Buzzter Company

Fase II

Nombre Código: Buzzter

2013

Equipo de Desarrollo

- Jorge Alexander Quijivix
- Estuardo José Díaz
- Gerson Alfredo Gómez
- Karen Pahola Mayorga

La información es el secreto para liberarnos de la esclavitud

Universidad Rafael Landívar
Campus Quetzaltenango
Curso: Ingeniería de Software I
Ing: Ivan de León

"FASE II - Buzzter.co"

Equipo de desarrollo

Jorge Alexander Quijivix	**Carné #: 15 551 09 – Director de proyecto**
Karen Pahola Mayorga	**Carné #: 15 118 10 – Desarrolladora**
Estuardo José Antonio Díaz	**Carné #: 15 470 10 – Desarrollador**
Gerson Alfredo Gómez	**Carné #: 15 186 09 – Desarrollador**

Ingeniería en Informática y Sistemas
Quetzaltenango 26 septiembre de 2013

SUMARIO

Una vez terminada la fase anterior, y habiendo detectado las necesidades más importantes del proyecto, se procedió a la realización de los diagramas UML necesarios que nos ayudarán a la codificación que es el siguiente paso.

Una vez definido los diagramas UML fue tiempo de la creación de la base de datos sobre la cual se establecerá del proyecto, además de ello, se procedió a la realización de propuestas de calidad junto a ello los estándares de desarrollo que nos guiarán a lo largo de la codificación.

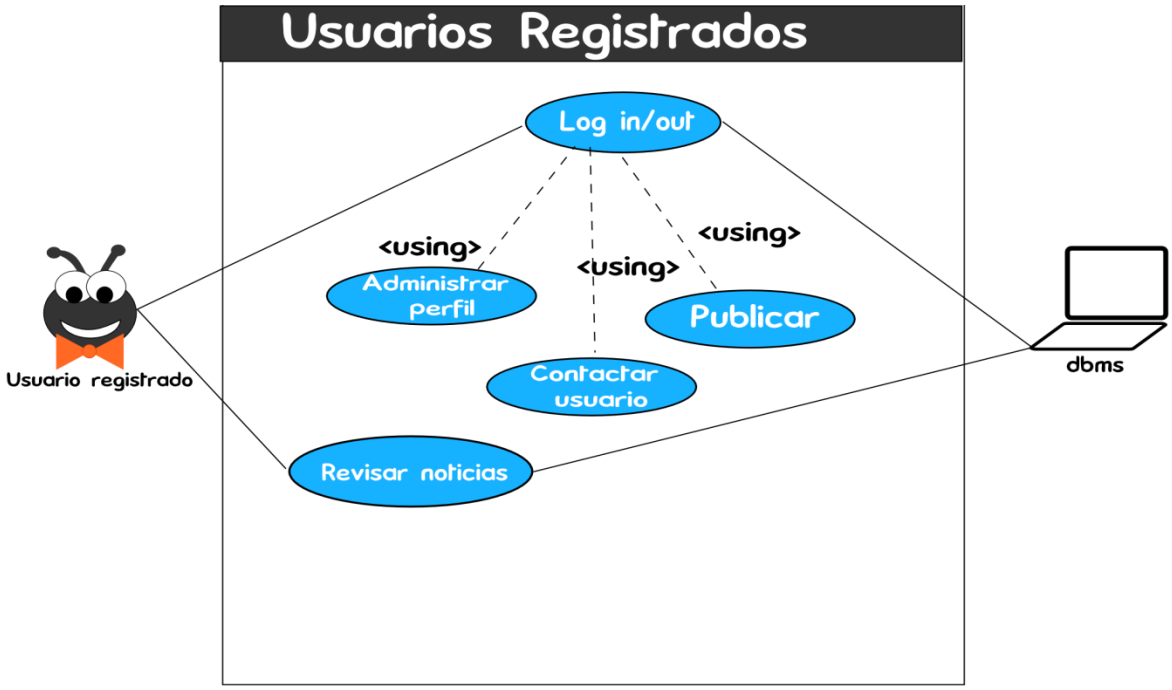
Por lo anterior mencionado, a continuación se presenta a detalle cada punto.

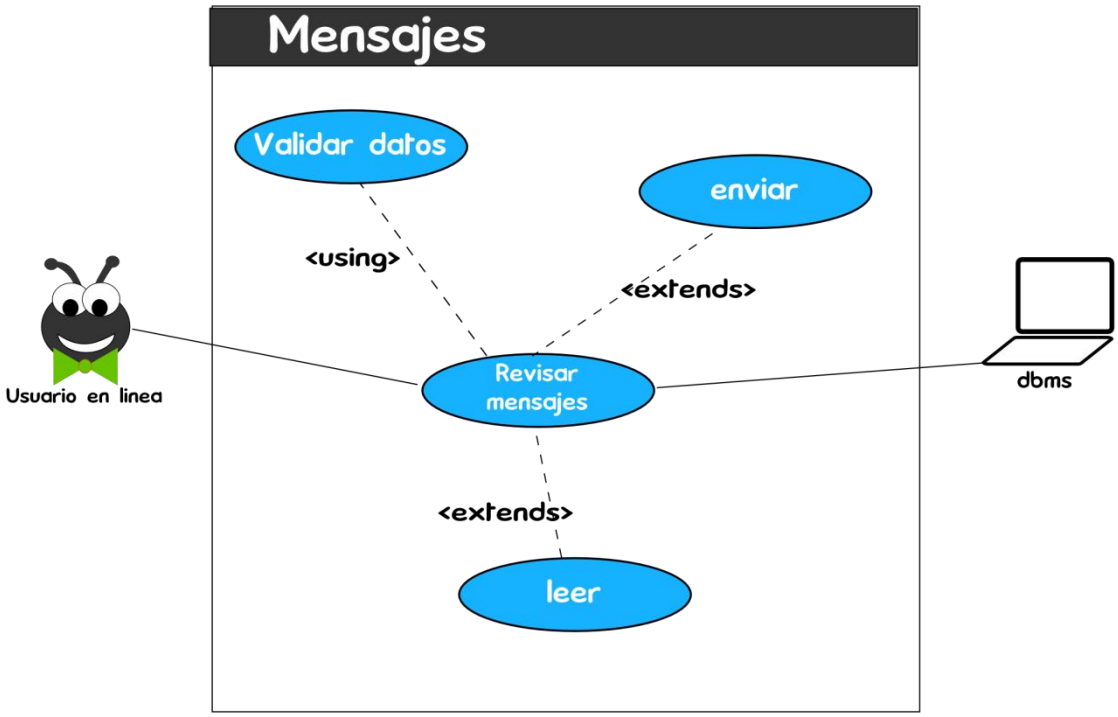
NOTA: Los diagramas aquí presentados y la fase 1, propuesta y planificación, se encuentran desde ya en el repositorio de GitHub.

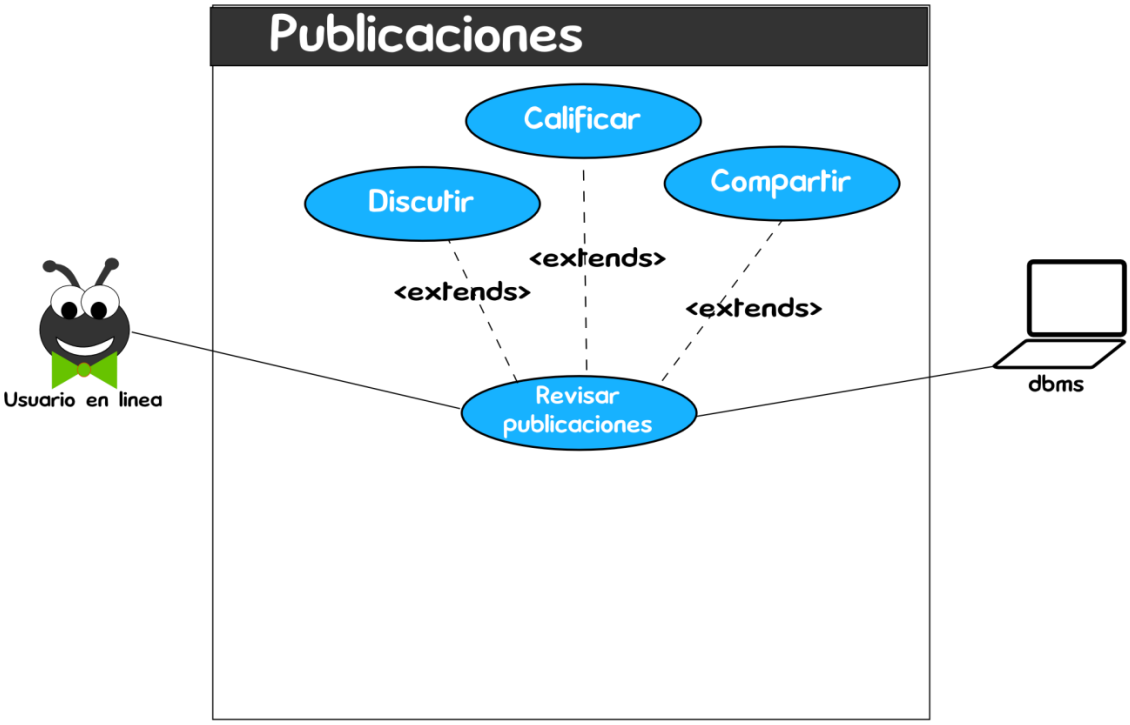
Link: https://github.com/JGAlex/BuzzterCo.git

DIAGRAMAS UML

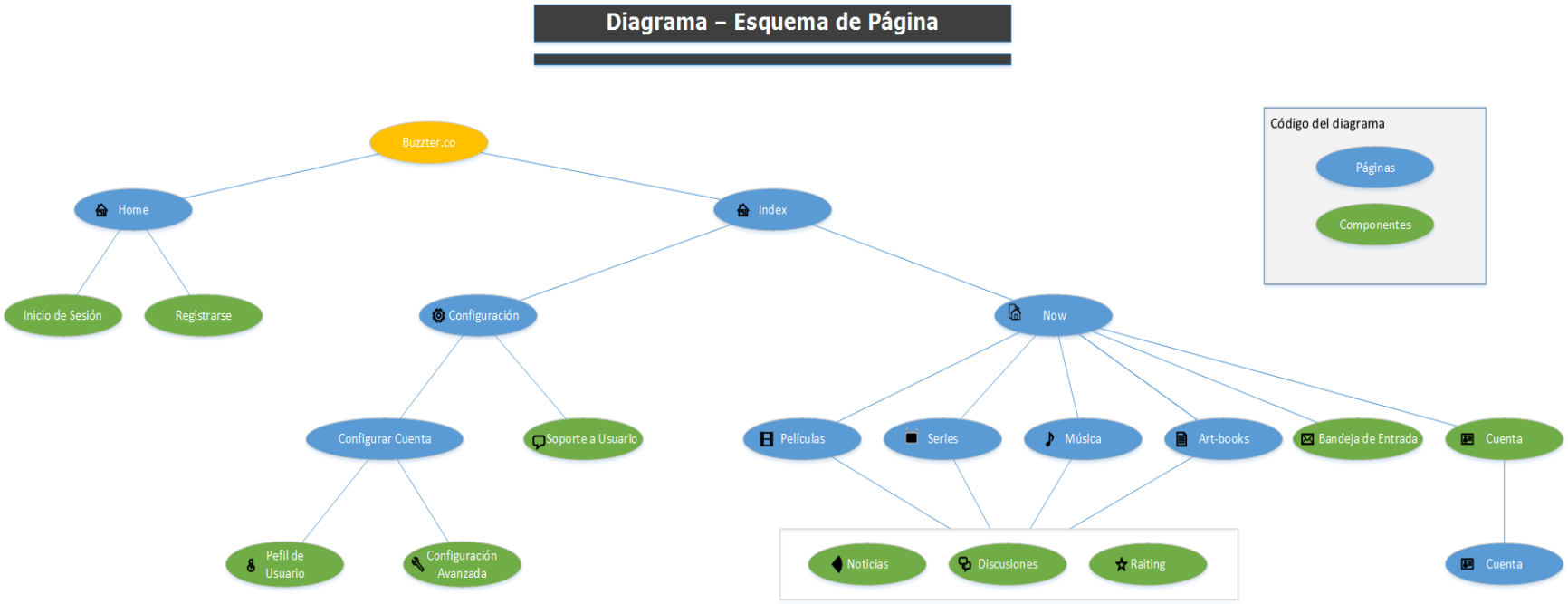
1. CASOS DE USO



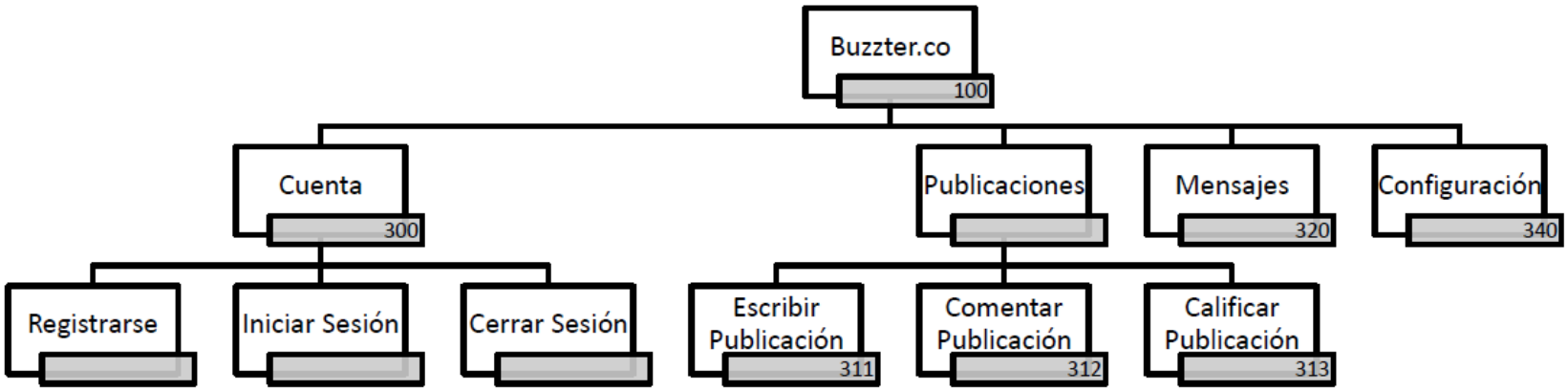

Mensajes
Validar datos
enviar
<using>
<extends>
Usuario en linea
Revisar mensajes
dbms
<extends>
leer

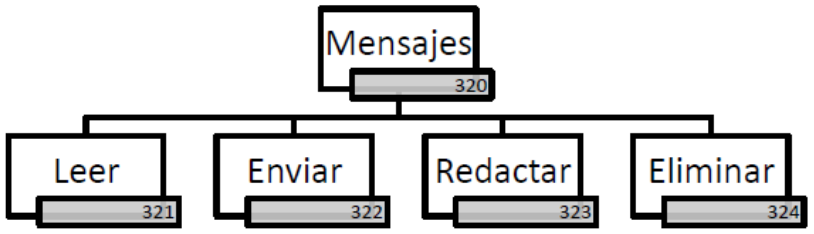

Publicaciones
Calificar
Discutir
Compartir
<extends>
<extends>
<extends>
Usuario en linea
Revisar publicaciones
dbms

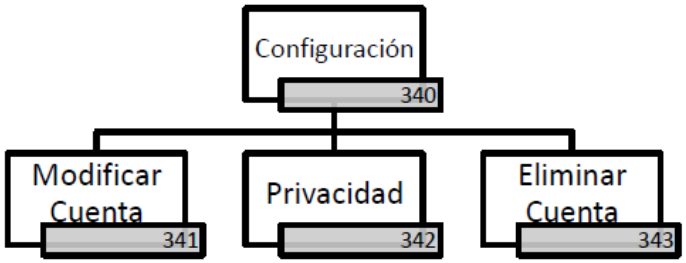
2. Esquema de página



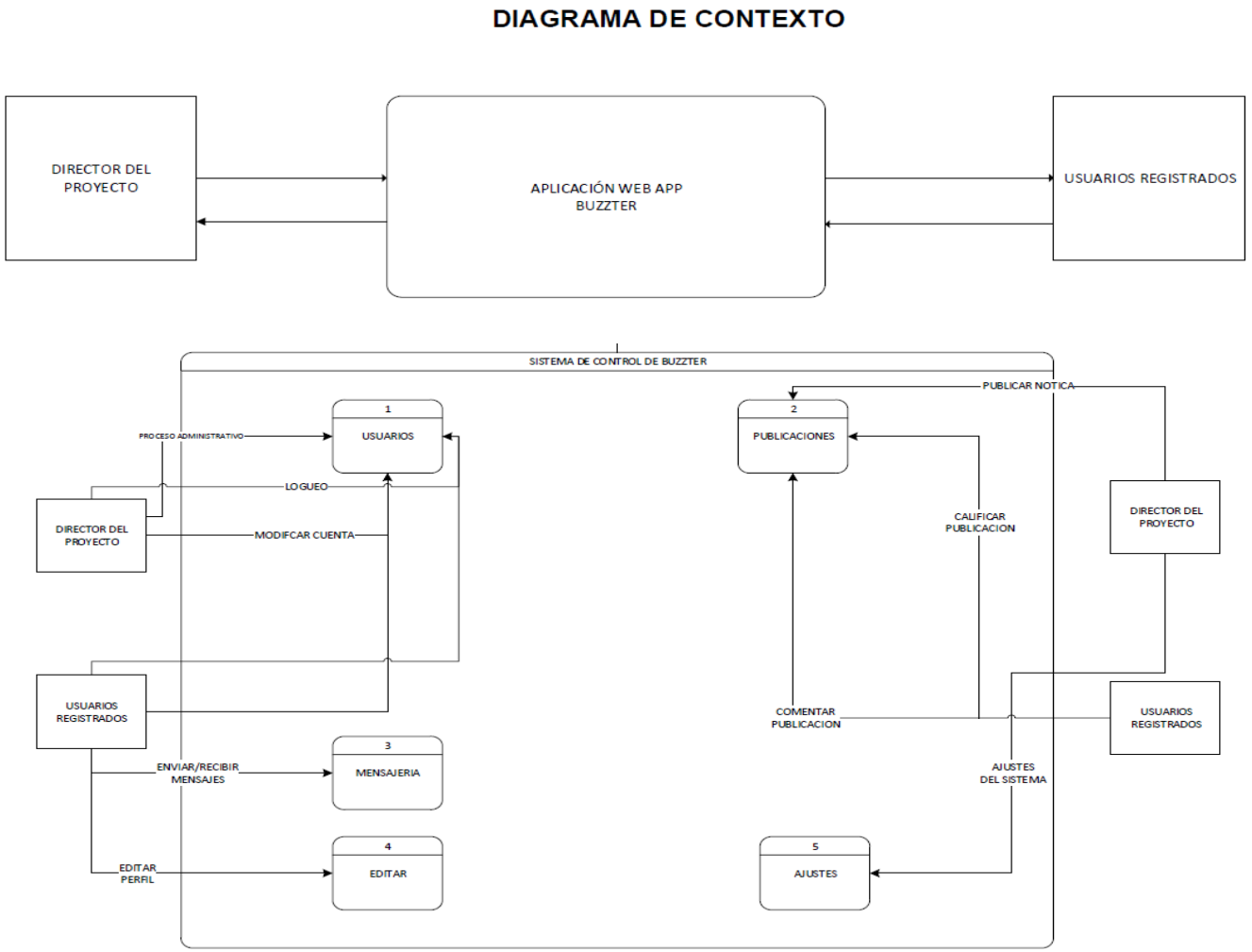
3. Diagrama de Estructura

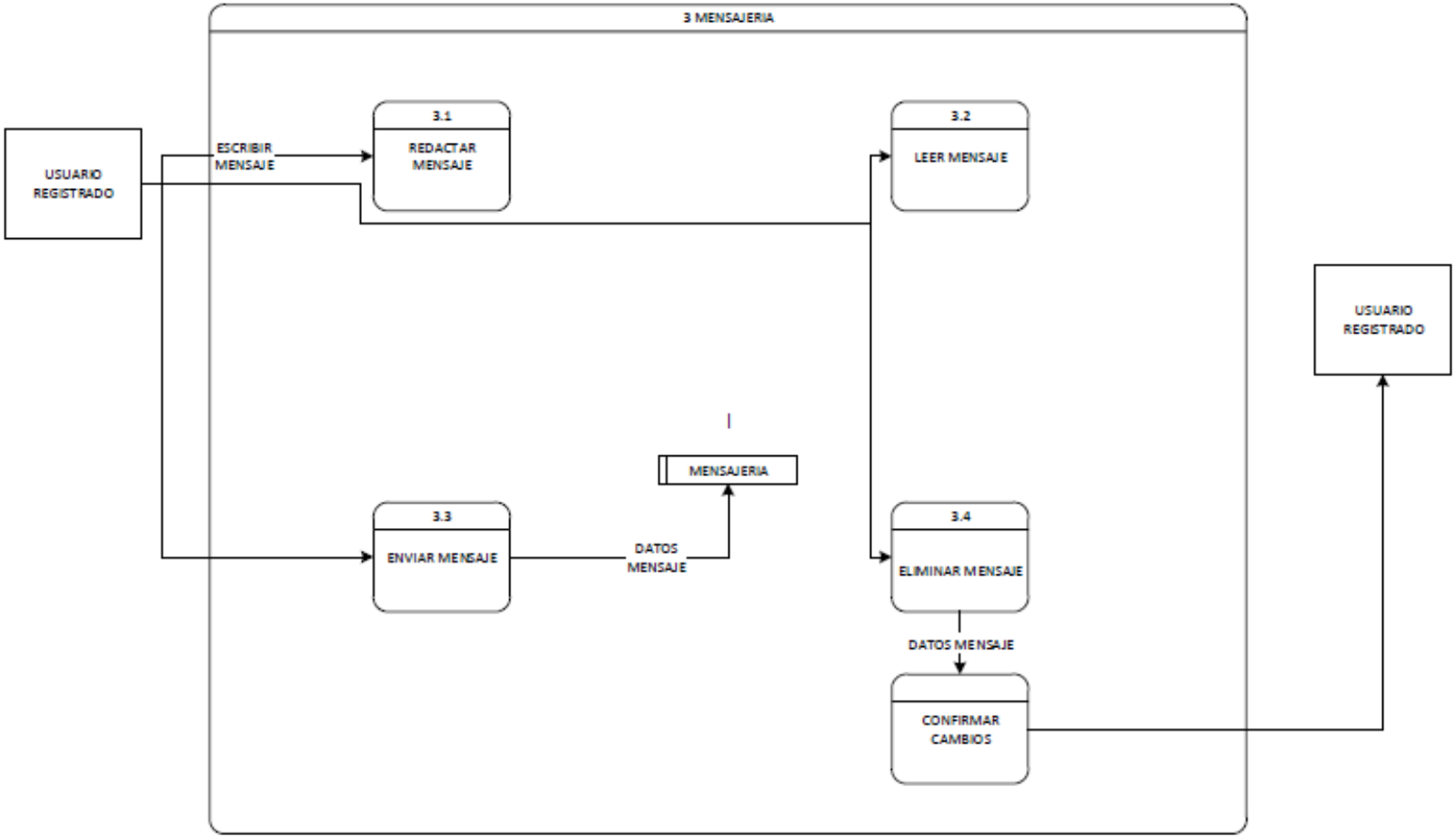


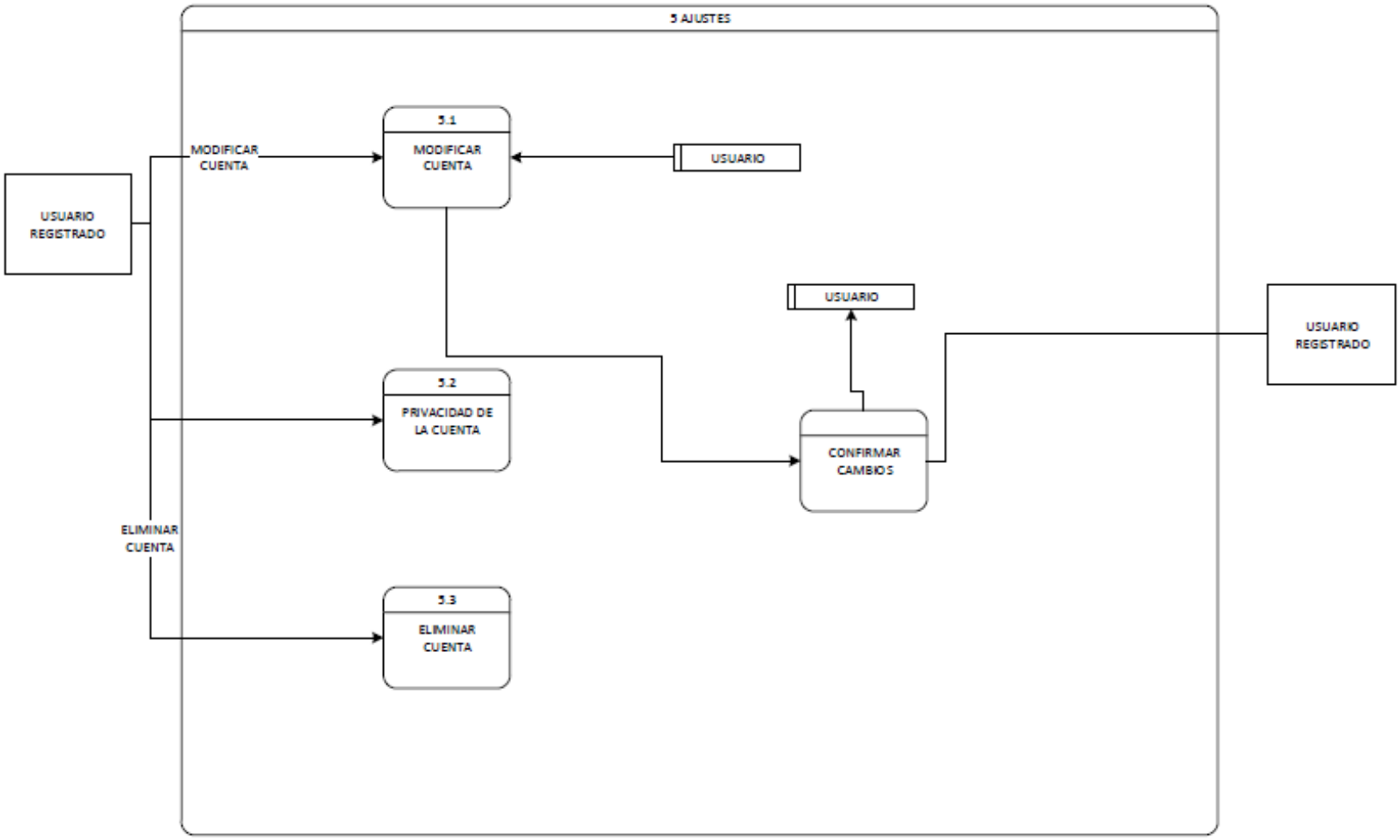


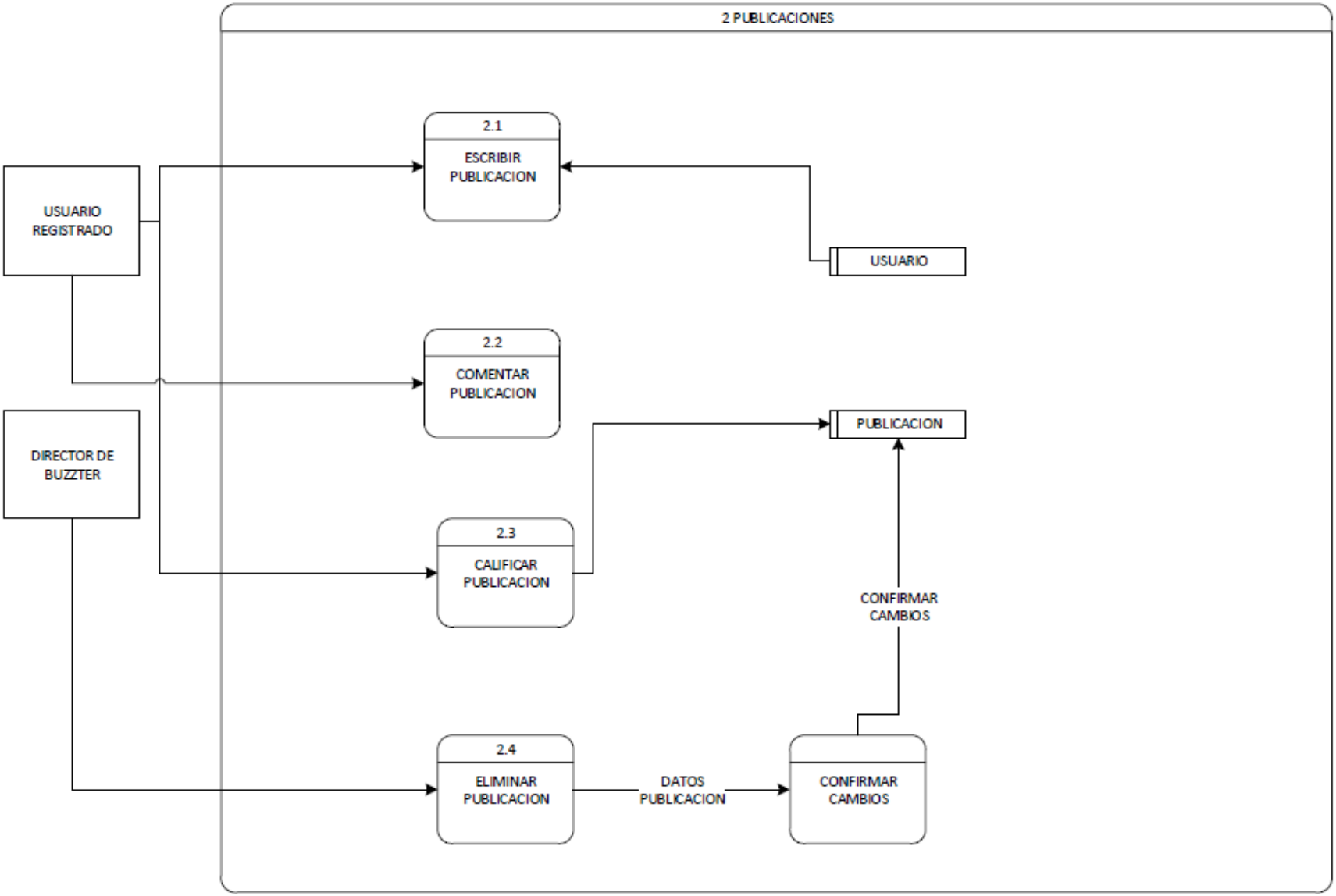


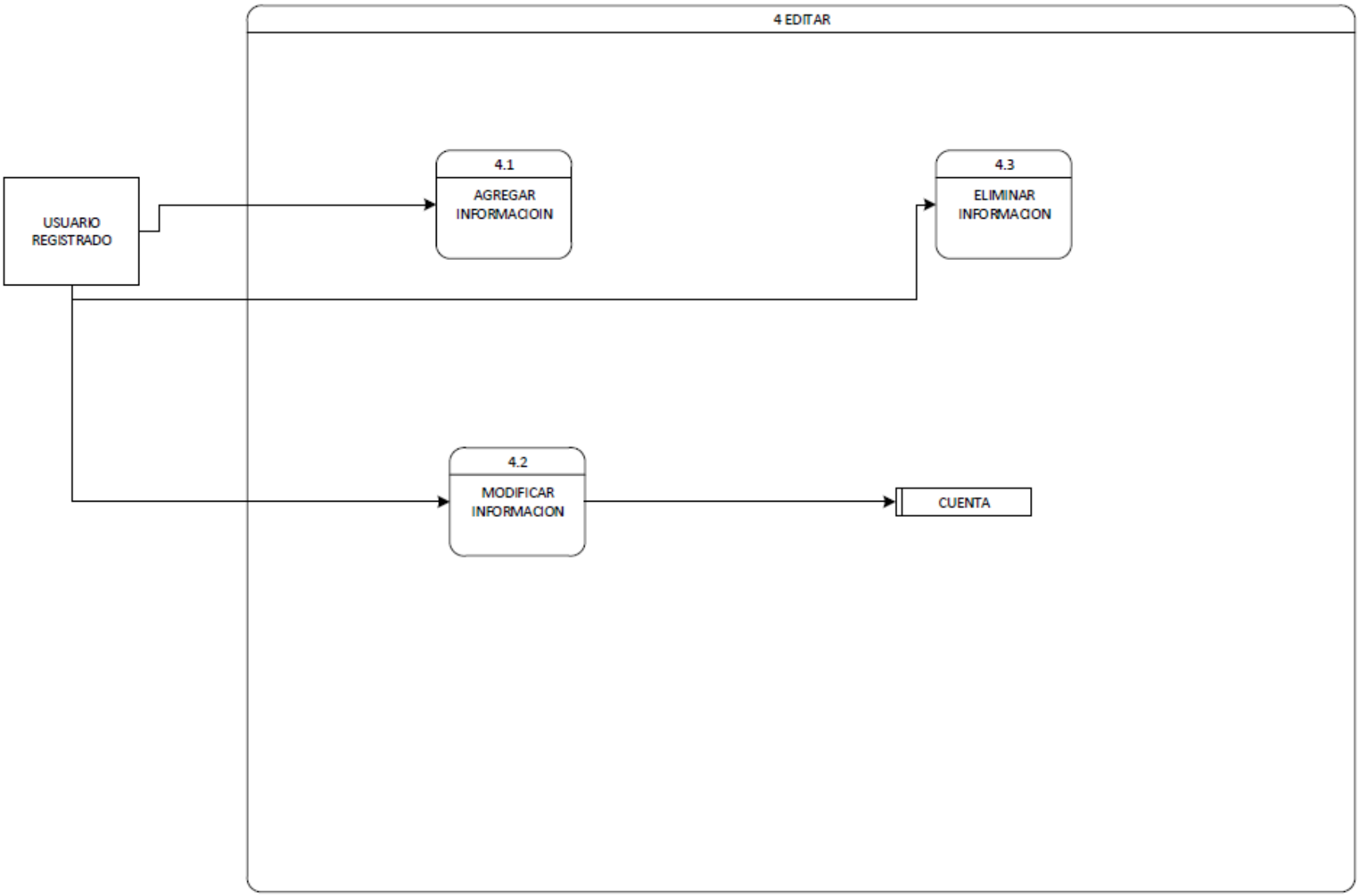
4. Diagramas de Flujo



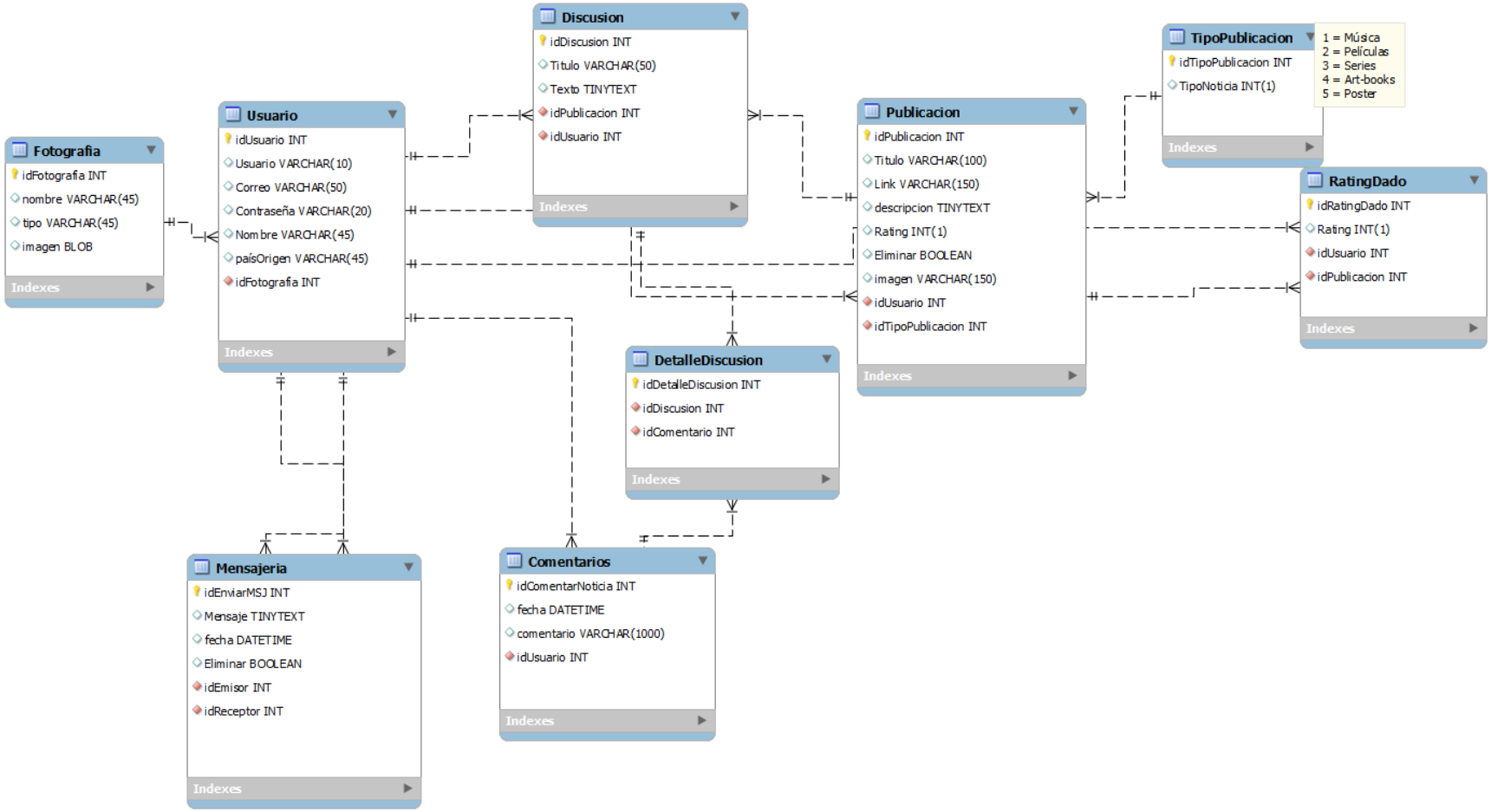

3 MENSAJERIA
USUARIO REGISTRADO
ESCRIBIR MENSAJE
3.1
REDACTAR MENSAJE
3.2
LEER MENSAJE
MENSAJERIA
3.3
ENVIAR MENSAJE
DATOS MENSAJE
3.4
ELIMINAR MENSAJE
DATOS MENSAJE
CONFIRMAR CAMBIOS
USUARIO REGISTRADO


5 AJUSTES
5.1
MODIFICAR CUENTA
USUARIO
MODIFICAR CUENTA
USUARIO REGISTRADO
USUARIO
USUARIO REGISTRADO
5.2
PRIVACIDAD DE LA CUENTA
CONFIRMAR CAMBIOS
ELIMINAR CUENTA
5.3
ELIMINAR CUENTA


2 PUBLICACIONES
2.1
ESCRIBIR PUBLICACION
USUARIO REGISTRADO
USUARIO
2.2
COMENTAR PUBLICACION
DIRECTOR DE BUZZTER
PUBLICACION
2.3
CALIFICAR PUBLICACION
CONFIRMAR CAMBIOS
2.4
ELIMINAR PUBLICACION
DATOS PUBLICACION
CONFIRMAR CAMBIOS


4 EDITAR
USUARIO REGISTRADO
4.1
AGREGAR INFORMACIOIN
4.3
ELIMINAR INFORMACION
4.2
MODIFICAR INFORMACION
CUENTA

Diseño Base de Datos



Pruebas de Calidad

Todas las empresas y usuarios saben de la importancia de la calidad en un sitio web, la imagen de quien está detrás de un sitio web está directamente relacionada con la calidad de su web. Uno de los factores para determinar dicha calidad es la usabilidad de ésta. Existe también un factor de estudio que son las pruebas heurísticas. Una de las ventajas de las pruebas heurísticas es su bajo costo, en comparación con otros métodos de control de calidad para sitios web.

Este costo estará determinado principalmente por el número de evaluadores y la cantidad de principios a evaluar.

Visibilidad del estado del sistema:

La prueba mide si el usuario siempre sabe qué está haciendo el sistema. Se revisa si existen los diferentes elementos que ayudan a esto:

- Indicación gráfica de donde se encuentra (rutas de accesos desde página principal)
- Indicación de que ha visto (marcar los enlaces visitados, recientemente y con frecuencia)
- Indicación de que hay un proceso en marcha
- Indicación de cuántos pasos faltan para terminar (como en el caso de que ya a un proceso de registro en el Sitio Web)

Similitud entre el sistema y el mundo real:

La prueba mide si el sitio se expresa de una manera comprensible para el usuario. Para ello se revisa si se emplean las convenciones habituales y que le permiten operar en el Sitio Web.

