$<!-- retornará 3125 -->  
Entero calculate= pow(5,5);  
$$ retronará 13.14159 -->  
Decimal calculate= pi () + 13  
$$ suponiendo que el nodo 'ah' es igual a 5 retornará 23.60 -->  
Decimal calculate= Sqrt (Objeto().ah);
```

5.9.31 Funciones de Fecha y Hora

Función hoy

Devolverá una cadena con la fecha actual en formato día/mes/año. Retornará: fecha

Sintaxis

```
hoy()
```

Ejemplo

```
Fecha calculate= hoy();
```

Función ahora

Devolverá la fecha y hora actual en formato dd/mm/yy hh:mi:ss. Retornará: Fecha y Hora

Donde:

- dd = día
- mm = mes
- yy = año
- hh = hora
- mi = minutos
- ss = segundos

Sintaxis

```
ahora ()
```

Ejemplo

```
FechaHora calculate= ahora();
```

Conversiones con fecha y hora

Convertirá el valor en formato tipo Fecha, Hora o fecha y hora desde una cadena como parámetro. Retornará: fecha, hora o fecha y hora.

Sintaxis

```
fecha( Cadena )  
Hora( Cadena )  
fechaHora( Cadena )
```

Ejemplo

```
Fecha calculate= Fecha ('15/10/2000');  
Hora calculate= Hora ("20:12:00");  
FechaHora calculate= FechaHora ('15/10/2000 20:12:00');
```

5.9.32 Funciones Multimedia

Son funciones que se utilizarán para poder mostrar ficheros tipo multimedia, este tipo de función solo se podrá usar dentro del procedimiento “Mostrar” y este deberá de encontrarse dentro de una plantilla Pregunta o Grupo. Los tipos de multimedia son: Imagen, Video, Audio.

Sintaxis

```
Imagen (ruta, autoplay[verdadero|Falso]);  
video(ruta, autoplay[verdadero|Falso]);  
audio (ruta, autoplay[verdadero|Falso]);
```

Ejemplo

```
imagen ("/ruta/relativa/a.jpg", verdadero);  
video ("/ruta/relativa/b.mp4", verdadero);  
audio ("/ruta/relativa/c.mp3", Falso);
```

5.9.33 Listas de Opciones

El lenguaje de alto nivel, tiene una estructura de datos predefinida, la cual ayudará a un mejor manejo de información de forma dinámica, de esta forma el lenguaje no se limitará únicamente a la utilización de vectores.

Esta lista de opciones podrá contener diferentes tipos de datos, estas crecerán o disminuirán su tamaño según sea necesario. Las listas utilizarán la palabra reservada "Opciones" y una vez sean instanciadas estas se encontrarán vacías. Estas listas podrán contener cualquier tipo de dato, dependiendo del uso que se dé, si la lista de opciones es usada en los "Formularios", "Grupos" o "Preguntas" deberá seguir el siguiente orden para los valores a insertar: nombre, etiqueta y ruta; en otro caso se podrá insertar cualquier elemento y de cualquier tipo en el orden que se desee.

Sintaxis

```
Opciones opcion = Nuevo Opciones ( );
```

Insertar

Este método permitirá insertar un elemento en la lista de datos, al momento de insertar un elemento este será insertado al final de la lista. Se podrá insertar varios elementos en una sola posición.

Sintaxis

```
Identificador.insertar(Elemento, elemento 2, elemento 3, ... elemento N);
```

Ejemplo

```
Opciones miLista = nuevo Opciones ();  
miLista.insertar("cadena", 10, 15 ,56 );  
miLista.insertar("Otra", 1, 2, 3);  
miLista.insertar("Otra mas", 4, 5, 6);
```

Acceso a elementos

El acceso a elementos de la lista se podrá realizar a través del nombre de la lista y la utilización de un índice numérico. El índice deberá ir dentro de llaves seguido del nombre de la lista. Si el índice esta fuera de la cantidad de elementos de la lista devolverá un resultado nulo. Los índices iniciarán en cero.

Sintaxis

```
Identificador.obtener(índice);
```

Ejemplo

```
Opciones miLista = nuevo Opciones ();  
Entero miNumero;  
miLista.insertar("cadena", 10, 15 ,56 );  
miLista.insertar("Otra", 1, 2, 3);  
miLista.insertar("Otra mas", 4, 5, 6);  
  
miNumero = miLista.obtener(2) [1]; $$miNumero=1  
miNumero = miLista.obtener(0) [3]; $$miNumero=56
```

Buscar

Esta sentencia buscará por elemento en la lista devolviendo el índice del mismo dentro de la lista. Si el elemento no se encontrase dentro de la misma devolverá un valor nulo.

Sintaxis

```
Identificador.buscar();
```

Ejemplo

```
Opciones miLista = Opciones Lista();  
Entero milIndice;  
  
miLista.insertar("cadena", 10, 15 ,56 );  
miLista.insertar("Otra", 1, 2, 3);  
miLista.insertar("Otra mas", 4, 5, 6);  
milIndice = miLista.buscar("Cadena")[3]; $$milIndice=56  
milIndice = miLista.buscar("Otra mas")[2]; $$milIndice=5
```

5.9.34 Formulario y Grupos

Los Formularios serán una plantilla que permitirá la creación de objetos de tipo pregunta y grupos. Dentro de la plantilla podrá venir atributos, procedimiento y funciones.

- Existirá un objeto llamado "Respuesta", este solo podrá ser llamado dentro de estas plantillas.
- Un grupo no requiere la necesidad de una instancia para poder ser utilizado.
- Los "Formularios" y los "Grupos", se encuentran dentro una clase.

- Para realizar la llamada a un formulario, se deberá utilizar la palabra reservada “Nuevo” seguido el nombre del formulario, en la llamada del formulario se podrá definir el atributo con el tipo de vista del formulario.

Sintaxis

```
Clase Ejemplo{
    Formulario Nombre_Formulario {
        ...
        Respuesta miRes;
        ...
    }
    Grupo ID_Grupo { $$varios atributos, procedimientos y funciones
        ...
        Respuesta miRes;
        ...
    }
    Principal ( ) {
        Nuevo Nombre_Formulario ( ).Pagina;
        Nuevo Nombre_Formulario ( ).Grupo;
        Nuevo Nombre_Formulario ( ).Todo;
        Nuevo Nombre_Formulario ( ).Cuadrícula;
    }
}
```

Ejemplo

\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```
Clase Ejemplo {
    Pregunta miPregunta ( ) {
        cadena etiqueta = “Cuántos hijos tienes?”;
        entero respuesta;
        publico respuesta (entero param_1) {
            respuesta = param_1;
        }
    }
    Formulario parámetros {
        Respuesta res;
        $$Parámetro no definido, inicio en 0
        miPregunta ( ). Respuesta (res.esEntero). rango(0, 10);
    }
    Principal ( ) {
        Nuevo Nombre_Formulario ( ).Pagina;
        $$Nuevo Nombre_Formulario ( ).Grupo;
        $$Nuevo Nombre_Formulario ( ).Todo;
        $$Nuevo Nombre_Formulario ( ).Cuadrícula;
    }
}
```

5.9.35 Preguntas

Las Preguntas y Grupos son plantillas para la creación de objetos, que servirán para el uso y creación de formularios. Una “Pregunta” tiene la misma funcionalidad que un

“Formulario” o “Grupo” con la diferencia de que existen atributos y procedimientos reservados, los cuales solo se podrán usar dentro de estas plantillas. La plantilla Pregunta no tiene la necesidad de una instancia para poder utilizarla ya que pertenecen a una clase.

Sintaxis

```
Clase Ejemplo{
    Pregunta ID_Pregunta{
        $$varios atributos, procedimientos y funciones
    }
}
```

Atributos Específicos

Atributo	Descripción	Ejemplo
Etiqueta	Este atributo mostrará la pregunta en la interfaz del formulario.	Cadena etiqueta = "EtiquetaPregunta";
Sugerir	Etiqueta de pregunta, esta se mostrará junto a la pregunta una sugerencia hacia la misma.	Cadena sugerir = "ServiráParaSugerirUnarRspuesta";
Respuesta	Este atributo almacenará la respuesta obtenida en el formulario, este atributo puede ser de diferentes tipos de datos.	Caracter respuesta = 'a'; Entero respuesta = 100; Cadena respuesta = 'a';
requeridoMsn	Almacenará el mensaje que se deberá mostrar al momento de omitir una pregunta.	requeridoMsn="Conteste la pregunta."
ruta	Este atributo contendrá la ruta de algún fichero.	Cadena ruta = "/ruta/no/relativa";

Procedimientos Específicos

En la siguiente tabla se describirán los procedimientos que podrán existir en las plantillas de “Pregunta”.

Procedimiento	Descripción	Ejemplo
Respuesta	Este procedimiento realizará la asignación de la respuesta obtenida del usuario. Lo componen varios elementos para el manejo, vistas y condiciones para entradas del formulario, la cual serán descritos en el siguiente punto.	<code>publico respuesta</code> (Entero param_1) { respuesta = param_1; }

Mostrar	Este procedimiento realizará las operaciones necesarias para la visualización de contenido multimedia. No lo compone ningún elemento para el manejo de formularios.	<code>publico Mostrar () { Audio (Ruta, verdadero); }</code>
Calcular	Este procedimiento deberá ser usado específicamente para realizar cálculos u operaciones con expresiones. No lo compone ningún elemento para el manejo de formularios.	<code>publico calcular (decimal cantidad) { respuesta=cantidad*0.18; }</code>

5.9.36 Procedimiento respuesta

Contiene los elementos que son de utilidad para la vista de los formularios, estos definirán la forma y el tipo a mostrar para la obtención de los datos.

Cadena

Mostrará una caja de texto, los cuales se podrá asignar cualquier tipo de texto como respuesta. Este tipo puede tener tres tipos de parámetros o ninguno, los cuales son: El tamaño mínimo, máximo y número de saltos de línea que puede tener una cadena, si un parámetro es faltante o no se desea asignar se deberá hacer uso de la palabra reservada 'Nada', haciendo referencia que no existe ningún parámetro asignado a esa posición.

Sintaxis

\$\$\$Son llamadas desde la plantilla Formulario de esta manera
Nombre_Pregunta().Respuesta() . cadena (mínimo, máximo, fila);
Nombre_Pregunta().Respuesta() . cadena ();

Ejemplo

Nombre_Pregunta().Respuesta (res. Escadena). cadena (Nada, 20, 10);
Nombre_Pregunta().Respuesta ("un prametro"). Cadena ();

Vista

Diagrama de una interfaz de usuario para una pregunta. El formulario tiene un título "Pregunta" y una descripción: "1) Escriba un resumen de una noticia importante reciente." Debajo de la descripción, hay una sugerencia para responder: "Campéon actual del mundo, último presidente de Guatemala, etc." Hay un campo de texto con el placeholder "Escriba su respuesta aquí." y un botón "Aceptar".

Entero

Mostrará una caja de tipo numérico, la cuales se podrá asignar solo valores de tipo entero como respuesta.

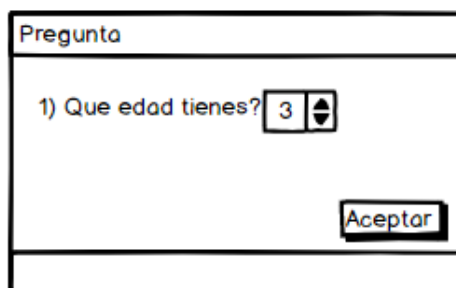
Sintaxis

```
$$Son llamadas desde la plantilla Formulario de esta manera  
Nombre_Pregunta( ).Respuesta( ) . Entero ( );
```

Ejemplo

```
Nombre_Pregunta( ).Respuesta (res. EsEntero). Entero ();  
Nombre_Pregunta( ).Respuesta (res.EsEntero ). Entero ();
```

Vista



The screenshot shows a form with a title bar labeled 'Pregunta'. Inside the form, there is a question '1) Que edad tienes?' followed by a numeric input field containing the value '3'. To the right of the input field is a small button with up and down arrows. At the bottom right of the form is a button labeled 'Aceptar'.

Decimal

Mostrará una caja de tipo decimal, la cuales se podrá asignar solo valores de tipo decimal como respuesta.

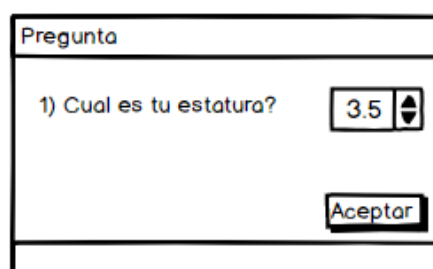
Sintaxis

```
$$Son llamadas desde la plantilla Formulario de esta manera  
Nombre_Pregunta( ).Respuesta( ) . Decimal ( );
```

Ejemplo

```
Nombre_Pregunta( ).Respuesta (res. esDecimal). Decimal ();  
Nombre_Pregunta( ).Respuesta (res.esDecimal ). Decimal ();
```

Vista



The screenshot shows a form with a title bar labeled 'Pregunta'. Inside the form, there is a question '1) Cual es tu estatura?' followed by a decimal input field containing the value '3.5'. To the right of the input field is a small button with up and down arrows. At the bottom right of the form is a button labeled 'Aceptar'.

Booleano

Mostrará dos botones, la cual se podrá seleccionar un solo valor de tipo booleano como respuesta.

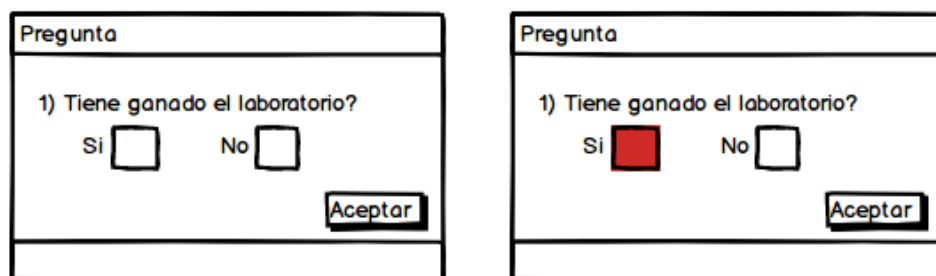
Sintaxis

```
$$$Son llamadas desde la plantilla Formulario de esta manera  
Nombre_Pregunta( ).Respuesta(res. esbooleano) . Condicion( Cad_T, Cad_F);
```

Ejemplo

```
Nombre_Pregunta( ).Respuesta(res. esbooleano) . Condicion ( 'Verdadero', 'Falso');  
Nombre_Pregunta( ).Respuesta(res. esbooleano) . Condicion ( "Si", "No");
```

Vista



Nota

Mostrará un mensaje, este tipo de elemento no obtendrán ningún valor como respuesta.

