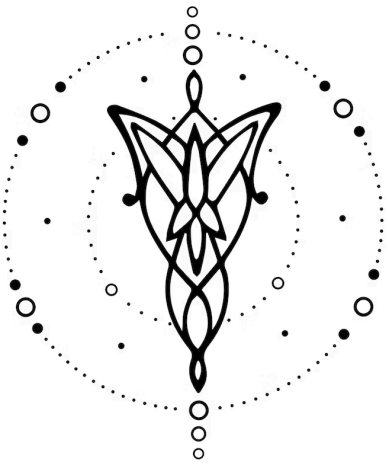

Especificación

Diseño e Implementación de un Lenguaje

V.1.0.0

1. Equipo

Nombre	Apellido	Legajo	E-Mail
Diego	Badin	63551	DIEGO BADIN
Diego	Rabinovich	63155	DIEGO JOSÉ RABINOVICH
Magdalena	Taurian	62828	MAGDALENA TAURIAN

2. Repositorio

La solución y su documentación serán versionadas en el repositorio: Repositorio

3. Dominio

Desarrollar un lenguaje de programación de dominio específico (DSL) “FlowForm” que permita diseñar formularios de propósito general o tipo encuestas con mucha capacidad de lógica condicional. El lenguaje deberá permitir que el usuario establezca preguntas, enunciados, secciones, mensajes de error, reglas de validación basadas en funciones predefinidas, así como reglas de visibilidad basadas en funciones predefinidas, y para los usuarios que busquen más libertad o complejidad: funciones customizadas.

Además se proveerá al usuario de configuraciones básicas de estilo general del formulario basado en temas que se relacionan con los álbumes de Taylor Swift. Algunas de las sintaxis y palabras clave también tendrán guiños para los fans de Taylor.

El código se construirá de manera jerarquizada, como se puede apreciar en el esquema del apéndice.

La salida dependerá de la configuración que establezca el usuario. El mismo soportará dos configuraciones de uso final: 1) Uso general. No se le asigna una función al botón de submit, y se entrega en un mismo archivo .html todo el formulario y la lógica en JavaScript. Esta configuración será ideal para quienes quieran usarlo de plantilla para embeberlo en páginas o aplicaciones web, o quieran darle un comportamiento customizado a la entrega del formulario. 2) Google Script Web App. Se le asigna al botón de submit la funcionalidad de crear entradas en una hoja de cálculo de google que se genera automáticamente. Se entregan 2 archivos, por un lado el Form.html con el formulario y la lógica, y por otro el code.gs con la lógica de la app. El usuario debería luego cargar ambos archivos en un proyecto de google scripts, deployar y en menos de 2 minutos tiene un formulario que puede compartir y recibir respuestas en una hoja de google.

4. Construcciones

El lenguaje desarrollado debería ofrecer las siguientes construcciones, prestaciones y funcionalidades:

I. Se podrán sobrescribir funcionalidades, comportamiento o estructuras anidando el símbolo @.
II. Se podrán sobrescribir textos asignando la etiqueta #.
III. Se podrán crear preguntas con el indicador “@Question”.
IV. Se podrán especificar los tipos de input que se desea obtener para cada pregunta con el indicador “#Type”. De no hacerlo el default el tipo default es texto.
V. Se podrán definir los labels de las preguntas con el indicador “@Label”

VI. Se podrá agregarle estructura al formulario con pasos, secciones, títulos y descripciones. Estos se indican con los elementos “@Step”, “@Section”, “#StepTitle”, “#StepDescription”, “#SectionTitle” y “#SectionDescription”.
VII. Se podrán crear funciones booleanas para definir condiciones de aparición de preguntas y opciones de respuesta en los casos de las preguntas desplegables, casillas y varias opciones con el indicador “@Showif”. Estas serán definidas mediante llaves (“{“, “}”) y se deberá de aclarar la sentencia o el elemento que determina el valor booleano.
VIII. Se podrán definir para cada pregunta una información de “ayuda” sobre lo que tiene que completar o su formato con la etiqueta “#Help”.
IX. Se podrán definir para cada pregunta los tipos de errores y los mensajes de error que se deberían mostrar en caso de que el usuario cometa uno. Esto se indicará mediante el elemento “@Glitch”.
X. Se podrán completar respuestas o realizar acciones en el caso de que se cumplan ciertos criterios en otras respuestas con el indicador “@Do” y “ @Task”.
XI. Se podrá indicar a qué paso se desea continuar con el indicador “@GetawayCar”.
XII. Se podrán acceder a distintos pasos del formulario con condicionales dentro del indicador “@GetawayCar”.
XIII. Se podrán crear comentarios de línea a partir de “//”
XIV. Se proveerán operadores relacionales como <, >, =, ≠, ≤ y ≥.
XV. Se proveerán operaciones aritméticas básicas como +, -, * y /.
XVI. Se proveerán operadores lógicos básicos como && (and) y || (or).
XVII. Se proveerá operadores de control condicionales básicos como IF, ELSE, ELSE IF, SWITCH.
XVIII. Se podrán indicar la cantidad de respuestas permitidas en las preguntas de casillas.
XIX. Los elementos se deben terminar con “;” o “{“ o “}” dependiendo del caso. Si se trata de un indicador que comienza con @ ó es una función custom, se debe abrir llaves y cerrar cuando termine. Si es un indicador que inicia con # o una sentencia que no inicia un indicador, se finaliza con “;”.
XX. Los nombres de los elementos se definen luego de su indicador.
XXI. Las variables dentro de las funciones se definen con “var ” seguido de un nombre, luego “ : ” valor y luego “ as ” tipo.
XXII. Se podrán agregarle un estilo al formulario seleccionando entre las opciones de temas que serán inspiradas en álbumes de Taylor Swift. Utilizando la etiqueta “#Theme”
XXIII. Se podrá definir un mensaje de salida del formulario con el indicador “#Closure”.
XXIV. El formulario se mostrará en un archivo HTML.
XXV. Se podrán almacenar las respuestas en un Google Sheets utilizando el indicador “#SafeAndSound” dentro de la configuración del formulario.