Control y libertad del usuario:

La prueba mide si los usuarios que se equivocan al hacer algo tienen forma de recuperarse de esos errores. Se revisa si existen formas de hacerlo. Por ejemplo: ¿Se puede deshacer una operación? ¿Se puede rehacer una operación?

Consistencia y cumplimiento de estándares:

La prueba mide si se cumplen los estándares que se usan en la Internet en el Sitio Web. Para ello se debe validar y revisar el sitio con las herramientas ya existentes que ofrecen distintos sitios web, como lo es www.w3.org

Prevención de errores:

La prueba permite validar si se cuenta con mecanismos que aseguren que el ingreso de cualquier información, por parte del usuario, permite evitarle errores. Para ello, se verifica si en las áreas en que los usuarios deben interactuar con el sistema, se les explica claramente lo que se espera de ellos. Uso de elementos destacados en los formularios: indicar los campos obligatorios con asteriscos (*) o campos obligatorios marcados con color.

Preferencia al reconocimiento que a la memorización:

La prueba permite revisar si el Sitio Web ayuda al usuario a recordar cómo se hacía una operación, o bien le obliga a aprenderse los pasos cada vez que ingresa. Para conseguir este objetivo se verifica la existencia de una línea gráfica uniforme en todo el Sitio Web y si se cuenta con un sistema de navegación coherente.

Flexibilidad y eficiencia de uso:

La prueba permite revisar si se ofrecen soluciones diferentes de acceso a los contenidos, a los usuarios novatos respecto de los expertos. Es decir, se puede contar con botones para los primeros y atajos de teclado para el experto. También es importante medir en esta prueba la carga rápida de los sitios mediante una buena construcción del código.

Estética y diseño minimalista:

La prueba pide que los elementos que se ofrezcan en la pantalla tengan una buena razón para estar presentes. Se verifica la existencia de elementos irrelevantes que no aportan ni ayudan a que el usuario distinga lo importante de lo superfluo. Para ello se verifica la existencia de:

- Jerarquías visuales: que permiten determinar lo importante con una sola mirada.
- Tamaño de imágenes: que no afectan la visión general de la información del Sitio Web; se verifica tanto tamaño como peso.

Ayuda ante errores:

Se verifica que el usuario sepa cómo enfrentar problemas en una página tanto online como offline; entre los elementos que se miden se cuentan:

- Mensaje 404 personalizado, con el fin de ofrecer una información y navegación alternativa cuando una página no es encontrada.

- Mensaje de falla ofrece una alternativa offline que permite que el usuario mantenga su confianza.

Ayuda y documentación:

Se revisa que el Sitio Web ofrezca ayuda relevante de acuerdo al lugar en que el usuario esté visitando; también se revisa la existencia de sistemas de búsqueda que permiten al usuario encontrar los elementos de ayuda que sean relevantes.

Estándares de Desarrollo

En buzzter estamos comprometidos con entregar un servicio de calidad a nuestros clientes, previo a iniciar la codificación de buzzter hemos puesto algunas pequeñas reglas que el equipo de desarrollo debe seguir al codificar.

A continuación se presentan las reglas que el equipo de buzzter debe seguir:

1. Utilizar minúsculas para nombrar las variables, y si estas tienen más de dos palabras, cada nueva palabra deberá ser iniciada con una letra mayúscula. Ej.:

```
#### Declarando una variable

text = "hola"

#### Declarando una variable con varias palabras

unTextoLargo = "Hola mundo"
```

2. Utilizar verbos para definir métodos, o al menos la primer palabra que forma parte del nombre del método, buzzter, se deben seguir las mismas reglas que en la declaración de variables. Ej.:

```
#### Definiendo el nombre de un método

def cargarDatos (self):
```

3. No se permite el uso de guiones bajos para separar palabras de métodos, excepto por lo definido bajo la sintaxis de python y el framework mismo.

4. Los nombres de las clases deben iniciar con mayúscula, se permite la definición de múltiples clases en un mismo archivo. Ej.:

```
class Persona:

    def cargarDatos:

        return
```

5. Se prohíbe las declaraciones simultáneas de variables, es decir no se pueden inicializar variables en una misma línea. Ej.:

```
#### No se permite

    Persona persona1, persona 2

#### Declaración permitida

    Persona persona1

    mensaje = "hola"
```

6. Puesto que Buzzter se basa en Django, hacer uso adecuado del Framework en cuestión

7. Se debe comentar respectivamente cada método y clase creado, los datos a incluir dentro del comentario al ser creado son:

    - Fecha y hora de creación de la clase o método
    - Branch en donde fue incluida la función por primera vez.
    - Autor del método o clase
    - Parámetros de la clase
    - valor que devuelve
    - Excepciones que deben ser capturadas
    - Última modificación

8. El sistema de archivos que se maneje debe cumplir con nuestros estándares también, al incluir URLs o URIs dentro del código deberá de utilizarse una dirección relativa, por lo cual los archivos en buzzter deben guardarse de la siguiente forma:

    - De incluirse imágenes, debe crearse una subcarpeta exclusiva para imágenes dentro de la carpeta original.
    - No se permitirá combinar imágenes y archivos de script o templates dentro de la misma carpeta.
    - Debe incluirse una carpeta para templates si van a utilizarse
    - Debe incluirse una carpeta de estilos si van a utilizarse
    - Las carpetas que contengan archivos de script no podrán contener archivos de otro tipo, sin embargo podrán incluir sub-carpetas
    - Se debe incluir un archivo readme dentro de cada subcarpeta, donde se describirá el fin que tiene el contenido del mismo

~~Editando HTML~~

Para diseñar buzzter, se debe hacer uso del estándar definido por HTML5, no se incluirán archivos que no estén definidos bajo la especificación HTML

IMPORTANTE: Todos los archivos, deben tener incluido, el autor de cada uno de ellos, y la fecha en que fue creado.

¿Cuál es el siguiente paso?

Fase III

Una vez terminada la fase anterior, procederemos a iniciar con la codificación de Buzzter. Las herramientas a utilizar para dicha codificación serán:

- Phyton
- Django
- HTML 5
- CSS
- JavaScript

Los detalles serán descritos en el siguiente informe.