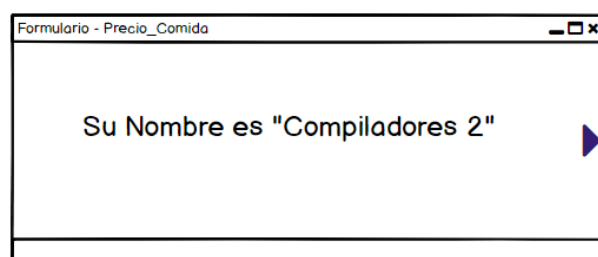
Sintaxis

```
$$$Son llamadas desde la plantilla Formulario de esta manera  
Nombre_Pregunta( ). nota ( );
```

Ejemplo

```
Nombre_Pregunta( ).nota ( );
```

Vista



Rango

Mostrará una caja de tipo numérico, donde verificará el limite inicial y final dados como parámetro, el valor obtenido como respuesta es de tipo entero.

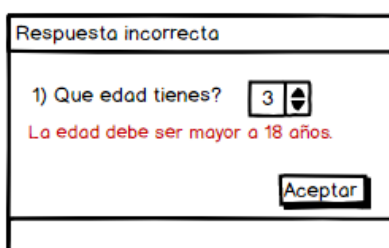
Sintaxis

```
$$$Son llamadas desde la plantilla Formulario de esta manera  
Nombre_Pregunta( ).Respuesta( ) . rango (Límite inferior, Límite superior);
```

Ejemplo

```
Nombre_Pregunta( ).Respuesta (res.esEntero ). rango (18, 100);
```

Vista



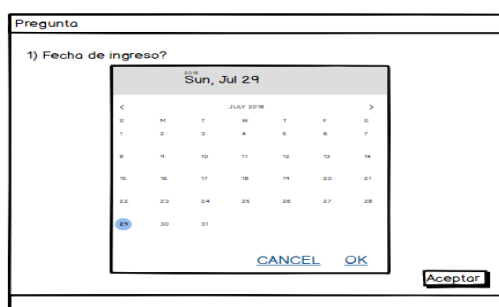
Fecha

Mostrará un calendario, para poder seleccionar una fecha, el valor obtenido como respuesta es de tipo fecha.

Sintaxis

```
$$$Son llamadas desde la plantilla Formulario de esta manera  
Nombre_Pregunta( ). Respuesta (res.esFecha). fecha( );
```

Vista



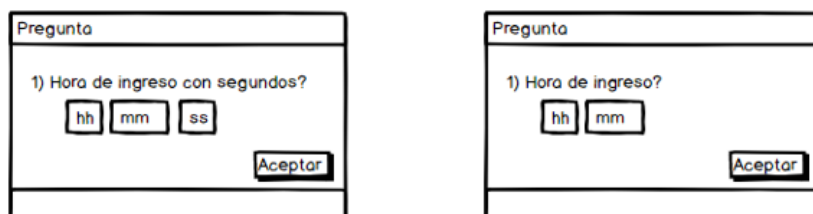
Hora

Mostrará un reloj, para poder seleccionar una hora específica, el valor obtenido como respuesta es de tipo hora.

Sintaxis

```
$$$Son llamadas desde la plantilla Formulario de esta manera  
Nombre_Pregunta( ). Respuesta (res.esHora). Hora( );
```

Vista



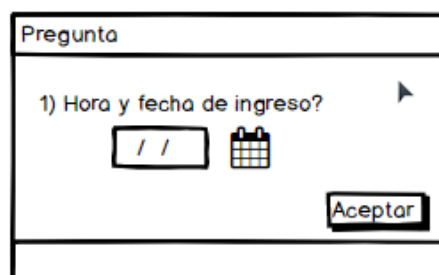
Fecha y Hora

Mostrará un Calendario y un reloj, para poder seleccionar una fecha y la hora específica, el valor obtenido como respuesta es de tipo fecha y hora.

Sintaxis

\$\$\$Son llamadas desde la plantilla Formulario de esta manera
Nombre_Pregunta(). Respuesta (res.esHora). fechaHora();

Vista



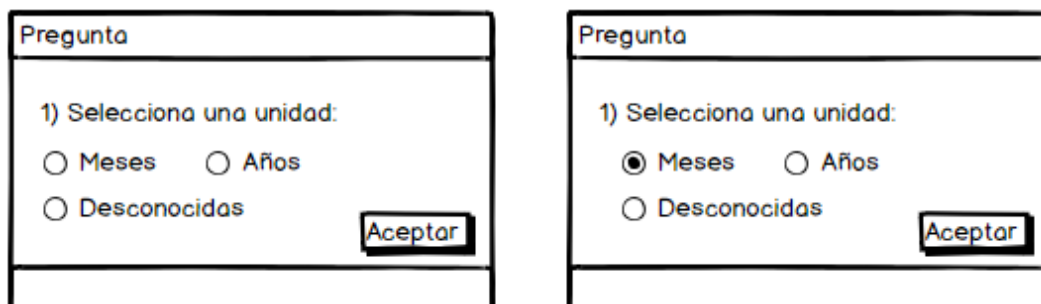
Seleccionar uno

Este atributo permitirá al usuario poder seleccionar un valor entre los distintos y posibles respuestas.

Sintaxis

\$\$\$Son llamadas desde la plantilla Formulario de esta manera
Nombre_Pregunta(). Respuesta (res.esHora). Seleccionar_1 (Lista_Opcion);

Vista



Seleccionar Múltiples

Este atributo permitirá al usuario poder seleccionar varios valores entre los distintos y posibles respuestas.

Sintaxis

```
$$$Son llamadas desde la plantilla Formulario de esta manera  
Nombre_Pregunta ( ). Respuesta (res.esHora). Seleccionar ( Lista_Opcion );
```

Vista



El formulario tiene un título "Pregunta" y una pregunta "1) Frutas Favoritas?". Debajo de la pregunta, hay tres opciones: "Manzana", "Pera" y "Banano". Cada opción tiene un cuadro de selección a su derecha. El cuadro de "Manzana" está seleccionado (rojo), el de "Pera" no (blanco) y el de "Banano" está seleccionado (rojo). En la parte inferior derecha del formulario hay un botón "Aceptar".

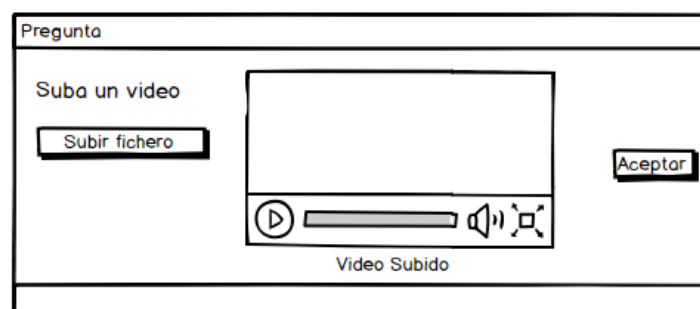
Subir Fichero

Este atributo permitirá al usuario poder seleccionar un archivo en el formato especificado y almacenarlo como respuesta.

Sintaxis

```
$$$Son llamadas desde la plantilla Formulario de esta manera  
Nombre_Pregunta ( ).Fichero( ".png, .jpg, .gif" );  
Nombre_Pregunta ( ).Fichero( ".mp3" );  
Nombre_Pregunta ( ).Fichero( );
```

Vista



El formulario tiene un título "Pregunta" y una instrucción "Suba un video". Debajo de la instrucción, hay un botón "Subir fichero". A la derecha del botón, hay un área de visualización de video con un reproductor de video que incluye un botón de play, una barra de progreso, un icono de altavoz y un icono de pantalla completa. En la parte inferior derecha del formulario hay un botón "Aceptar".

6 Ejemplos de Entrada y Salida

Se definirán varios ejemplos de las entradas (XLS) y su respectiva salida en lenguaje de alto nivel y vista del formulario o pregunta.

6.1 Tipos de Columnas

Se definirá ejemplos con los tipos de columnas que maneja el estándar para la elaboración de formularios.

Columna de Sugerencias (Sugerir)

Vista Tabla

tipo	IdPregunta	etiqueta	sugerir
texto	capital	Capital de Guatemala?	Inicia con la palabra Ciudad.

Salida

\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```
Clase ejemplo {
    Principal () {
        Nuevo Ejemplo ();
    }
    Formulario Ejemplo () {
        Respuesta res;
        Capital ().respuesta (res.esCadena).cadena( );
    }
    Pregunta capital ( ) {
        cadena etiqueta = "Capital de Guatemala?";
        cadena sugerir = "Inicia con la palabra Ciudad.";
        cadena respuesta = Nulo;
        publico respuesta (cadena param_1) {
            respuesta = param_1;
        }
    }
}
```

Vista Formulario

Columnas Código Pre y Post

Vista Tabla

tipo	IdPregunta	etiqueta	Codigo_pre	Código_post
texto	capital	Capital de Guatemala?	Imprimir ("Consola saldrá Esto");	Imprimir ("Contesto la Capital " + @);

Salida

\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```
Clase ejemplo {
    Principal () {
        Nuevo Ejemplo ();
    }
    Pregunta capital ( ) {
        cadena etiqueta = "Capital de Guatemala?";
```

```

cadena sugerir = "Inicia con la palabra Ciudad.";
cadena respuesta = Nulo;
publico respuesta (cadena param_1) {
    respuesta = param_1;
}
}
Formulario Ejemplo () {
    Imprimir ( "Consola saldrá Esto" ); $$se agregó esto como PRE
    Capital ().respuesta (res.esCadena).Cadena;
    Imprimir ("Contesto la Capital " + capital ().respuesta ); $$se agregó esto como POST
}
}

```

Columna de restricciones (restringir)

Vista tabla

tipo	IdPregunta	etiqueta	Restringir
entero	edad	Cuántos años tienes?	.<= 18

Salida

\$\$ el ejemplo mostrará solo donde afecta las etiquetas

```

Clase ejemplo {
    Principal () {
        Nuevo Ejemplo ();
    }
    Formulario Ejemplo () {
        Respuesta res;
        edad ().respuesta (res.esEntero).cadena( );
    }
    Pregunta edad ( ) {
        cadena etiqueta = "Cuántos años tienes?";
        entero respuesta;
        publico respuesta (entero param_1) {
            si (param_1 <= 18) {
                respuesta = param_1;
            }
        }
    }
}

```

Vista

Respuesta incorrecta

1) Que edad tienes?

La edad debe ser mayor a 18 años.

Aceptar

Columna de Mensaje de restricción (RestringirMsn)

Será el mensaje que se muestre al usuario si la respuesta no fue valida.

Vista tabla

tipo	IdPregunta	etiqueta	Restringir	RestringirMsn
entero	edad	Cuanto años tienes?	. <= 18 && .>10	Debes de ser mayor de edad

Salida

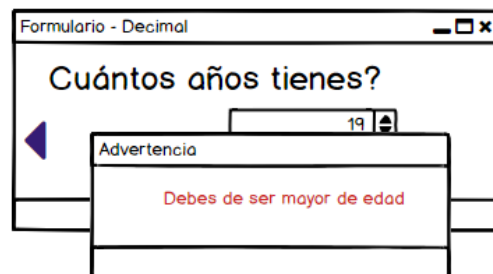
\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```

Clase ejemplo {
    Principal () {
        Nuevo Ejemplo ();
    }
    Formulario Ejemplo () {
        Respuesta res;
        edad ().respuesta (res.esEntero).cadena( );
    }
    Pregunta edad ( ) {
        cadena etiqueta = "Cuántos años tienes?";
        entero respuesta;
        publico respuesta (entero param_1) {
            si (param_1 <= 18 && param_1 > 10) {
                respuesta = param_1;
            }
            sino {
                mensaje ("Debes de ser mayor de edad");
            }
        }
    }
}

```

Vista



Columna valor necesario (requerido)

Vista tabla

tipo	IdPregunta	etiqueta	restringir	requerido
entero	edad	¿Cuántos años tienes?	. <= 150	verdadero

Salida

\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```

Clase ejemplo {
    Principal () {
        Nuevo Ejemplo ();
    }
    Formulario Ejemplo () {
        Respuesta res;
        edad ().respuesta (res.esEntero).cadena( );
    }
    Pregunta edad ( ) {

```

```

cadena etiqueta = "Cuántos años tienes?";
entero respuesta;
booleano requerido = verdadero;
publico respuesta (entero param_1) {
    si (param_1 <= 150) {
        respuesta = param_1;
    }
}
}
}

```

Vista

Columna Mensaje de valor necesario (requeridoMsn)

Vista tabla

tipo	IdPregunta	etiqueta	requeridoMsn	requerido
entero	ages	¿Cuántos años tienes?	Ingrese el dato	verdadero

Salida

\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```

Clase ejemplo {
    Principal () {
        Nuevo Ejemplo ();
    }
    Formulario Ejemplo () {
        Respuesta res;
        edad ().respuesta (res.esEntero).cadena( );
    }
    Pregunta edad ( ) {
        cadena etiqueta = "Cuántos años tienes?";
        entero respuesta;
        booleano requerido = verdadero;
        cadena requeridomsn = "Ingrese el dato";
        publico respuesta (entero param_1) {
            respuesta = param_1;
        }
    }
}
}

```

Vista

Columna valor por defecto (predeterminado)

Vista tabla

tipo	IdPregunta	etiqueta	predeterminado
FechaHora	hoy		
Fecha	Nueva_fecha	Preguntas fecha?	15/10/2016

Salida

\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```

Clase ejemplo {
    Principal () {
        Nuevo Ejemplo ();
    }
    Formulario Ejemplo () {
        Respuesta res;
        Hoy ().respuesta (res.esFechaHora).FechaHora ( );
        Nueva_fecha ().respuesta (res.esFecha).Fecha ( );
    }
    Pregunta Nueva_fecha ( ) {
        cadena etiqueta = "Preguntas fecha?"
        fecha respuesta = "15/10/2016";
        publico respuesta (fecha param_1) {
            respuesta = param_1;
        }
    }
    Pregunta hoy ( ) {
        fechaHora respuesta;
        publico respuesta (fechaHora param_1) {
            respuesta = param_1;
        }
    }
}

```

Vista

Columna condición aplicable (Aplicable)

Vista tabla

tipo	IdPregunta	etiqueta	aplicable
Texto	Nombre	Cuál es tu nombre?	
Texto	Nombre_comun	Tu nombre es común?	#[Nombre] = "Jose"
Texto	Nombre_no_comun	Tu nombre no es común?	

Salida

```

Clase ejemplo {
    Principal () {
        Nuevo Ejemplo ();
    }
    Pregunta Nombre () {
        cadena etiqueta = "Cuál es tu nombre?"
        cadena respuesta;
        publico respuesta (cadena param_1) {
            respuesta = param_1;
        }
    }
    Pregunta Nombre_comun () {
        cadena etiqueta = "Tu nombre es común?"
        cadena respuesta;
        publico respuesta (cadena param_1) {
            respuesta = param_1;
        }
    }
    Pregunta Nombre_no_comun () {
        cadena etiqueta = "Tu no es común?"
        cadena respuesta;
        publico respuesta (cadena param_1) {
            respuesta = param_1;
        }
    }
    Formulario Ejemplo () {
        Respuesta resp;
        Nombre( ).respuesta(resp.EsCadena).cadena ();
        Si (Nombre( ).respuesta == "Jose") {
            Nombre_comun( ).respuesta(resp.esCadena).cadena ( );
        }
        Nombre_no_comun( ).respuesta(resp. esCadena).cadena ( );
    }
}

```

Columna solo de lectura (Lectura)

Vista tabla

tipo	IdPregunta	etiqueta	predeterminado	lectura
entero	edad	Cuanto años tienes?	18	verdadero

Salida

\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```

Clase ejemplo {
    Principal () {
        Nuevo Ejemplo ();
    }
}

```

```

    }
    Formulario Ejemplo () {
        Respuesta res;
        edad ().respuesta (res.esEntero).entero( );
    }
    $$ método respuesta es privado
    Pregunta edad ( ) {
        cadena etiqueta = "Cuántos años tienes?";
        entero respuesta = 18;
        privado respuesta (entero param_1) {
            respuesta = param_1;
        }
    }
}

```

Vista

Columna de cálculo (Calculo)

Vista tabla

tipo	IdPregunta	etiqueta	cálculo
decimal	cantidad	¿Cuál fue el precio de la comida?	
calcular	propina		#[cantidad] * 0.18
Nota	monitor	18% de propina para su comida es: #[propina]	

Salida

```

$$ el ejemplo mostrará solo donde afecta a las etiquetas
Clase ejemplo {
    Principal () {
        Nuevo Ejemplo ();
    }
    Pregunta cantidad ( ) {
        cadena etiqueta = "¿Cuál fue el precio de la comida?";
        decimal respuesta;
        publico respuesta (decimal param_1) {
            respuesta = param_1;
        }
    }
    $* No existe variable etiqueta, ni método respuesta
    Se deberá crear todos los parámetros, a utilizar en el cálculo *$
    Pregunta propina ( ) {
        decimal respuesta;
        publico calcular (decimal cantidad) {
            respuesta = cantidad * 0.18;
        }
    }
    Pregunta monitor (decimal propina) {

```

```

        cadena etiqueta = "18% de propina para su comida es:" + propina;
        decimal respuesta;
        publico respuesta () {
        }
    }
    Formulario Ejemplo ( ) {
        Respuesta resp;
        cantidad (). respuesta(resp.esEntero).entero ( );
        propina (). calcular(cantidad ().respuesta);
        monitor(propina.respuesta).nota ( );
    }
}

```

Columna número de repeticiones (repetición)

Vista tabla

tipo	IdPregunta	etiqueta	Repetición
entero	cantidad	¿Cuántas repeticiones desea?	
Iniciar Repetición	Valor		#[cantidad]
Nota	Actual	El valor actual es---> #[valor]	
Finalizar repetición	Valor		

Salida

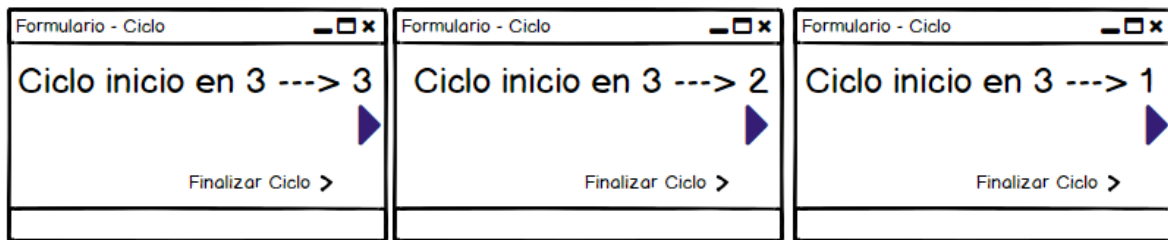
\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```

Clase ejemplo {
    Principal () {
        Nuevo Ejemplo ();
    }
    Pregunta cantidad ( ) {
        cadena etiqueta = "¿Cuántas repeticiones desea?";
        entero respuesta;
        publico respuesta (entero param_1) {
            respuesta = param_1;
        }
    }
    Pregunta Actual (entero valor) {
        cadena etiqueta = "El valor actual es--->" + valor;
        publico respuesta () {
        }
    }
}
Formulario repeticiones ( ) {
    Respuesta resp;
    cantidad (). respuesta(resp.esEntero).entero ( );
    para (entero valor; valor < cantidad (). respuesta; valor ++ ) {
        Actual (cantidad (). respuesta).Nota ( );
    }
}
}

```

Vista



Columna de multimedia (Multimedia)

Vista tabla

tipo	IdPregunta	etiqueta	multimedia
Nota	Imagen	Mira la imagen	Media_imagen "/ruta/imagen.png"
Nota	video	Mira el Video	Media_video reproduccion = true "/ruta/video.png"
Nota	audio	Escucha el Audio	Media_audio "/ruta/imagen.png" reproduccion = true
Nota	audio2	Escucha el Audio	Media_audio "/ruta/imagen.png"

Salida

\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```
Clase ejemplo {
    Principal () {
        Nuevo Ejemplo ();
    }
    Pregunta imagen () {
        cadena etiqueta = "Mira la imagen";
        cadena ruta = "/ruta/imagen.png";
        publico Mostrar () {
            Imagen("/ruta/imagen.png", verdadero);
        }
    }
    Pregunta video () {
        cadena etiqueta = "Mira la video";
        cadena ruta = "/ruta/video.png";
        publico Mostrar () {
            Video("/ruta/video.mp4", verdadero);
        }
    }
    Pregunta audio () {
        cadena etiqueta = "Escucha el audio";
        cadena ruta = "/ruta/audio.mp3";
        publico Mostrar () {
            Audio("/ruta/audio.mp3", verdadero);
        }
    }
    Pregunta audio2 () {
        cadena etiqueta = "Escucha el audio";
        cadena ruta = "/ruta/audio2.mp3";
        publico Mostrar () {
            Audio("/ruta/audio2.mp3", falso);
        }
    }
}
```

```

Formulario Ejemplo ( ) {
    Imagen ( ). Mostrar ( ). imagen ( );
    Video ( ). Mostrar ( ). Video ( );
    Audio ( ). Mostrar ( ). Audio ( );
    Audio2 ( ). Mostrar ( ). Audio ( );
}

```

Salida



Columna de apariencia (apariencia)

Vista tabla

tipo	IdPregunta	etiqueta	Apariencia
entero	edad	Cuanto años tienes?	Cadena
cadena	valor	Cuanto años tienes?	Entero
cadena	anio	Qué año es?	C4

Salida

\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```

Clase ejemplo {
    Principal ( ) {
        Nuevo apariencias ();
    }
    Pregunta edad ( ) {
        cadena etiqueta = "Cuántos años tienes?";
        entero respuesta;
        publico respuesta (entero param_1) {
            respuesta = param_1;
        }
    }
    Pregunta valor ( ) {
        cadena etiqueta = "Cuántos años tienes?";
        entero respuesta;
        publico respuesta (entero param_1) {
            respuesta = param_1;
        }
    }
    Pregunta anio ( ) {
        cadena etiqueta = "Que año es?";
        entero respuesta;
        publico respuesta (entero param_1) {
            respuesta = param_1;
        }
    }
}

```



```

Formulario apariencias ( ){
    Respuesta res;
    Edad ( ). Respuesta (res.esEntero) . apariencia ( ). cadena ( );
    valor ( ). Respuesta (res.esCadena) . apariencia ( ). entero ( );
    $$ejemplo de estilo, Celdas 3
    valor ( ). Respuesta (res.esCadena). apariencia ("C3");
}

```

Vista

Columna de parámetros (Parámetro)

Vista Tabla

tipo	idPregunta	etiqueta	parametro
rango	cuantos	Cuanto hijos tienes	Iniciar=0 finalizar=10

Salida

\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```

Clase ejemplo {
    Principal () {
        Nuevo parametros ();
    }
    Pregunta cuantos ( ) {
        cadena etiqueta = "Cuántos hijos tienes?";
        entero respuesta;
        publico respuesta (entero param_1) {
            respuesta = param_1;
        }
    }
    Formulario parámetros ( ){
        Respuesta res;
        Cuantos ( ). Respuesta (res.esEntero). rango(0, 10);
    }
}

```

Vista

6.2 Hoja de Configuración

Se definirá un ejemplo donde se dé el funcionamiento de la hoja de configuraciones.

Ejemplo

Titulo_formulario	idForm	Estilo	importar	codigo_principal	codigo_global
	impor_arbol		importar Arbol.xform		
	impor_nodo		importar Nodo.xform		
	codigo			abo1 = nuevo Arbol(); abo2 = nuevo Arbol();	Arbol abo1; Arbol abo2;
Arbol Binario	Abb	Todo			

Salida

```

$$Se hizo las importaciones de los dos codigos
Importar Arbol.xform
Importar Nodo.xform
Clase Abb {
    $$Definición de variables globales
    Arbol abo1;
    Arbol abo2;
    Principal () {
        $$Codigo agregado en principal antes de ejecutar el formulario
        abo1 = nuevo Arbol();
        abo2 = nuevo Arbol();
        ...
        Nuevo Abb ( ).Todo; $$Ejecutará el Formulario con estilo grupo
    }
    Formulario Abb ( ) {
        ....
    }
}

```

6.2.1 Estilos

Se definirá ejemplos sobre los estilos o vista de los formularios, estas serán configuradas en la hoja de opciones.

Pagina

Ejemplo

Titulo_formulario	idForm	Estilo
Precio_Comida	pC	Pagina

Salida

```

Clase AX {
    Pregunta A ( ){
        ...
    }
    Pregunta B ( ){
        ...
    }
}

```

```

}
Pregunta C ( ){
...
}
Principal () {
    Nuevo Abb ( ).Pagina; $$Ejecutará el Formulario con estilo grupo
}
Formulario Abb ( ) {
    Respuesta res;
    A ().respuesta(res.esCadena).cadena();
    B ().respuesta(res.esCadena).cadena();
    C ().respuesta(res.esCadena).cadena();
}$$mostrara pregunta por pregunta
}

```

Vista

Todo

Este estilo mostrará todas las preguntas en una sola página del formulario y podrán ser contestar todas las preguntas a la vez. Se deberán de validar las preguntas ocultas o las que se contestan si se cumple una condición, estas deberán de aparecer una vez se haya cumplido dicha condición en la pregunta.

Ejemplo

Titulo_formulario	idForm	Estilo
Precio_Comida	pC	Todo

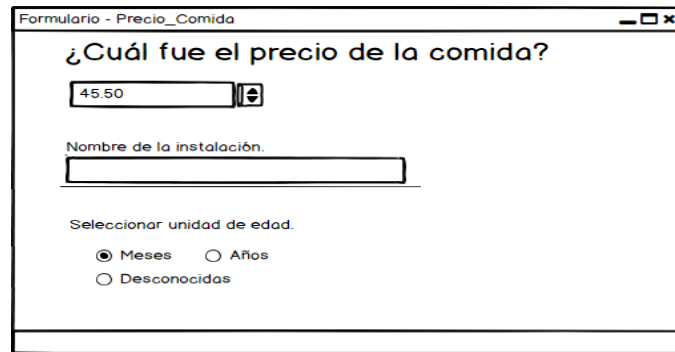
Salida

```

Clase AX {
    Grupo AB ( ) {
        Respuesta res;
        A ().respuesta(res.esEntero).entero();
        B ().respuesta(res.esCadena).cadena();
        C ().respuesta(res.esCadena).cadena();
    }
    Pregunta A ( ) { ... }
    Pregunta B ( ) { ... }
    Pregunta C ( ) { ... }
    Principal () {
        Nuevo Abb ( ).Todo; $$Ejecutará el Formulario con estilo grupo
    }
    Formulario Abb ( ) {
        AB ( );
    }
}

```

Vista



Cuadrícula

Vista Tabla

tipo	IdPregunta	etiqueta	parametro	Apariencia
iniciar agrupación	AB	
...	A	C4
...	B	C1
...	C	C1
...	D	C1
Finalizar agrupación				
iniciar agrupación	DD	
...	E	C2
...	F	C2
Finalizar agrupación				

Vista tabla (Hoja de Configuración)

Titulo_formulario	idForm	Estilo
Precio_Comida	pC	Cuadrícula

Salida

```
Clase AX {
  Grupo AB () {
    Respuesta res;
    A ().respuesta(res.esCadena).apariencia ("C4").Cadena ();
    B ().respuesta(res.esCadena).apariencia ("C1").Cadena ();
    C ().respuesta(res.esCadena).apariencia ("C1").cadena ();
    D ().respuesta(res.esCadena).apariencia ("C1").cadena ();
  }
  Grupo DD () {
    Respuesta res;
    E ().respuesta(res.esCadena).apariencia ("C2").cadena ();
    F ().respuesta(res.esCadena).apariencia ("C2").cadena ();
  }
  Pregunta A () { ... }
  Pregunta B () { ... }
  Pregunta C () { ... }
  Pregunta D () { ... }
  Pregunta E () { ... }
  Pregunta F () { ... }
  Principal () {
```

```

    Nuevo Abb ( ). Cuadrícula; $$Ejecutará el Formulario con estilo grupo
}
Formulario Abb ( ) {
    AB ( );
    DD ( );
}
}

```

Vista

Grupo AB

Formulario Completo		
1.1) Nombre de la instalación. <input type="text"/>		
1.1) Edad difunta. <input type="text" value="3"/>	Seleccionar unidad de edad. <input checked="" type="radio"/> Meses <input type="radio"/> Años <input type="radio"/> Desconocidas	1.3) Sexo difunta. Masculino <input type="checkbox"/> Hembra <input type="checkbox"/>

Grupo DD

2.1) Fecha de la lesión. yyyy/mm/dd	2.1) Fecha de la muerte. yyyy/mm/dd
<input type="text"/>	
<input type="text"/>	

6.3 Tipos de Preguntas

Se definirán los ejemplos básicos de los tipos de preguntas, mostrando su entrada y salida.

Tipo Texto (Texto)

Vista Tabla

tipo	IdPregunta	etiqueta	parametro
texto	capital	Capital de Guatemala?	cad_max = 20 cad_fila = 10
texto	Capital2	Capital de Guatemala?	cad_max = 20

Salida

```

$$ el ejemplo mostrará solo donde afecta a las etiquetas
Clase ejemplo {
    Principal () {
        Nuevo Ejemplo ();
    }
    Pregunta capital ( ) {
        cadena etiqueta = "Capital de Guatemala?";
        cadena respuesta = Nulo;
        publico respuesta (cadena param_1) {
            respuesta = param_1;
        }
    }
}

```

```

    }
    Pregunta capital2 ( ) {
        cadena etiqueta = "Capital de Guatemala?";
        cadena respuesta = Nulo;
        publico respuesta (cadena param_1) {
            respuesta = param_1;
        }
    }
    Formulario ejemplo ( ){
        Respuesta res;
        $$ Parámetros min, max, fila
        capital ( ). Respuesta (res. Escadena). cadena (Nada, 20, 10);
        $$ No se especificó ningún parámetro
        Capital2 ( ). Respuesta (res. EsCadena). cadena ( );
    }
}

```

Salida

Tipo Entero (Entero)

Vista Tabla

tipo	IdPregunta	etiqueta
Entero	cantidad	Cuantos países conoces?

Salida

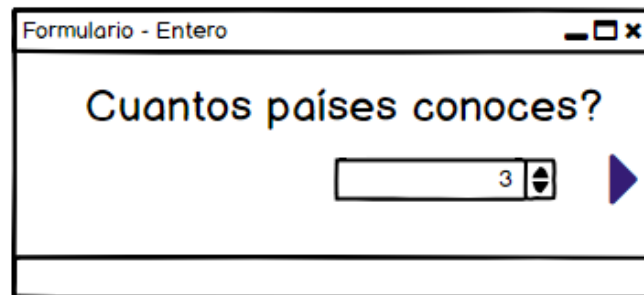
\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```

Clase ejemplo {
    Principal () {
        Nuevo Ejemplo ();
    }
    Pregunta cantidad ( ) {
        cadena etiqueta = "Cuantos países conoces?";
        entero respuesta;
        publico respuesta (entero param_1) {
            respuesta = param_1;
        }
    }
    Formulario ejemplo ( ){
        Respuesta res;
        $$ Pregunta tipo entero
        cantidad ( ). Respuesta (res.EsEntero). Entero ( );
    }
}

```

Salida



Tipo Decimal (Decimal)

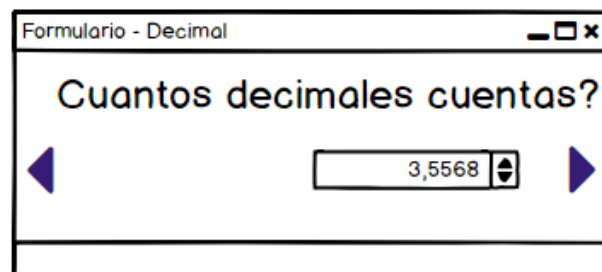
Vista Tabla

tipo	IdPregunta	etiqueta
decimal	cantidad	Cuantos decimales cuentas?

Salida

```
$$ el ejemplo mostrará solo donde afecta a las etiquetas
Clase ejemplo {
    Principal () {
        Nuevo Ejemplo ();
    }
    Pregunta cantidad ( ) {
        cadena etiqueta = "Cuantos decimales cuentas?";
        decimal respuesta;
        publico respuesta (decimal param_1) {
            respuesta = param_1;
        }
    }
    Formulario texto_ejemplo ( ){
        Respuesta res;
        $$ Pregunta tipo Decimal
        cantidad ( ). Respuesta (res.EsDecimal). Decimal ( );
    }
}
```

Salida



Tipo Rango (Rango)

Vista Tabla

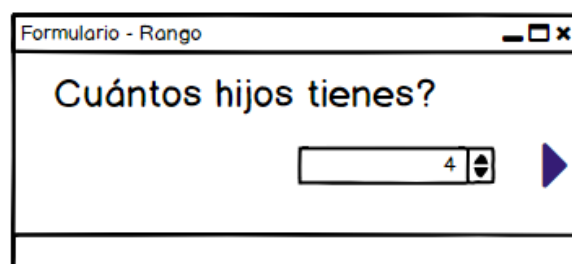
tipo	idPregunta	etiqueta	parametro
rango	cuantos	Cuanto hijos tienes	finalizar=10

Salida

\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```
Clase ejemplo {
    Principal () {
        Nuevo Ejemplo ();
    }
    Pregunta cuantos ( ) {
        cadena etiqueta = "Cuántos hijos tienes?";
        entero respuesta;
        publico respuesta (entero param_1) {
            respuesta = param_1;
        }
    }
    Formulario parámetros ( ){
        Respuesta res;
        $$Parámetro no definido, inicio en 0
        Cuantos ( ). Respuesta (res.esEntero). rango (0, 10);
    }
}
```

Salida



Tipo Condicional (Condicion)

Vista Tabla

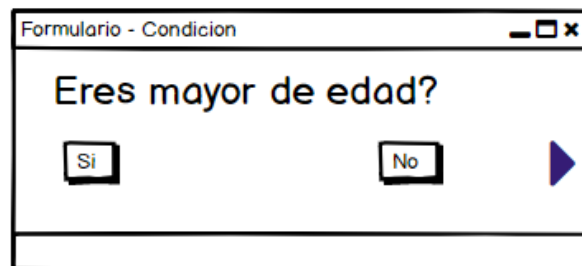
tipo	idPregunta	etiqueta	parametro
condicion	mayor	Eres mayor de edad?	Opción=V_F

Salida

\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```
Clase ejemplo {
    Principal () {
        Nuevo parámetros ();
    }
    Pregunta mayor ( ) {
        cadena etiqueta = "Eres mayor de edad?";
        booleano respuesta;
        publico respuesta (booleano param_1) {
            respuesta = param_1;
        }
    }
    Formulario parámetros ( ){
        Respuesta res;
        mayor ( ). Respuesta (res.esBooleano). condición('Verdadero', 'Falso');
    }
}
```


Salida



Tipo Fecha (Fecha)

Vista Tabla

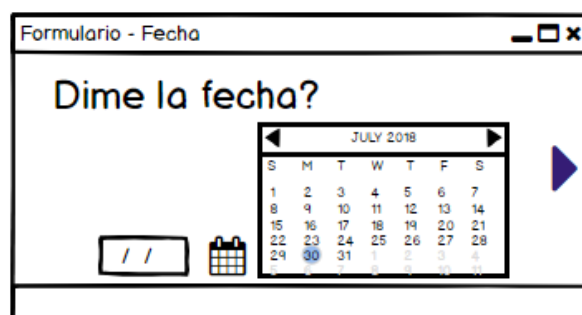
tipo	idPregunta	etiqueta
fecha	mayor	Dime la fecha?

Salida

\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```
Clase ejemplo {
    Principal () {
        Nuevo parámetros ();
    }
    Pregunta mayor () {
        cadena etiqueta = "Dime la Fecha?";
        fecha respuesta;
        publico respuesta (fecha param_1) {
            respuesta = param_1;
        }
    }
    Formulario parámetros () {
        Respuesta res;
        Mayor ( ). Respuesta (res.esFecha). fecha ( );
    }
}
```

Salida



Tipo Tiempo (Hora)

Ejemplo Vista Tabla

tipo	idPregunta	etiqueta
hora	mayor	Dime la hora?

Salida

\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```

Clase ejemplo {
    Principal () {
        Nuevo parámetros ();
    }
    Pregunta mayor ( ) {
        cadena etiqueta = "Dime la hora?";
        hora respuesta;
        publico respuesta (hora param_1) {
            respuesta = param_1;
        }
    }
    Formulario parámetros ( ){
        Respuesta res;
        mayor ( ). Respuesta (res.esFecha). hora ( );
    }
}

```

Vista

Pregunta

1) Hora de ingreso con segundos?

hh mm ss

Aceptar

Pregunta

1) Hora de ingreso?

hh mm

Aceptar

Tipo Fecha y Hora (FechaHora)

Vista Tabla

tipo	idPregunta	etiqueta
hora	mayor	Dime la fecha y hora?

Salida

\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```

Clase ejemplo {
    Principal () {
        Nuevo parámetros ();
    }
    Pregunta mayor ( ) {
        cadena etiqueta = "Dime la fecha y hora?";
        fechahora respuesta;
        publico respuesta (fechahora param_1) {
            respuesta = param_1;
        }
    }
    Formulario parámetros ( ){
        Respuesta res;
        Mayor ( ). Respuesta (res.esFecha). fechaHora ( );
    }
}

```

Vista



Tipo Respuesta de Selección

Vista Tabla

tipo	idPregunta	etiqueta	Parámetro
Selecciona_uno generos	Genero	Dime un género?	
Selecciona_multiples general	numeros	Elige los numeros	

Vista Tabla Listas (Hoja de Opciones)

Nombre_lista	nombre	Etiqueta	multimedia
fechas	mes	Meses	
fechas	anio	Año	
fechas	otro	Desconocido	
fruta	man	Manzana	/ruta/relativa/a.jpg
fruta	pe	Pera	/ruta/relativa/b.jpg
fruta	ban	Banano	/ruta/relativa/c.jpg

Salida

\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```

Clase ejemplo {
    Principal () {
        Nuevo parámetros ();
    }
    Pregunta géneros () {
        cadena etiqueta = "Selecciona una unidad?";
        Cadena respuesta;
        publico respuesta (Cadena param_1) {
            respuesta = param_1;
        }
    }
    Pregunta números () {
        cadena etiqueta = "Frutas Favoritas";
        Entero respuesta;
        publico respuesta (Entero param_1) {
            respuesta = param_1;
        }
    }
    Formulario parámetros () {

```

```

    $$la lista al agregar (key, valor, ruta)
    $$la lista al agregar (key, valor)
    Opciones fechas = Nuevo Opciones ( );
    fechas.agregar ("mes", "Meses");
    fechas.agregar ("anio", "Años");
    fechas.agregar ("otro", "Desconocido");
    Opciones frutas = Nuevo general ( );
    frutas.agregar ("ma", "Manzana", "/ruta/relativa/a.jpg" );
    frutas.agregar ("pe", "Pera", "/ruta/relativa/a.jpg");
    frutas.agregar ("ba", "Banano", "/ruta/relativa/a.jpg");
    Respuesta res;
    Genero ( ). Respuesta (res.esCadena). Seleccionar_1 (generos);
    numeros ( ). Respuesta (res.esCadena). Seleccionar (general);
}

```

Vista Selección uno

Vista Selección Múltiple

Tipo Nota (Nota)

Vista tabla

tipo	IdPregunta	etiqueta	cálculo
Nota	Mi_nota	Soy una Nota	

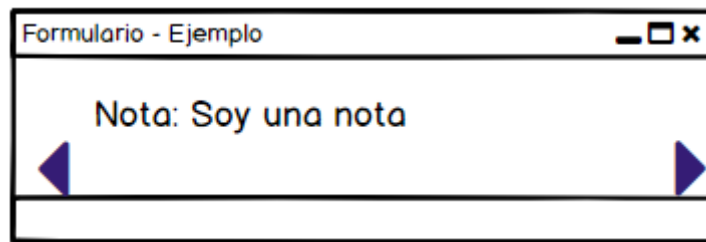
Salida

```

$$ el ejemplo mostrará solo donde afecta a las etiquetas
Clase ejemplo {
    Principal () {
        Nuevo ejemplo ();
    }
    Pregunta mi_nota () {
        cadena etiqueta = "Soy una nota";
        publico respuesta () {
        }
    }
    Formulario Ejemplo () {
        Respuesta resp;
        Mi_nota ( ).nota( );
    }
}

```

Salida



Tipo Multimedia

Vista tabla

tipo	IdPregunta	etiqueta
Fichero “.png, .jpg, .gif”	Mi_imagen	Sube una Imagen
Fichero “.mp3”	Mi_Audio	Sube un audio
Fichero	Mi_fichero	Sube cualquier archivo

Salida

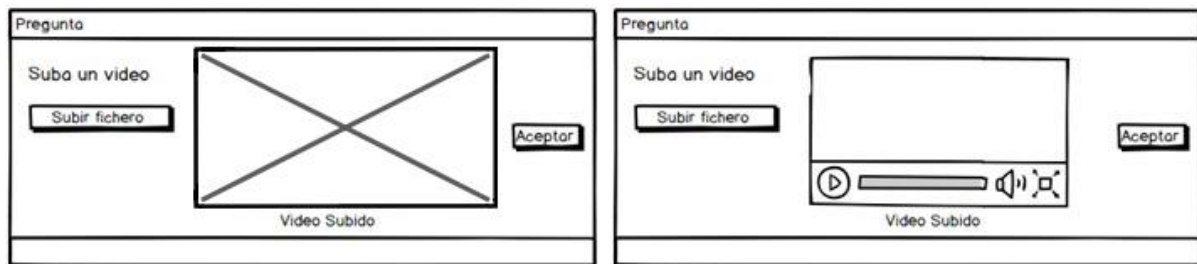
\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```

Clase ejemplo {
    Principal () {
        Nuevo ejemplo ();
    }
    Pregunta mi_imagen () {
        cadena etiqueta = "Sube una imagen";
        publico respuesta () {
        }
    }
    Pregunta mi_audio () {
        cadena etiqueta = "Sube una imagen";
        publico respuesta () {
        }
    }
    Pregunta mi_fichero () {
        cadena etiqueta = "Sube cualquier archivo";
        publico respuesta () {
        }
    }
    Formulario Ejemplo_fichero () {
        Respuesta resp;
        Mi_imagen ().Fichero(“.png, .jpg, .gif”);
        Mi_Audio ().Fichero(“.mp3”);
        Mi_fichero ().Fichero( );
    }
}

```

Salida



Tipo Calculo (Calcular)

Vista tabla

tipo	IdPregunta	etiqueta	cálculo
calcular	numeros		$150 * 0.18 + 100$
calcular	logicos		Verdadero && Falso && !(falso)
Nota	Ver_resultado	La operación numérica es: #[numeros] La operación lógica es: #[lógicos]	

Salida

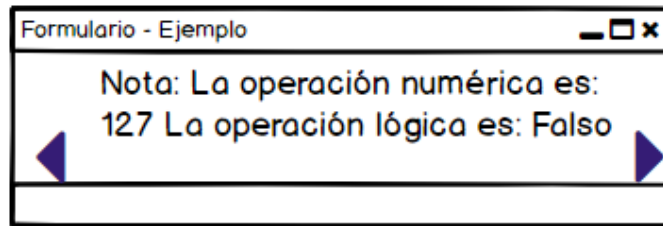
\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```

Clase ejemplo {
    Principal () {
        Nuevo ejemplo ();
    }
    Pregunta números () {
        decimal respuesta;
        publico calcular () {
            respuesta = 150 * 0.18 + 100;
        }
    }
    Pregunta logicos () {
        booleano respuesta;
        publico calcular () {
            respuesta = Verdadero && Falso && !(falso);
        }
    }
    Pregunta Ver_resultado (decimal numeros, booleano logicos) {
        cadena etiqueta = "La operación numérica es:" + #[numeros] + "La operación lógica es:" + #[lógicos];
        publico respuesta () {
        }
    }
    Formulario Ejemplo () {
        Respuesta resp;
        Números ( ). Calcular ();
        Lógicos ( ). Calcular ();
        Ver_resultado (numeros.respuesta, logicos.respuesta).nota( );
    }
}

```

Vista



6.4 Agrupaciones

Se definirá los ejemplos básicos para el uso de las agrupaciones.

Grupos

Vista tabla

tipo	IdPregunta	etiqueta	cálculo
Iniciar agrupación	operarGrupo		
calcular	numeros		150 * 0.18 + 100
calcular	logicos		Verdadero && Falso && !(falso)
Nota	Ver_resultado	La operación numérica es: #[numeros] La operación lógica es: #[lógicos]	
Finalizar agrupación			

Salida XForm

```
$$ el ejemplo mostrará solo donde afecta a las etiquetas
Clase ejemplo {
    Principal () {
        Nuevo ejemplo ();
    }
    Pregunta números () {
        decimal respuesta;
        publico calcular () {
            respuesta = 150 * 0.18 + 100;
        }
    }
    Pregunta logicos () {
        booleano respuesta;
        publico calcular () {
            respuesta = Verdadero && Falso && !(falso);
        }
    }
    Pregunta Ver_resultado (decimal numeros, booleano logicos) {
        cadena etiqueta = "La operación numérica es:" + #[numeros] + "La operación lógica es:" + #[lógicos];
        publico respuesta () {
        }
    }
}
Grupo OperarGrupo {
    Respuesta resp;
```

```

        Números (). calcular();
        Lógicos (). calcular ();
        Ver_resultado (numeros.respuesta, lógicos.respuesta).nota( );
    }
    Formulario Ejemplo ( ) {
        OperarGrupo ( );
    }
}

```

Agrupación anidada de Preguntas

Vista tabla

tipo	IdPregunta	etiqueta	aplicable	cálculo
Iniciar agrupación	Operaciones			
Iniciar agrupación	Gnumeros			
calcular	numeros			150 * 0.18 + 100
Iniciar agrupación	Glogicos		#[numeros] <= 100 &&#[numeros] >= 0.15	
calcular	logicos			Verdadero && Falso && !(falso)
Finalizar agrupación	Glogicos			
Finalizar agrupación	Gnumeros			
Nota	Ver_resultado	La operación numérica es:#[números] La operación lógica es:#[lógicos]		
Finalizar agrupación	Operaciones			

Salida

\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```

Clase ejemplo {
    Principal () {
        Nuevo ejemplo ();
    }
    Pregunta números ( ) {
        decimal respuesta;
        publico calcular () {
            respuesta = 150 * 0.18 + 100;
        }
    }
    Pregunta logicos ( ) {
        booleano respuesta;
    }
}

```



```

        publico calcular () {
            respuesta = Verdadero && Falso && !(falso);
        }
    }
    Pregunta Ver_resultado (decimal numeros, booleano logicos) {
        cadena etiqueta = "La operación numérica es:" + #[numeros] + "La operación lógica
es:" + #[lógicos];
        publico respuesta () {
        }
    }
    Grupo Gnumeros {
        Respuesta resp;
        numeros(). calcular();
    }
    Grupo Glogicos {
        Respuesta resp;
        logicos (). calcular().;
    }
    Grupo Operaciones {
        Gnumeros ( );
        $$Se ingreso al grupo, después a la pregunta
        Si (Gnumeros ( ).numeros ( ). respuesta <= 100 && Gnumeros ( ).numeros ( ).
        respuesta >= 0.15) {
            Glogicos ( );
        }
    }
    Formulario Ejemplo ( ) {
        Operaciones ( );
        Ver_resultado (numeros.respuesta, logicos.respuesta).nota( );
    }
}

```

Ciclos

Vista tabla

tipo	IdPregunta	etiqueta	aplicable	cálculo	repetir
Iniciar ciclo	Operaciones				
Iniciar agrupación	Gnumeros				
calcular	numeros			150 * 0.18 + 100	
Iniciar ciclo	Glogicos		#[numeros] <= 100 && #[numeros] >= 0.15		90 - #[numeros]
calcular	logicos			Verdadero && Falso && !(falso)	
Finalizar ciclo					
Finalizar agrupación					
Nota	Ver_resultado	La operación numérica es: #[numeros]			

		La operación lógica es:#[lógicos]			
Finalizar agrupación					

Salida XForm

\$\$ el ejemplo mostrará solo donde afecta a las etiquetas

```

Clase ejemplo {
    Principal () {
        Nuevo ejemplo ();
    }
    Pregunta números ( ) {
        decimal respuesta;
        publico calcular () {
            respuesta = 150 * 0.18 + 100;
        }
    }
    Pregunta logicos ( ) {
        booleano respuesta;
        publico calcular () {
            respuesta = Verdadero && Falso && !(falso);
        }
    }
    Pregunta Ver_resultado (decimal numeros, booleano logicos) {
        cadena etiqueta = "La operación numérica es:" + #[numeros] + "La operación lógica es: " + #[lógicos];
        publico respuesta () {
        }
    }
    Grupo Gnumeros {
        Respuesta resp;
        numeros(). calcular();
    }
    Grupo Glogicos {
        Respuesta resp;
        logicos (). calcular();
    }
    Grupo Operaciones {
        Gnumeros ( );
        Para (entero Glogicos = 1; Glogicos <= 90 - Gnumeros ( ).numeros ( ). respuesta; Glogicos ++){
            Si (Gnumeros ( ).numeros ( ). respuesta <= 100 && Gnumeros ( ).numeros ( ). respuesta >= 0.15) {
                Glogicos ( );
            }
        }
    }
    Formulario Ejemplo_grupos ( ) {
        Entero Operaciones = 0;
        para (operaciones = 0 ; verdadero; operaciones ++ ) {
            Operaciones ( );
        }
        Ver_resultado (numeros.respuesta, logicos.respuesta).nota( );
    }
}

```

7 Ejemplo para generar un formulario

En el siguiente ejemplo se detallarán los pasos a seguir para la construcción de un formulario, así como su flujo básico a través de ambos módulos de la aplicación Form-USAC, los primeros 2 pasos se desarrollarán con la herramienta de Excel, los pasos del 3 al 5 se realiza con los componentes para la construcción del formulario con el lenguaje XForm y el resto de pasos se llevará a cabo con la Generación y llenado de formulario.

Para mostrar la funcionalidad deseada se plantea un ejemplo en el link, el cual se definen un conjunto de preguntas en XLS y su respectiva salida que serán manipuladas en XForm.

<https://drive.google.com/open?id=1wjMvPCTS10YuFSJfZsh4npMopzkeF4OB>

7.1 Pasos de interpretación del ejemplo

A continuación, se describirá los pasos a seguir para la creación de un formulario:

7.1.1 Paso 1

Para poder realizar un formulario de preguntas, se podrá hacer uso de la herramienta de Excel o una con las mismas características, en ella se deberá colocar las instrucciones para la elaboración de un formulario. El formulario elaborado se deberá de almacenar en formato XLS.

A1																	
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	
1	tipo	IdPregunta	etiqueta	parametro	calcular	aplicable	sugerencia	restringir	restringirm	requeridoMs	requerido	PorDefecto	repeticion	aparencia	codigo_pre	codigo_post	
2	Iniciar Agrupacion	info1	Agrupación 1														
3	texto	nombre	Cual es tu nombre?	cad_min=5			no se puede omitir			Ingrese un nombre	verdadero						
4	fecha	hoyes	Que fecha es hoy?	cad_fila=1								hoy()					
5	Nota	nota_datos	Se llenara dos arboles Binarios														
6	finalizar Agrupacion																
7																	
8	Iniciar Agrupacion	Info2	Agrupación 2														
9	Nota	arbol1	Se inicio la instancia del primer Arbol?													abo1 = nuevo ArbolBinario	
10	Iniciar Ciclo	llenar1	Ciclo Arbol 1														
			Dame un numero @(color: #FFFF00, tam:15, negrilla): { Mayor }@ a 1 y @(color: #FFFF00, tam:15, negrilla): { Menor }@ a 20														
11	entero	insertar1		inicia=1 finaliza= 20							verdadero					abo1.insertar (@);	
12	finalizar ciclo																
13	finalizar Agrupacion																
14																	
15	Iniciar agrupacion	info3	Agrupación 3														
16	Nota	arbol2	Se inicio la instancia del segundo arbol													abo2 = nuevo ArbolBinario	
17	Iniciar Ciclo	llenar2	Ciclo Arbol 2														
			Dame un numero @(color: #FFFF00, tam:15, negrilla): { Mayor }@ a 20 y @(color: #FFFF00, tam:15, negrilla): { Menor }@ a 30														
18	rango	insertar2		inicia=20 finaliza= 30							verdadero					abo2.insertar (@);	
19	finalizar ciclo																
20	finalizar Agrupacion																

encuesta

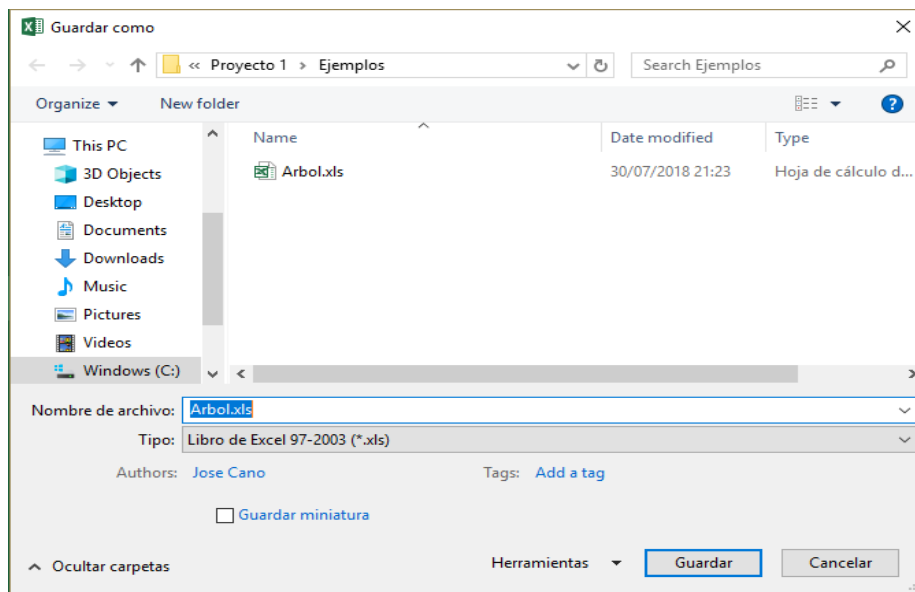
opciones

configuracion

</

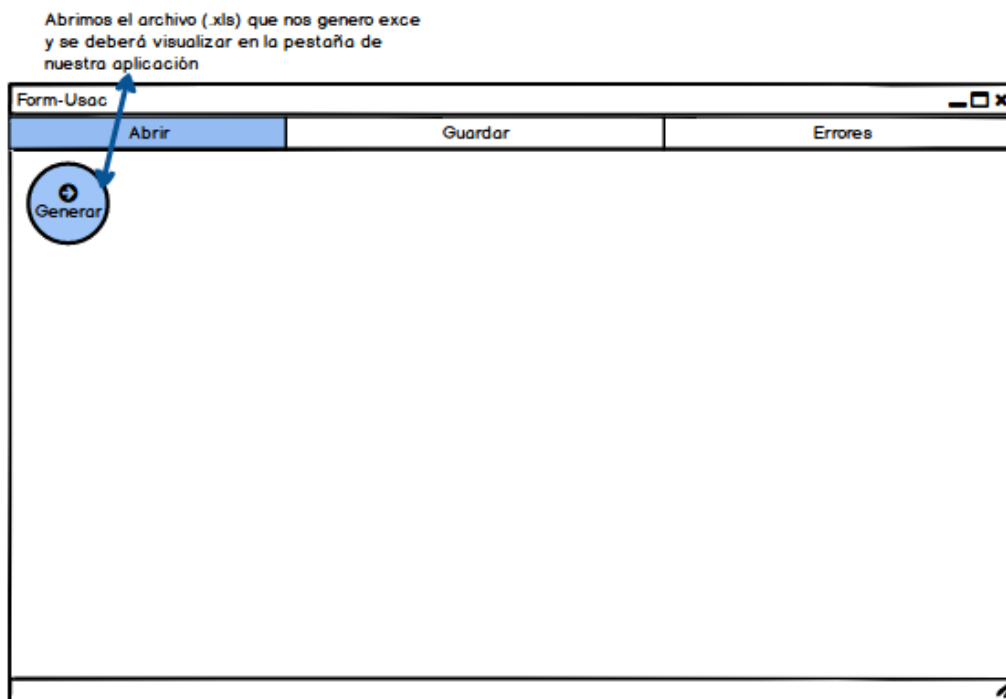
7.1.2 Paso 2

El formulario elaborado en el paso anterior se deberá de almacenar en formato XLS, para poder proceder a su construcción en código de alto nivel.



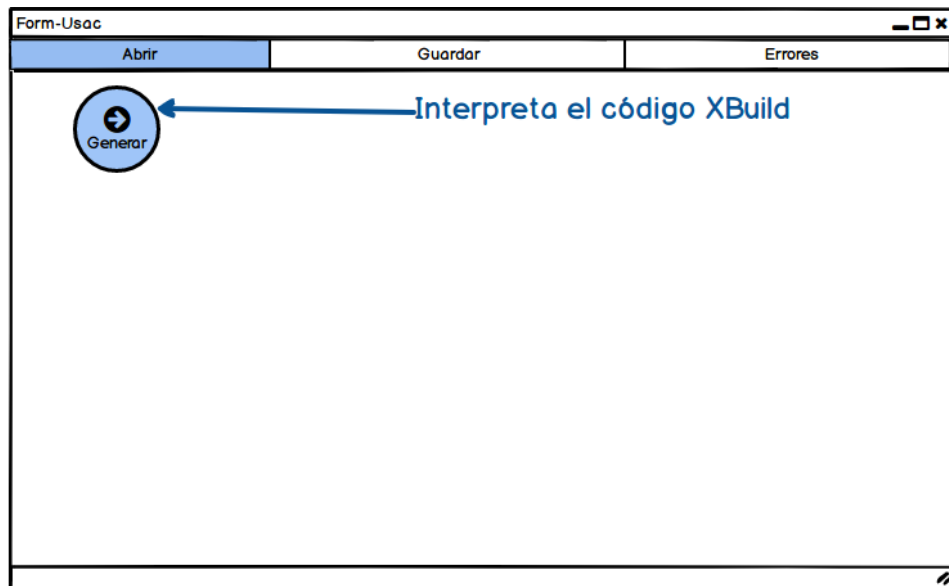
7.1.3 Paso 3

En el componente de construcción del formulario se deberá abrir el archivo XLS y se deberá mostrar en el área de edición.



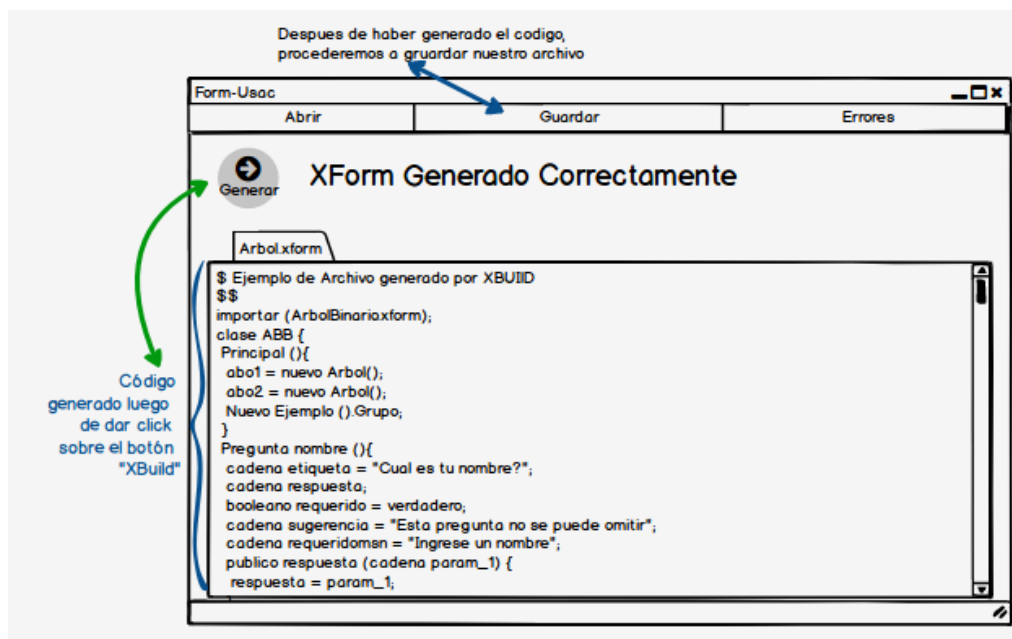
7.1.4 Paso 4

Se agregarán los elementos del formulario en formato XForm. Tras haber verificado los elementos agregados, se procederá a dar clic en el botón Generar, el cual ejecutará la interpretación de la entrada.



7.1.5 Paso 5

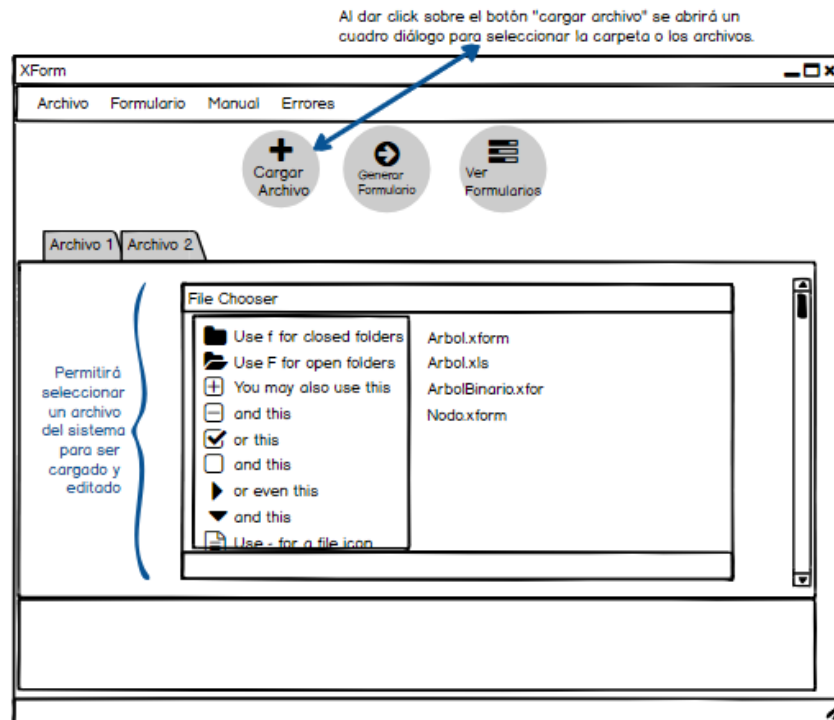
Al haber concluido la interpretación se abrirá una nueva pestaña con la salida generada por el intérprete, esta salida será en formato XForm. Y con el código correctamente generado procederemos almacenar el archivo (xform).



Al tener el código generado se procederá ir al componente de Generación y llenado de formulario.

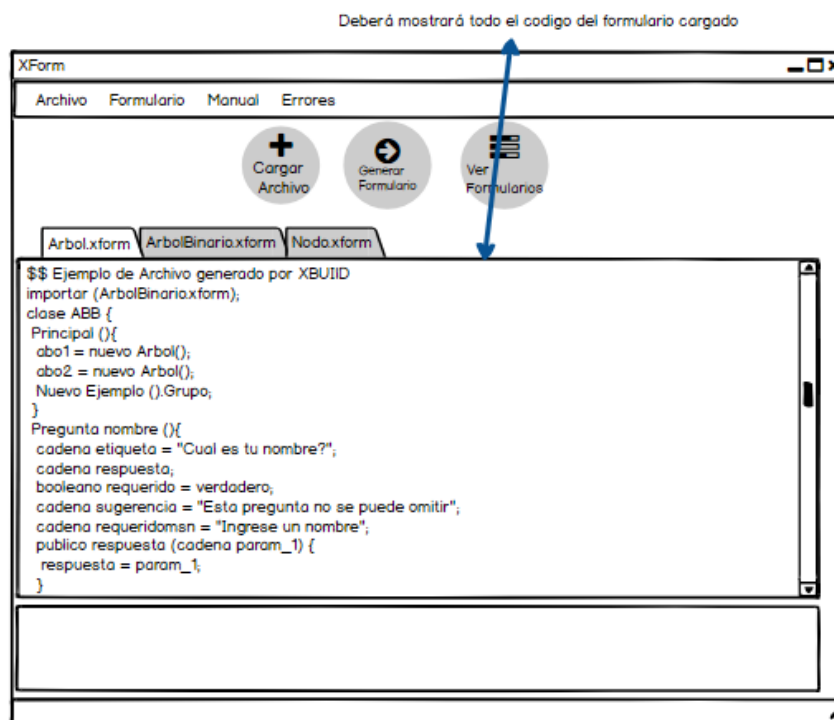
7.1.6 Paso 6

En el componente de generación y llenado de formulario, se procederá a cargar el archivo generado en el paso anterior haciendo clic sobre el botón “Cargar archivo” y se mostrará un cuadro de diálogos donde se deberá seleccionar el archivo (.xform).



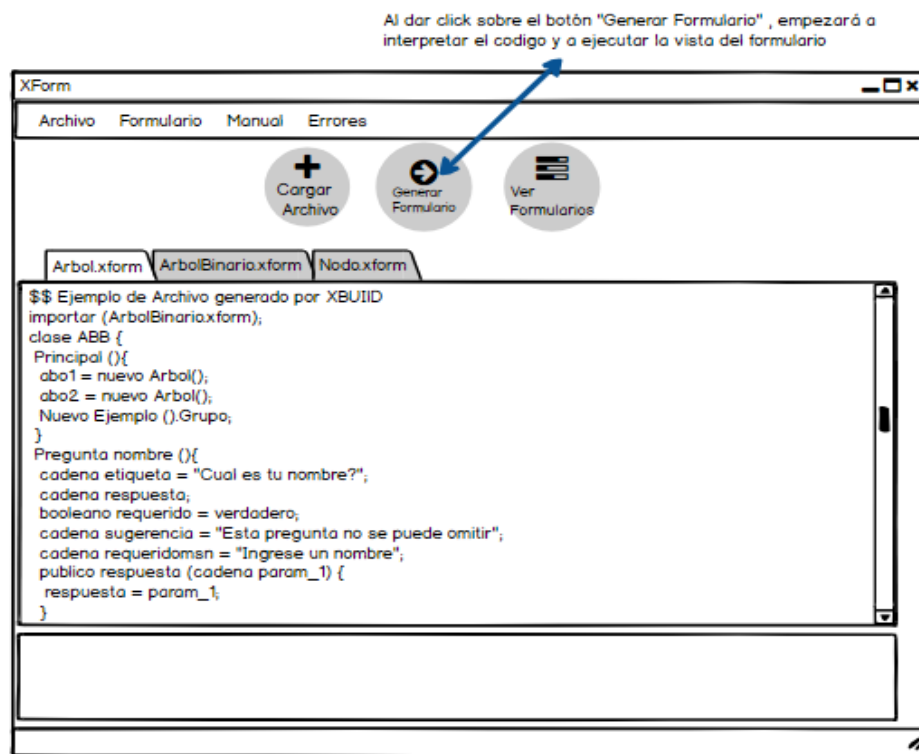
7.1.7 Paso 7

Después de haber seleccionado el archivo, se mostrará el código fuente en el área de edición.



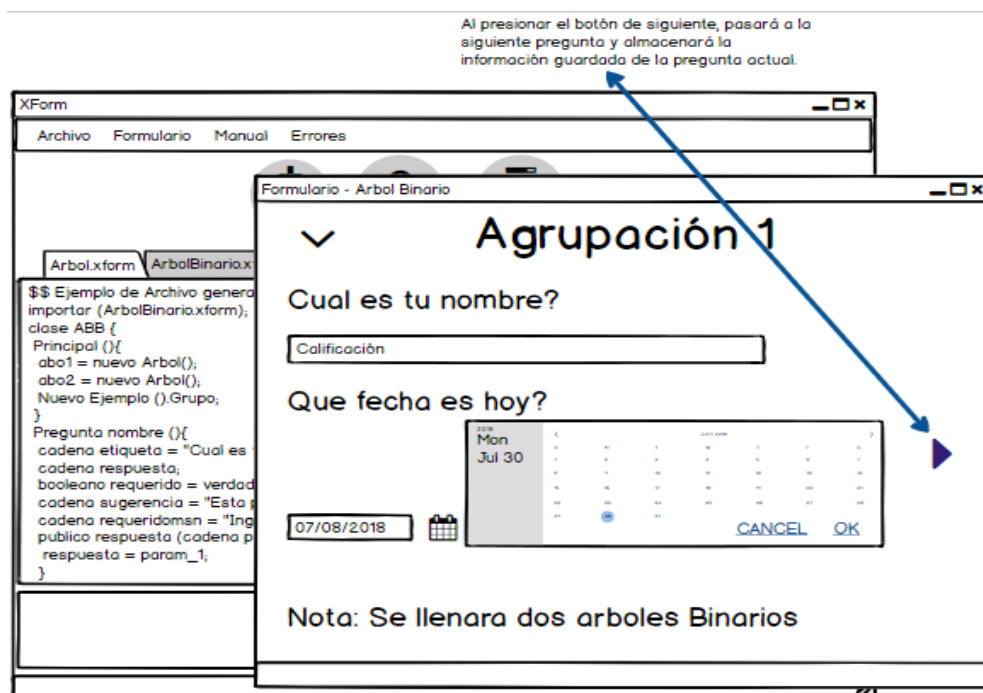
7.1.8 Paso 6

Cuando se dé clic sobre el botón **generar formulario** se interpretará el código que se encuentra en área de edición y se mostrará un formulario como resultado del proceso de interpretación del código XForm.

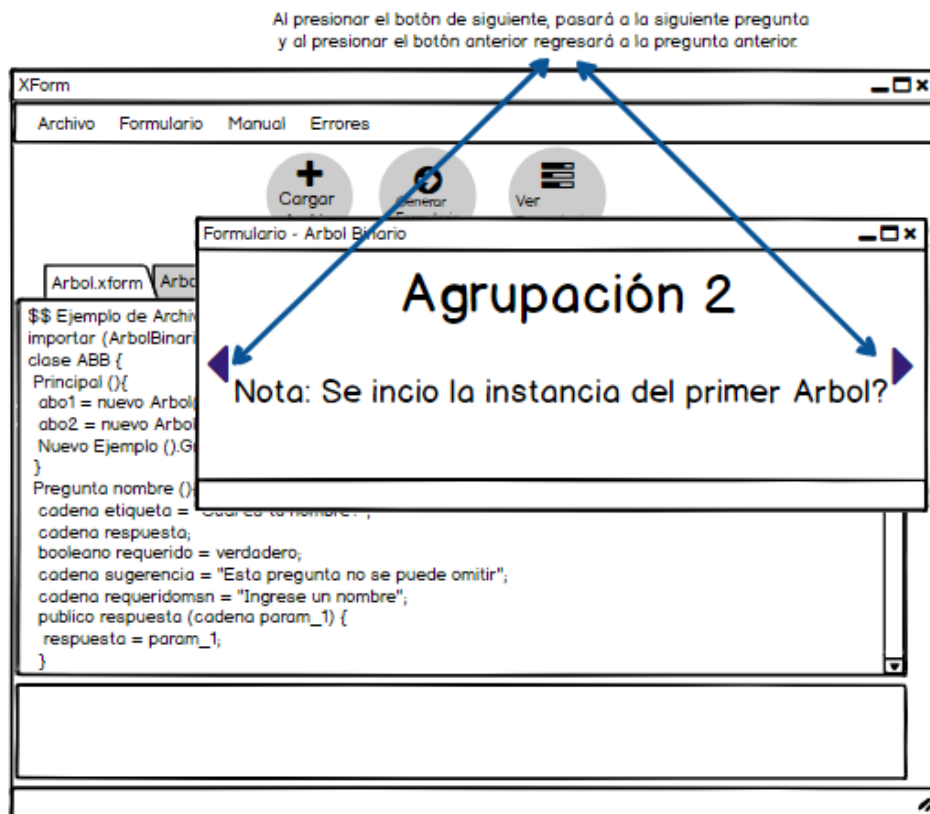


7.1.9 Paso 9

Se deberá mostrar en una nueva ventana el flujo del formulario, para poder proceder a responder el formulario.

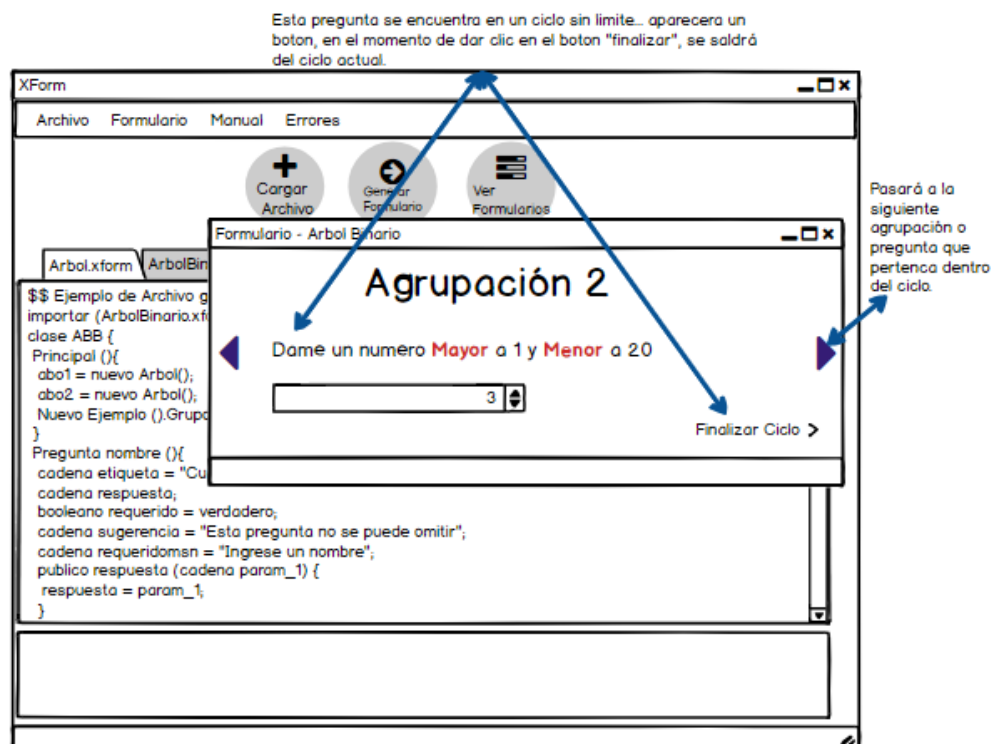


Pasará al siguiente grupo de preguntas del formulario.



7.1.10 Paso 10

Se procederá a ingresar las veces que se desee un número en el formulario. Para salir del ciclo se dará clic en “finalizar Ciclo” y pasará a la siguiente agrupación o preguntas fuera del ciclo.



7.1.11 Paso 11

Al salir del ciclo, mostrará el siguiente set de preguntas.

The screenshot shows the XForm application window. The menu bar includes 'Archivo', 'Formulario', 'Manual', and 'Errores'. The toolbar has three buttons: 'Cargar' (with a plus icon), 'Generar' (with a circular arrow icon), and 'Ver' (with a list icon). A tab labeled 'Arbol.xform' is active. The main content area displays a code editor with the following text:

```
$$ Ejemplo de Archivo g
importar (ArbolBinario.x
class ABB {
Principal (){
abo1 = nuevo Arbol();
abo2 = nuevo Arbol();
Nuevo Ejemplo ().Grup
}
Pregunta nombre (){
cadena etiqueta = "Cu
cadena respuesta;
booleano requerido = verdadero;
cadena sugerencia = "Esta pregunta no se puede omitir";
cadena requeridomsn = "Ingrese un nombre";
publico respuesta (cadena param_1) {
respuesta = param_1;
}
```

Overlaid on the code editor is a dialog box titled 'Formulario - Arbol Binario'. It contains the heading 'Agrupación 3' and the text 'Nota: Se inicio la instancia del segundo arbol?'. The dialog has left and right arrow buttons for navigation.

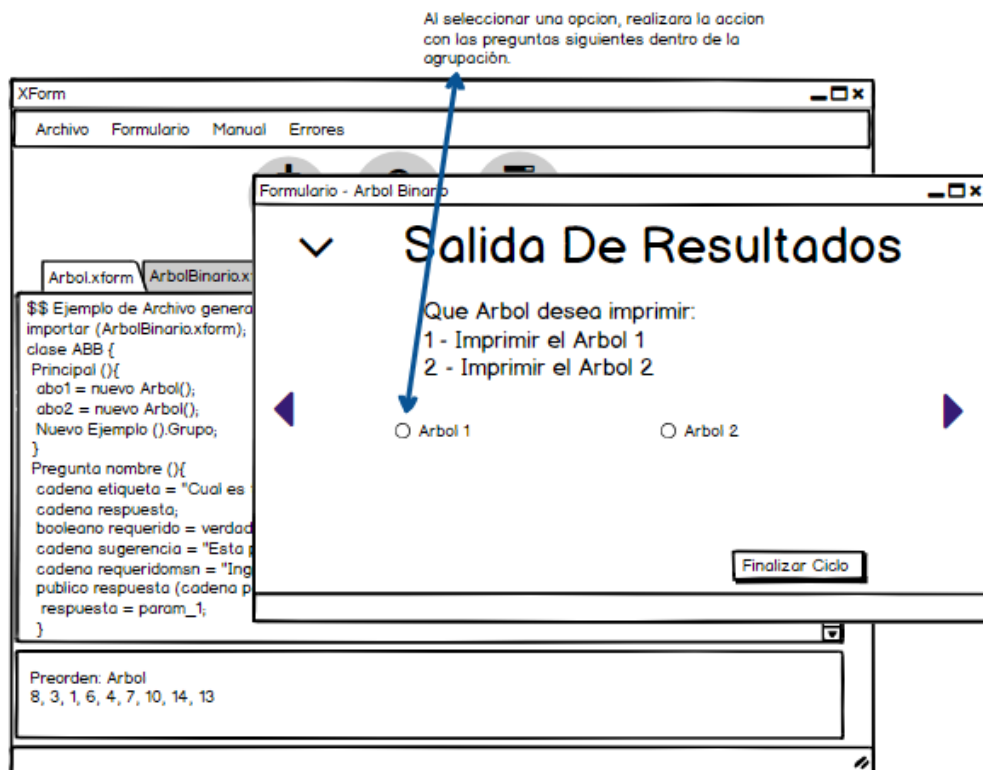
This screenshot shows the XForm application window in a later state. The menu bar and toolbar are the same as in the previous image. The code editor now shows a different portion of the code:

```
$$ Ejemplo de Archivo g
importar (ArbolBinario.x
class ABB {
Principal (){
abo1 = nuevo Arbol();
abo2 = nuevo Arbol();
Nuevo Ejemplo ().Grup
}
Pregunta nombre (){
cadena etiqueta = "Cu
cadena respuesta;
booleano requerido = verdadero;
cadena sugerencia = "Esta pregunta no se puede omitir";
cadena requeridomsn = "Ingrese un nombre";
publico respuesta (cadena param_1) {
respuesta = param_1;
}
```

The dialog box 'Formulario - Arbol Binario' is still present, showing 'Agrupación 3'. The text inside has changed to 'Dame un numero Mayor a 20 y Menor a 30'. Below the text is a text input field containing the number '3'. To the right of the input field is a 'Finalizar Ciclo >' button.

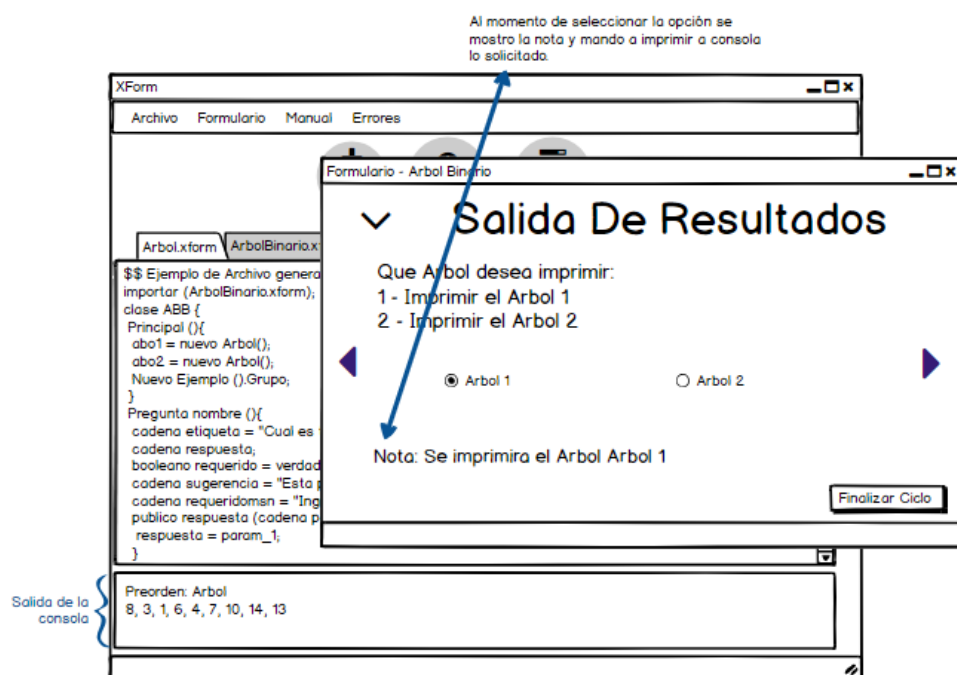
7.1.12 Paso 12

Esta pregunta se encuentra en un ciclo y solicitará al usuario poder seleccionar una de las dos opciones, al momento de aceptar la opción proseguirá a la siguiente pregunta de tipo nota.



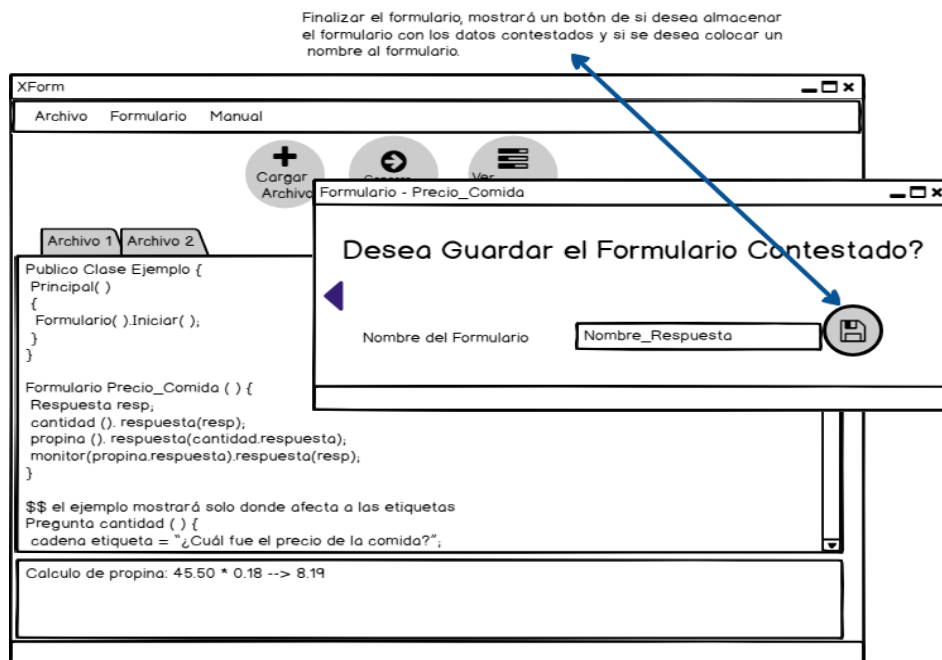
7.1.13 Paso 13

Al seleccionar la opción de “Árbol 1” nos mostrará un mensaje y mandará a imprimir en la consola la información solicitada.



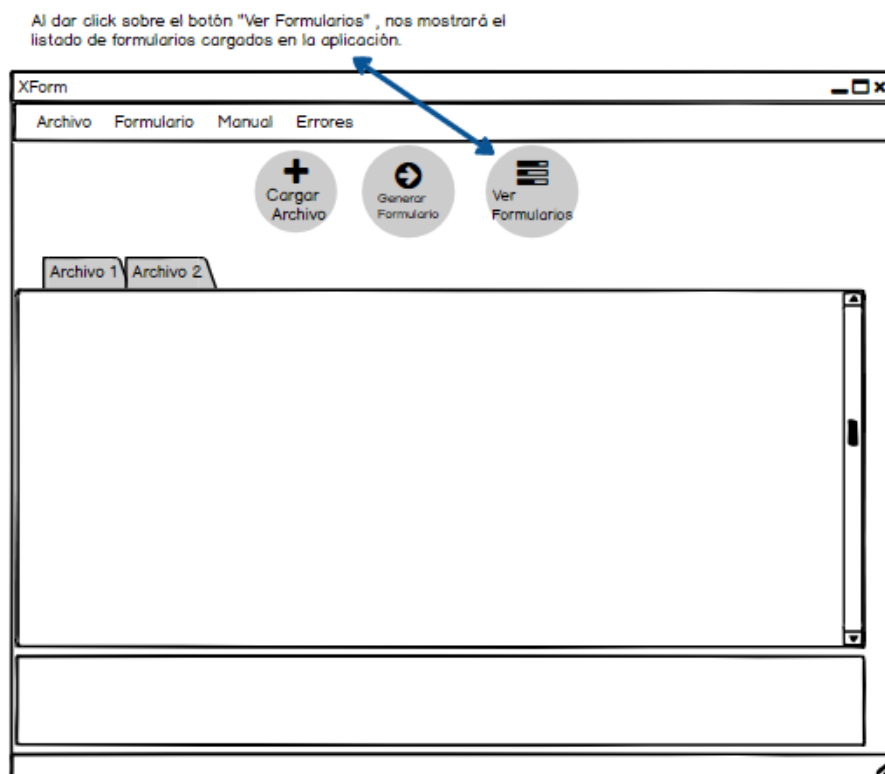
7.1.14 Paso 14

Al terminar de contestar el formulario se procederá almacenar las respuestas, se podrá dar un nombre a las respuestas y si no se desea, se almacenará con un nombre por defecto.



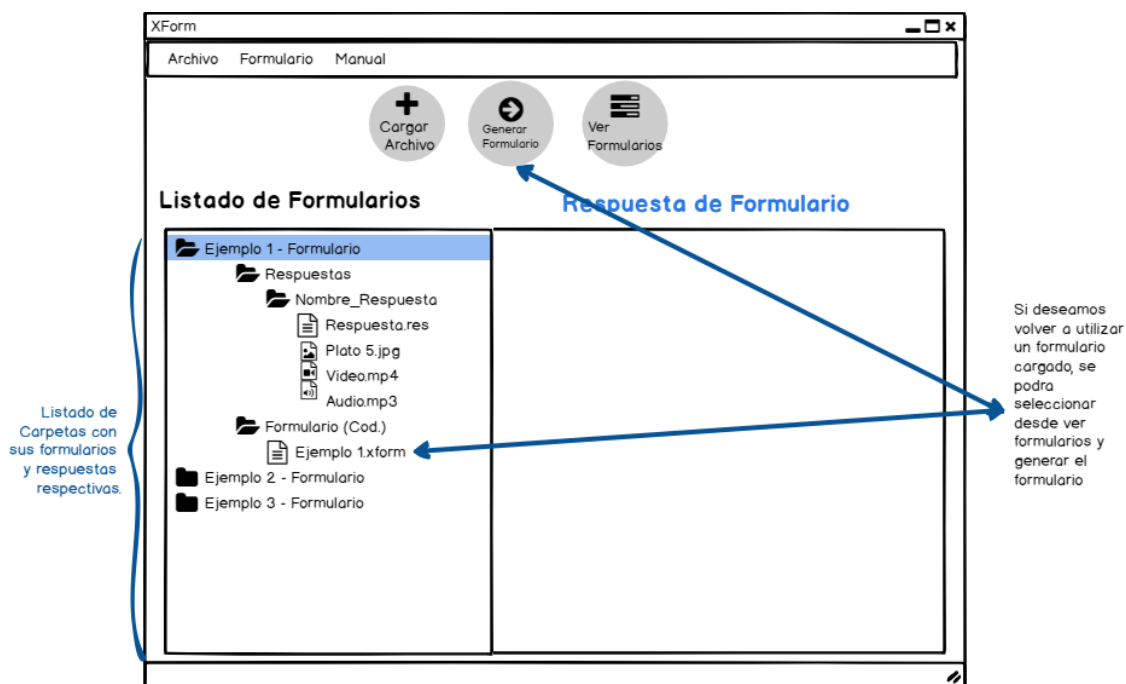
7.1.15 Paso 15

Tras haber generado el formulario en el paso anterior, se generará una carpeta que contendrá los metadatos del mismo. Al presionar el botón "Ver Formulario", nos mostrará el listado de formularios con sus respectivas respuestas.



7.1.16 Paso 16

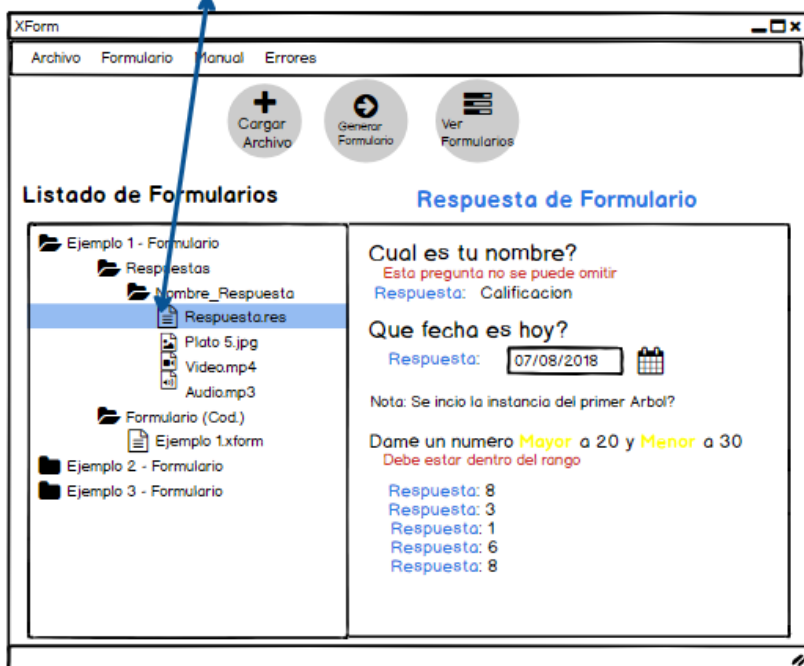
Cada vez que se colecten datos con un formulario se accederá al directorio principal del mismo y generará una nueva carpeta en la cual se encontrarán los archivos multimedia, así como archivos con el contenido de las respuestas.



7.1.17 Paso 17

Se podrá seleccionar el archivo de respuestas que han sido almacenadas. Al dar clic sobre el archivo, se abrirá una nueva ventana con el formulario y sus respectivas respuestas que fueron almacenadas.

Al dar click sobre los respuesta de un formulario, nos mostrará el listado de respuestas almacenados en la aplicación.



8 Manejo de errores

Los componentes: construcción del formulario, generación y llenado del formulario deberán ser capaces de encontrar errores en cada fase del análisis de interpretación, para cada error se indicará la línea, columna, descripción del mismo y el tipo de error que ha ocurrido. Lo tipos de errores que se deberán de manejar son los siguientes:

- Errores léxicos.
- Errores sintácticos.
- Errores semánticos.

8.1 Manejo de errores del lenguaje de alto nivel

Un **error léxico** es aquel en el que se encuentra un carácter que no está permitido como parte de un lexema.

Ejemplo 1:

```
Publico ente$ro numero = 15;
```

En la cadena anterior se introdujo arbitrariamente el componente léxico "\$", esto deberá de provocar un error léxico. La forma en que se recuperará de un error léxico es con la estrategia del modo de pánico, por lo que se ignorarán los siguientes caracteres hasta encontrar un token bien formado. En el ejemplo anterior se ignorarán los siguientes caracteres "ente\$ro" hasta terminar de formar el token "numero".

Los **errores sintácticos** son aquellos que darán como resultado de ingresar una cadena de entrada que no corresponde con la sintaxis definida para el lenguaje.

Ejemplo 2:

```
Publico numero entero = 10;
```

En la entrada del ejemplo anterior se invirtieron el orden de los tokens que se usan para formar una declaración de una variable (ver ejemplo 3). Al encontrarse con un error sintáctico la estrategia que se tomará será la de terminar el proceso de análisis y reportar el error en ese mismo instante.

Ejemplo 3:

```
Publico entero numero = 10;
```

En el ejemplo anterior se muestra el formato correcto en el que se declarará una variable.

Un **error semántico** será por ejemplo de intentar usar una variable que no ha sido declarada.

Ejemplo 4:

```
...  
...  
...
```

```
Publico entero numero1 = numero2 + 10;
```

```
...  
...  
...
```

Como se ve en ejemplo 4 la variable "numero2" no ha sido declarada por lo que se mostrará un error semántico, la forma en que se manejaran los errores semánticos consistirá en parar el análisis y mostrar un mensaje de error.

Consideraciones

- Si se existiesen varios errores léxicos se podrá continuar con el análisis de la cadena de entrada y los errores detectados se mostrarán al final del análisis.
- Si se encontrara algún error sintáctico en la cadena de entrada se deberá de reportar en ese instante y pararse el proceso de análisis.
- Si se hallará un error de tipo semántico se procederá a parar el proceso de análisis y se reportará el error de inmediato.

8.2 Manejo de Errores en la interpretación de XLS

Los errores podrán presentarse en cualquier momento de la interpretación del formulario, la aplicación deberá ser capaz de detectarlos, para poder generar un lenguaje de alto nivel sin ninguna complicación.

Errores Léxicos

Este tipo de errores en el archivo XLS puede ser los siguientes:

- Definir un carácter no valido en las celdas de la columna de identificadores, tipo, restricciones, calculo, multimedia, parámetros.

Ejemplo:

tipo	IdPregunta	etiqueta	parametro
texto	Cap\$ital	Capital de Guatemala?	cad_max = 20 cad_fila = 10
\$texto	Capital2	Capital de Guatemala?	~cad_max = 20

En la tabla anterior se introdujo arbitrariamente el componente léxico "\$" y "~", esto deberá de provocar un error léxico. La forma en que se recuperará de un error léxico es con la estrategia del modo de pánico, por lo que se ignorarán la pregunta (fila) hasta encontrar un token bien formado.

Errores Sintácticos

Este tipo de errores en el archivo XLS pueden ser los siguientes:

- Definir el valor de un parámetro y después la variable de asignación.
- Hacer la llamada a un identificador de una variable sin antes haber colocado el símbolo \$ y dentro de corchetes colocar el identificador.
- Definir una columna con un tipo inexistente.

Errores semánticos

- Definir una agrupación o ciclo y no contener ninguna pregunta dentro.
- Finalizar una agrupación o ciclo sin antes haberla iniciado.

La aplicación que analizará los archivos XLS al momento de encontrar algún error, semántico, sintáctico o léxico, la acción que realizará será: continuar con el análisis hasta terminarlo. Luego mostrará todos los errores que se encontraron. Si se ha detectado uno o más errores, la aplicación no deberá generar el código XForm correspondiente a la entrada.

9 Requisitos Mínimos

Para que la entrega y calificación del primer proyecto sea efectiva, la solución entregada deberá contener los siguientes requisitos mínimos:

- Componente de interpretación
 - Análisis del archivo XLS.
 - Interfaz gráfica.
- Estándar para elaboración de formularios
 - Archivos XLS
 - Manejo de errores
 - Tipo de dato
 - Case insensitive
 - Hoja de encuestas
 - Tipos de columnas
 - Sugerir
 - Restricción
 - Mensaje de restricción
 - Valor necesario
 - Mensaje de valor necesario
 - Condición aplicable
 - Lectura
 - Cálculo
 - Numero de repeticiones
 - Parámetros
 - Tipos de preguntas
 - Texto
 - Entero
 - Decimal
 - Condicional
 - Fecha y hora
 - Respuesta selección
 - Nota
 - Cálculo

- Agrupaciones de preguntas
 - Agrupaciones simples
 - Agrupaciones anidadas
 - Agrupación de tipo ciclo
- Operadores y funciones
- Componentes de generación de formularios
 - Generación de interfaz de los formularios
 - Interfaz gráfica.
 - Manejo de errores
 - Lenguaje de alto nivel
 - Case insensitive
 - Polimorfismo
 - Recursividad
 - Tipos de dato
 - Almacenamiento de respuestas
 - Comentarios
 - Operaciones aritméticas
 - Condiciones
 - Expresiones relacionales
 - Expresiones lógicas
 - Visibilidad
 - Importaciones
 - Clases
 - Atributos
 - Declaración de variables
 - Asignación de variables
 - Declaración e instancia de objetos
 - Métodos y funciones
 - Método principal
 - Constructor
 - Acceso a atributos y métodos de un objeto.
 - Sentencias de control
 - Sentencia Si
 - Sentencia Si sino
 - Sentencia Si Sino Si
 - Sentencia caso de
 - Bucles
 - Sentencia Romper
 - Sentencia Mientras
 - Para
 - Imprimir

- Funciones nativas
 - Funciones numéricas
 - Funciones con cadena
- Lista de opciones
- Formularios y grupos
- Preguntas
- Procedimiento respuesta

10 Entregables y Restricciones

10.1 Entregables

- Código fuente de las aplicaciones.
- Las aplicaciones deberán ser funcionales
- Ejecutables de ambas aplicaciones.
- Gramáticas de ambos lenguajes
- Manual Técnico
- Manual de Usuario

10.2 Criterios de Entrega

- La forma de entrega será virtual y en los días previos a la fecha de entrega se publicará un enlace web en el cual los estudiantes deberán subir su proyecto, el contenido a subir deberá ser todos los entregables mencionados en el enunciado, comprimidos en formato zip.
- El desarrollo del proyecto y la entrega del mismo es individual.
- El proyecto deberá ser desarrollado en los lenguajes y con las herramientas para la generación de analizadores descritos a continuación:
 - **El intérprete de XLS a lenguaje XForm deberá ser desarrollado en el lenguaje Java y con los analizadores generados con JavaCC.**
 - **El intérprete del lenguaje XForm deberá ser desarrollado en el lenguaje C# y con los analizadores generados con Irony.**
- No se recibirán proyectos después de la fecha y hora de entrega.
- Deberán entregarse todos los archivos necesarios para la ejecución de las aplicaciones, así como el código fuente de ambas aplicaciones, las gramáticas y la documentación.
- El estudiante es completa y únicamente responsable de verificar el contenido de los entregables.

10.3 Criterios de Calificación

- La calificación del proyecto se realizará presencialmente.
- La calificación del proyecto se realizará con los archivos entregados en la fecha establecida y desde los ejecutables.
- No se podrá agregar o quitar algún símbolo en los archivos de entrada. El proyecto deberá funcionar con los archivos que sean proveídos por los auxiliares para la calificación, sin modificación alguna.

- No será permitido compartir los archivos de entrada durante ni después de la calificación.
- La calificación del proyecto será personal y existirá un tiempo límite.
- Se debe tomar en cuenta que no pueden estar personas ajenas a la calificación, de lo contrario no se calificará el proyecto.
- Anomalías o copias detectadas en el proyecto tendrán de manera automática una nota de 0 puntos y los involucrados serán reportados a la Escuela de Ingeniería en Ciencias y Sistemas, para que se apliquen las sanciones correspondientes.
- Existirá horarios para la calificación de cada proyecto, por el cual el estudiante deberá de elegir el horario que mejor le convenga.
- Anomalías detectadas en los archivos entregables tendrá de manera automática una nota de 0 puntos, por ejemplo: no se envió código el código correcto, se envió parte del código y no el código completo, archivos ajenos a los entregables del proyecto, no se hizo uso de las herramientas descritas en el enunciado de cada proyecto, entre otras.
- Los archivos de entrada contendrán errores semánticos, sintácticos y léxicos para la verificación de recuperación de errores de la aplicación.

10.4 Restricciones

- **El intérprete de XLS a lenguaje XForm deberá ser desarrollado en el lenguaje Java y con los analizadores generados con JavaCC.**
- **El intérprete del lenguaje XForm deberá ser desarrollado en el lenguaje C# y con los analizadores generados con Irony.**
- Copias de proyectos tendrán automáticamente nota de 0 puntos y se reportará a los involucrados a la Escuela de Ingeniería en Ciencias y Sistemas
- No se recibirán proyectos después de la fecha y hora de entrega.
- La calificación se realizará sobre archivos ejecutables.
- La entrega deberá de cumplir con los requisitos mínimos.

Fecha de entrega: martes 18 de septiembre de 2018 hasta las 08:00 p.m.

El día de la entrega se publicarán los links donde deberán subir su proyecto.