5. Casos de prueba

Se proponen los siguientes casos iniciales de prueba de aceptación:

I. Un formulario que contenga N preguntas, siendo $N > 0$.

II. Un formulario que tenga distintas secciones.
III. Un formulario donde la visualización de las entradas dependa de de una condición booleana no auto-referencial.
IV. Un formulario donde la visualización de las secciones dependa de de una condición booleana no auto-referencial.
V. Un formulario que dependiendo de la respuesta de una pregunta te envíe a otro paso del formulario.
VI. Un formulario con diferentes tipos de entradas (checkboxes, radio buttons, dropdowns, etc).
VII. Un elemento que sea seteado en base a otro valor de manera automática, a través de los indicadores @Do y @Task de otro elemento.
VIII. Un formulario que tiene un mensaje de salida personalizado.
IX. Un formulario que tiene un diseño inspirado en un álbum de Taylor Swift.
X. Un formulario que muestre un mensaje de error en caso de encontrarse con uno.
XI. Un formulario que almacena sus respuestas en un Sheets de Google.

Además, los siguientes casos de prueba de rechazo:

I. Una entrada con condiciones de visualización no booleanas.
II. Un elemento con condiciones de visualización auto-referenciadas en funciones predefinidas. Es decir, que dependa de su propio input si es visualizado o no.
III. Un formulario con mala terminación de sentencias.
IV. Un formulario con indicadores obligatorios no definidos.
V. Un formulario que no respete la estructura jerárquica.
VI. Un formulario con indicadores faltantes y que sean obligatorios por la aparición de su padre.
VII. Un formulario con más de una configuración definida.
VIII. Una entrada que intente generar un input con un tipo no soportado.
IX. Un elemento que intenta usar funciones de un tipo incorrecto. i.e. un input Text al que se le intenta agregar @Options.
X. Un formulario con referencias a entradas o campos inexistentes.

6. Ejemplos

a. Crear un formulario básico para uso general con 2 preguntas, en la cual una depende de la otra.

```
#FormName "Basic Form";
@Question q1 {
   #Label "Username";
   #Type Text;
   #PlaceHolder "Enter your username";
}
@Question q2 {
   #Label "Question 2";
   #Type Text;
   @ShowIf {
      q1.value.equals(“Diego”) || q1.value.equals(”Maggie”);
   }
}
```

b. Crear un formulario para uso de encuesta con google scripts, donde las respuestas se guardan en una sheet que se va a llamar “basic-form-results”. Se pide que una de las preguntas aparezca solo si la otra tiene un valor mayor a 10, y cambia de opciones según si el valor es menor o mayor a 50. Además se chequea que una de las preguntas recibe un valor positivo.

```
#FormName "Basic Form";
@FormConfig {
    #SafeAndSound "basic-form-results";
}

@Question q1 {
    #Label "Question 1";
    #Type Numeric;
    @Glitch {
        @Error {
            when value <= 0;
            #Message "q1 must be higher than 0";
        }
    }
}
@Question q2 {
    #Label "Question 2";
    #Type DropDown;
    #PlaceHolder "Select an Option";
    var isLower : q1.value < 50 as boolean;
```

```
    @Options {
        "opt1" showIf(isLower),
        "opt2" showIf(isLower),
        "opt3",
        "opt4" showIf(!isLower);
    }

    @ShowIf {
        q1.value > 10;
    }
}
```

c. Un formulario de tres preguntas, una en cada paso, donde la segunda se saltea si la entrada de la primera contiene la palabra “hi”.

```
#FormName "3rd Form";

@Step s1 {
    #StepTitle "Step 1";
    @Question q1 {
        #Label "q1";
        #Type Text;
    }
    @GetawayCar {
        if (q1.value.containsWord("hi")){
            s3;
        }
    }
}

@Step s2 {
    #StepTitle "Step 2";
    @Question q2 {
        #Label "q2";
        #Type Text;
    }

}
```

```
@Step s3 {
    #StepTitle "Step 3";
    @Question q3 {
        #Label "q3";
        #Type Text;
    }
}
```

7. Apéndice

El lenguaje tendrá una jerarquía claramente definida como lo muestra el siguiente diagrama: