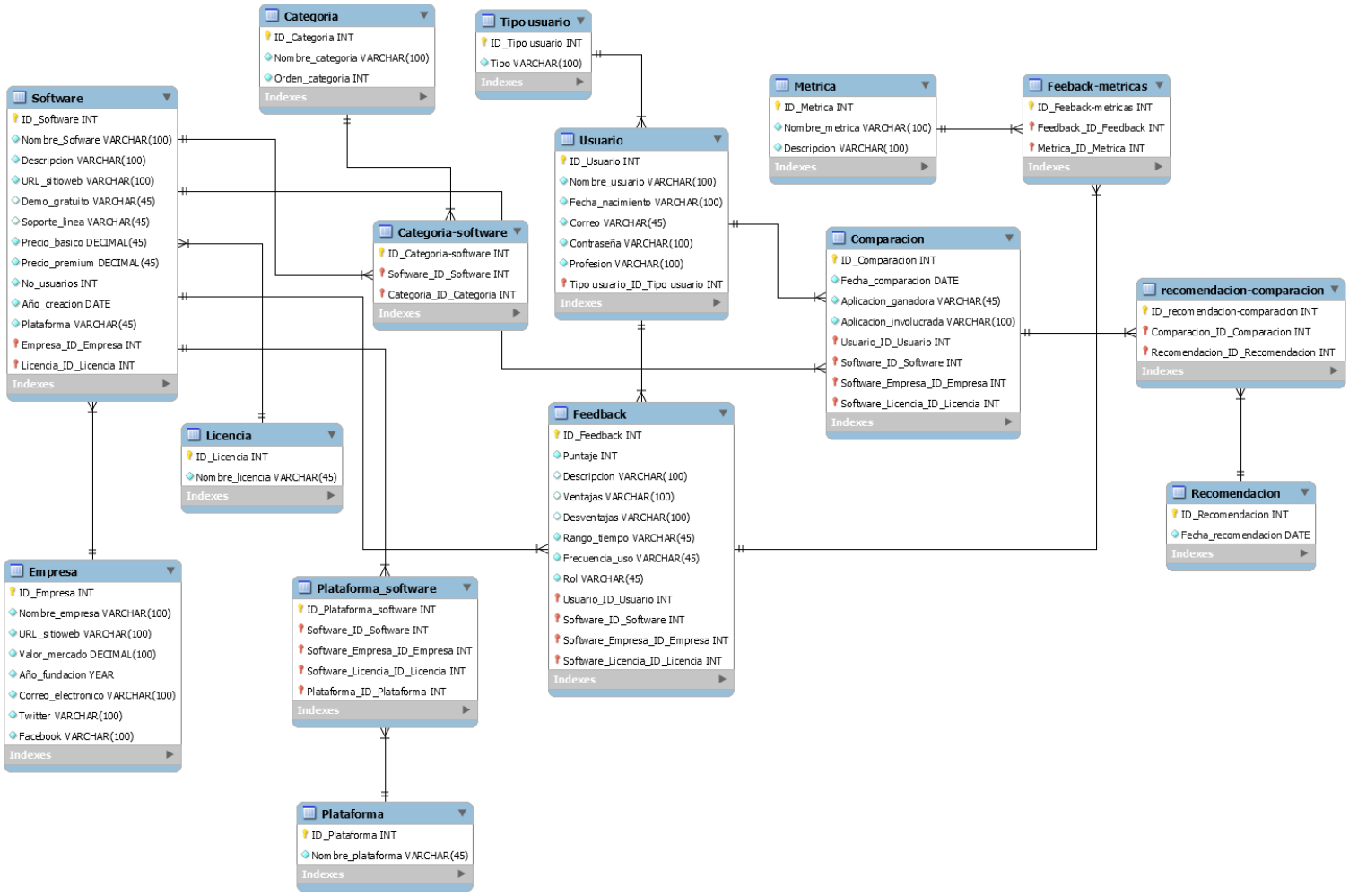
Diagrama entidad relación



Esquema base de datos

```
CREATE TABLE Metrica(
ID_Metrica numeric(10) not null,
ID_Nombre_Metrica varchar(100) not null,
Descripcion_Metrica varchar(100) not null,
Primary Key(ID_Metrica),
Unique(ID_Nombre_Metrica)
);

CREATE TABLE Empresa(
ID_Empresa numeric(10) not null,
ID_Nombre_Empresa varchar(100) not null,
URL_StoreWeb varchar(100) not null,
Valor_Mercado decimal(38) not null,
Año_Fundacion int not null,
Correo_Empresa varchar(100) not null,
Twitter varchar(100) not null,
Facebook varchar(100) not null,
Primary Key(ID_Empresa),
Unique(ID_Nombre_Empresa)
);

CREATE TABLE Plataforma(
ID_Plataforma numeric(10) not null,
ID_Nombre_Plataforma varchar(100) not null,
Primary Key(ID_Plataforma),
Unique(ID_Nombre_Plataforma)
);

CREATE TABLE Categoria(
ID_Categoria numeric(10) not null,
ID_Nombre_Categoria varchar(100) not null,
Primary Key(ID_Categoria),
Unique(ID_Nombre_Categoria)
);
```

```
CREATE TABLE Tipo_usuario(
ID_Tipo_Usuario numeric(10) not null,
Tipo_Usuario varchar(100) not null,
Primary Key(ID_Tipo_Usuario)
);

CREATE TABLE Licencia(
ID_Licencia numeric(10) not null,
ID_Nombre_Licencia varchar(100) not null,
Primary Key(ID_Licencia),
Unique(ID_Nombre_Licencia)
);

CREATE TABLE Profesion(
ID_Profesion numeric(10) not null,
Nombre_Profesion varchar(100) not null,
Primary Key(ID_Profesion)
);

CREATE TABLE Metrica(
ID_Metrica numeric(10) not null,
ID_Nombre_Metrica varchar(100) not null,
Descripcion_Metrica varchar(100) not null,
Primary Key(ID_Metrica),
Unique(ID_Nombre_Metrica)
);
```

```
----------Foraneas----------

CREATE TABLE Software(
ID_Software numeric(10) not null,
ID_Nombre_Software varchar(100) not null,
Descripcion_Software varchar(100) not null,
URL_SitoWeb_Software varchar(100) not null,
Demo_gratuito varchar(100) not null,
Soporte_linea varchar(100) not null,
Precio_basico_Software decimal(38) not null,
Precio_premium_Software decimal(38) not null,
No_usuarios int not null,
Año_creacion_Software int not null,
ID_Licencia numeric(10) not null,
ID_Empresa numeric(10) not null,
Primary Key(ID_Software),
Unique(ID_Nombre_Software),
Foreign Key(ID_Licencia) references Licencia(ID_Licencia),
Foreign Key(ID_Empresa) references Empresa(ID_Empresa)
);

CREATE TABLE Usuario(
ID_Nombre_Usuario varchar(100) not null,
Fecha_Nacimiento varchar(100) not null,
ID_Correo varchar(100) not null,
Contraseña varchar(100) not null,
ID_Profesion numeric(10) not null,
ID_Tipo_Usuario numeric(10) not null,
Primary Key (ID_Correo),
Unique(ID_Nombre_Usuario),
Foreign Key(ID_Tipo_Usuario) references Tipo_usuario(ID_Tipo_Usuario),
Foreign Key(ID_Profesion) references Profesion(ID_Profesion)
);
```

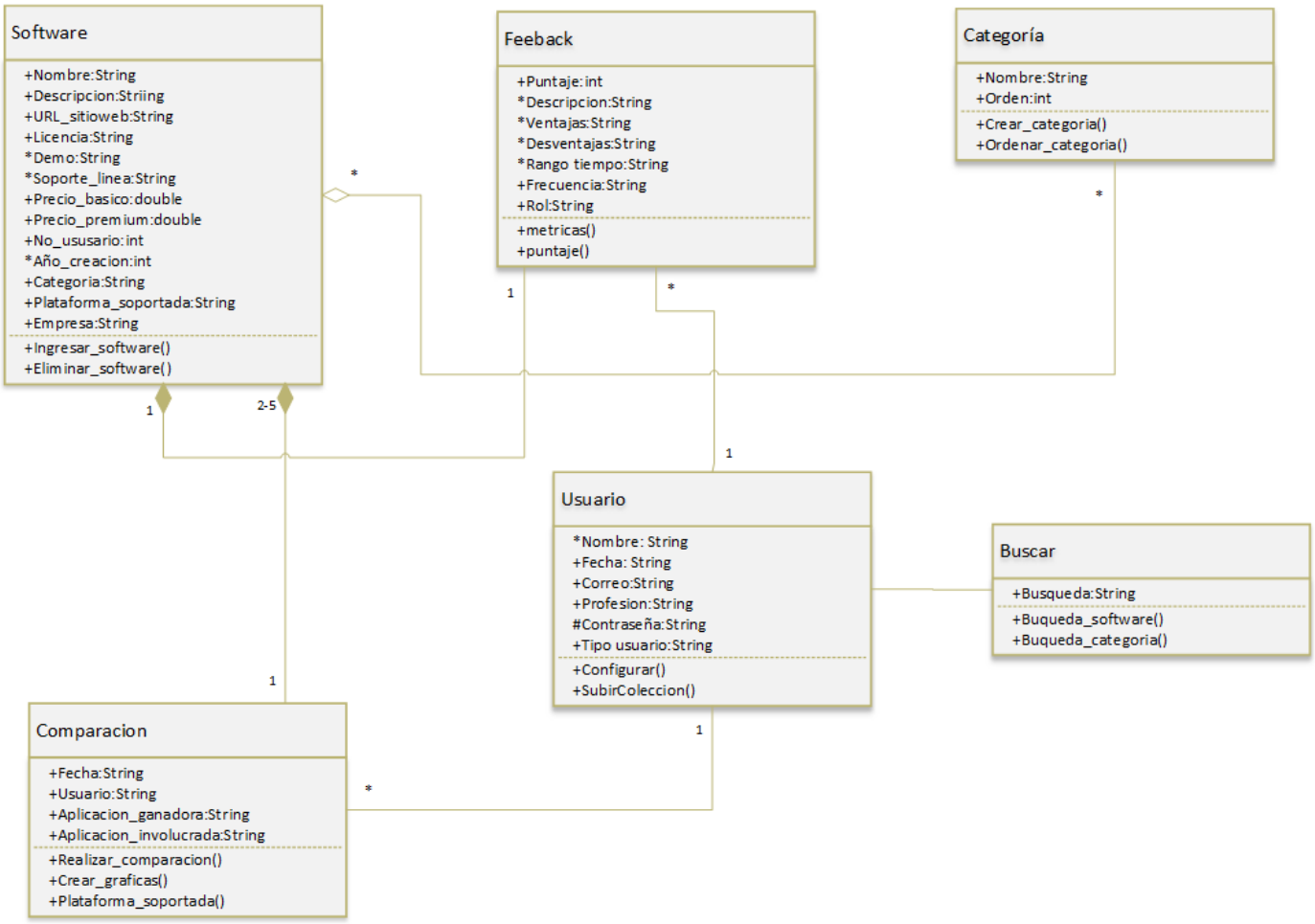

```
CREATE TABLE Usuario(
ID_Nombre_Usuario varchar(100) not null,
Fecha_Nacimiento varchar(100) not null,
ID_Correo varchar(100) not null,
Contraseña varchar(100) not null,
ID_Profesion numeric(10) not null,
ID_Tipo_Usuario numeric(10) not null,
Primary Key (ID_Correo),
Unique(ID_Nombre_Usuario),
Foreign Key(ID_Tipo_Usuario) references Tipo_usuario(ID_Tipo_Usuario),
Foreign Key(ID_Profesion) references Profesion(ID_Profesion)
);
```


```
----------Compartidas----------

CREATE TABLE Categoria_Software(
ID_CS numeric(10) not null,
ID_Categoria numeric(10) not null,
ID_Software numeric(10) not null,
Primary Key(ID_CS),
Foreign Key(ID_Categoria) references Categoria(ID_Categoria),
Foreign Key(ID_Software) references Software(ID_Software)
);

CREATE TABLE Plataforma_Software(
ID_PS numeric(10) not null,
ID_Plataforma numeric(10) not null,
ID_Software numeric(10) not null,
Primary Key(ID_PS),
Foreign Key(ID_Plataforma) references Plataforma(ID_Plataforma),
Foreign Key(ID_Software) references Software(ID_Software)
);
```

Modelo conceptual



Glosario

Términos:
Algoritmo (algorithm)
Método que describe cómo se resuelve un problema en término de las acciones que se ejecutan y especifica el orden en que se ejecutan estas acciones. Los algoritmos ayudan al programador a planificar un programa antes de su escritura en un lenguaje de programación.
Ámbito de clase (scope class)
Las variables privadas definidas fuera de los métodos internos a la clase tienen ámbito de clase. Son accesibles desde todos los métodos del interior de la clase, con independencia del orden en que están definidas. El método privado también tiene ámbito de clase.
Análisis (análysis)
Proceso de identificación, modelado y descripción de lo que hace un sistema y de cómo trabaja.
Aplicación (application)
Programa autónomo Java tal como cualquier programa escrito utilizando un lenguaje de alto nivel. Las aplicaciones se pueden ejecutar desde cualquier computadora con un intérprete Java. Las aplicaciones no están sometidas a las restricciones impuestas los applets de Libro Java 2 Java. Una clase aplicación debe contener un método main. Se utiliza como sinónimo de programa.
Array (array, vector, lista)
Objeto contenedor que almacena una secuencia indexada de los mismos tipos de datos. Normalmente los elementos individuales se referencian por el valor de un índice. El índice es un valor entero que, suele comenzar, en 0 para los primeros elementos, 1 para el segundo y así sucesivamente.
Argumento (argument)
Información pasada a un método. Los argumentos se suelen llamar también parámetros. Un método que espera recibir argumentos debe contener una declaración de argumentos formales por cada argumento actual como parte de la cabecera del mismo. Cuando se invoca a un método, los valores de los argumentos actuales 8reales) se copia en los correspondientes argumentos formales.
Clase (clase)
Colección encapsulada de datos y operaciones que actúan sobre los datos. El concepto de clase es fundamental en programación orientada a objetos. Una clase consta de métodos y datos. Los métodos de una clase definen el conjunto de operaciones permitidas sobre los datos de una clase (sus atributos). Una clase puede tener muchas instancias de la clase u objetos.
Bloque (block)
Sentencias y declaraciones encerradas entre una pareja de llaves (apertura y cierre, ´{´ y ´}´. Por ejemplo, un cuerpo Libro Java 3 de una clase, es un bloque, al igual que el cuerpo de un método, Un bloque delimita un nivel de ámbito.
Diagrama de clases (class diagram).
Una representación gráfica construida utilizando una notación formal para visualizar y documentar las relaciones entre clases de un sistema.

Encapsulamiento, encapsulación (encapsulation)
Localización y protección de las características internas y estructura de un objeto. Combinación de métodos y datos en una única estructura de datos. En Java se conoce como clase.
Herencia (inheritance)
Una relación entre clases en que una subclase se extiende desde una superclase.
Interprete (Interpreter)
Software que interpreta y ejecuta bytecode de Java. La máquina virtual Java (JVM) es un intérprete de bytecodes de Java que proporciona una emulación de software de un procesador de máquina.
Parámetro formal (formal parameter)
Declaración de una variable parámetro en una lista de parámetros de un método.
Tipo de datos (data type)
Los tipos de datos se utilizan para definir variables. Java soporta los tipos de datos primitivos y tipos de datos objeto.
Prueba/ probar (test)
En términos de programación, la actividad de verificación sistemática de que un programa funciona correctamente.
Código fuente (source code)
Texto de un programa antes de ser compilado. El texto se crea y edita utilizando en editor ordinario y contiene caracteres normales, legibles. El código fuente ser utiliza para las personas para describir programas y sus componentes han de ser lo más legibles y comprensibles posibles.
Método (method)
Una colección de sentencias que se agrupan juntos para ejecutar una operación.
Variable estática (static variable)
Una relación entre clases en que una subclase se extiende desde una superclase.
Subclase (subclass)
Una clase que hereda o se extiende de una superclase.
Diseño orientado a objetos OOD (object.oriented design)
Diseño realizado en términos de objetos, clases y selecciones de clases.
Interfaz (interface)
Una interfaz se trata como una clase especial de Java. Cada interface se compila en un archivo independiente de bytecode, tal como una clase ordinaria. No se puede crear una instancia de la interfaz. La estructura de una interfaz Java es similar al de una clase abstracta en la que se puede tener datos y métodos. Los datos, sin embargo, deben ser constantes y los métodos pueden tener sólo declaraciones sin implementación. En Java existe sólo herencia simple y una clase puede heredar de una supereclase. Esta restricción se puede superar por el uso de una interfaz.
Instancia (instance)
Objeto de una clase
IDE (integrated development)
Software para ayudar a los programadores a escribir código eficientemente.
Implementación (implementation)
La actividad de escribir, compilar, probar y depurar el código de un programa.

Diseño (diseño)
Actividad de definir como se debe estructurar e implementar un programa.
Depuración (debugging)
Proceso de encontrar, fijar y eliminar errores en un programa. Para estas tareas se suele utilizar una herramienta de programación conocida como depurador
Compilación (compiling)
Nombre dado al proceso de traducción del código fuente a bytecodes
Bytecode (códigos de byte)
Resultado de la compilación del código fuente Java. La JVM (Java Virtual Machine) interpreta los bytecodes con la finalidad de ejecutar un programa Java. El bytecode es independiente de la máquina y se puede ejecutar en cualquier máquina que tenga un entorno de ejecución. Los bytecodes se almacenan en archivos class.
Abstraction (abstracción)
Propiedad y/o técnica de software que oculta los detalles de la implementación. Java soporta abstracción de clases y abstracción de métodos. La abstracción de métodos se define separando el uso de un método sin conocer como está implementado ese método. Si decide combinar la implementación, el programa cliente será afectado. De modo similar la abstracción de clases oculta la implementación de la clase del cliente.
Acoplamiento (coupling)
Medida del grado en el que un objeto o componente depende de otro. Bajo acoplamiento minimiza las dependencias y es una indicación de un buen diseño.
Agregación (aggregation)
Relación en la que un objeto se compone o está construido de uno o más objetos, de modo que la colección completa representa un todo. Las relaciones de agregación se especifican entre clases y se reflejan en instancias de objetos.
Excepción (exception)
Un suceso (evento) no previsto que indica que un programa ha fallado en alguna forma. Las excepciones se representan por objetos excepción en java. Las excepciones se manejan con un bloque de sentencias try/catch.
Expresión (expresión)
Una subparte de una sentencia que representa un valor. Por ejemplo, la expresión aritmética´2+5´ representa el valor 7. En Java, cualquier construcción sintáctica legal que represente un valor es una expresión.
Final (final)
Modificador de clases, datos, métodos y variables locales. Una clase final no se puede extender, un dato final o variable local es una constante y un método final no se puede anular (sustituir) en una subclase.
Formal parameter (parámetro formal)
Parámetros definidos en la signatura o declaración del método.
Fuente del suceso (event source)
El objeto que genera el suceso.
Función (function)
Construcción matemática a la que se pueden aplicar valores y que devuelve un resultado.
Entidad relación (entity relationship)
es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades.
Caso de usos (use cases)

es una forma de diagrama de comportamiento UML mejorado. El Lenguaje de Modelado Unificado (UML), define una notación gráfica para representar casos de uso llamada modelo de casos de uso. UML no define estándares para que el formato escrito describa los casos de uso, y así mucha gente no entiende que esta notación gráfica define la naturaleza de un caso de uso; sin embargo, una notación gráfica puede solo dar una vista general simple de un caso de uso o un conjunto de casos de uso.
Unsigned (Unsigned)
os campos que contienen números enteros admiten el parámetro UNSIGNED que implica que no admita signos por lo que solo aceptaría enteros positivos.
Zerofill (Zerofill)
Todos los campos numéricos admiten el parámetro ZEROFILL cuya función es completar el campo con ceros a la izquierda hasta su longitud máxima.
Inclusión (include)
Es una forma de interacción o creación, un caso de uso dado puede "incluir" otro caso de uso. El primer caso de uso a menudo depende del resultado del caso de uso incluido.
Extensión (extend)
Es otra forma de interacción, un caso de uso dado (la extensión) puede *extender* a otro. Esta relación indica que el comportamiento del caso de la extensión se utiliza en casos de uso, un caso de uso a otro caso siempre debe tener extensión o inclusión.

Formato de alto nivel

Casos de uso

Caso de uso: Subir colecciones de software.
Actores: Usuario administrador.
Tipo: Primario.
Descripción: El usuario registra cada tipo de software rellenando los datos del mismo.

Caso de uso: Gestionar configuraciones.
Actores: Usuario administrador.
Tipo: Primario.
Descripción: El usuario debe actualizar y categorizar cada tipo de software conforme los datos del mismo lo requieran.

Caso de uso: Configuración categorías.
Actores: Usuario administrador.
Tipo: Secundario.
Descripción: El usuario debe categorizar cada tipo de software conforme el tipo de servicio que realice la plataforma.

Caso de uso: Configuración métricas.
Actores: Usuario administrador.
Tipo: Secundario.
Descripción: El usuario debe crear cada tipo de métricas para que realicen las calificaciones otros usuarios.

Caso de uso: Buscar.
Actores: Usuario básico, usuario premium.
Tipo: Primario.
Descripción: El usuario puede buscar cada tipo de software, por medio del nombre del software o por la categoría a la cual pertenece.

Caso de uso: Búsqueda software.
Actores: Usuario básico, usuario premium.
Tipo: Secundario.
Descripción: El usuario puede buscar todo tipo de software, apareciendo en forma ordena.

Caso de uso: Búsqueda categoría.
Actores: Usuario básico, usuario premium.
Tipo: Secundario.
Descripción: El usuario puede buscar cada tipo de software conforme a la categoría a la cual pertenezca, apareciendo en forma ordena.

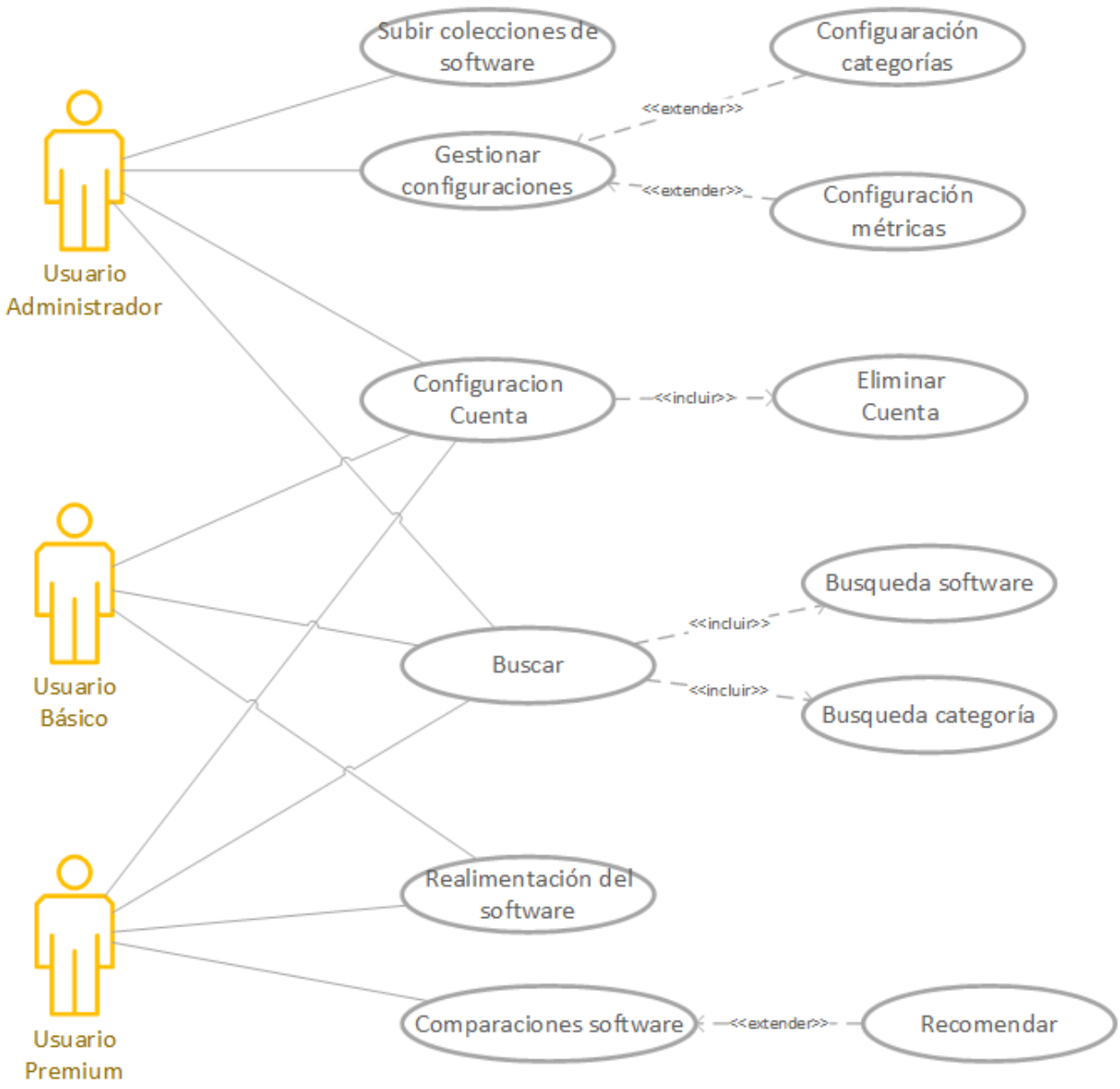
Caso de uso: Realimentación del software.
Actores: Usuario básico, usuario premium.
Tipo: Primario.
Descripción: El usuario puede realizar calificaciones de cada tipo de software, como también una descripción de mismo.

Caso de uso: Comparaciones software.
Actores: Usuario premium.
Tipo: Primario.
Descripción: El usuario puede realizar comparaciones de cada tipo de software, con una restricción de 2 a 5 aplicaciones.

Caso de uso: Configuración Cuenta
Actores: Usuario básico, usuario premium, usuario administrador
Tipo: Primario
Descripción: El usuario puedo realizar modificaciones a su cuenta.

Caso de uso: Eliminar Cuenta
Actores: Usuario básico, usuario premium, usuario administrador
Tipo: Secundario
Descripción: El usuario puedo eliminar su cuenta.

Diagrama


Subir colecciones de software
Configuaración categorías
<<extender>>
Gestionar configuraciones
<<extender>>
Configuración métricas
Usuario Administrador
Configuracion Cuenta
<<incluir>>
Eliminar Cuenta
Busqueda software
<<incluir>>
Buscar
<<incluir>>
Busqueda categoría
Usuario Básico
Realimentación del software
Comparaciones software
<<extender>>
Recomendar
Usuario Premium

Formato esencial expandidos

Caso de uso: Subir colecciones de software.
Actores: Usuario administrador.
Propósito: Subir cada tipo de software a la plataforma.
Tipo: Primario.
Descripción: El usuario registra cada tipo de software rellenando los datos del mismo.
Curso normal de los eventos:
1. El usuario introduce los datos del software.
2. El sistema buscara entre los softwares si existe un software con el mismo identificador.
3. El sistema informa al usuario que el software se ha registrado satisfactoriamente.
Cursos alternos: El sistema informa al usuario que ya existe un software con el mismo identificador en el sistema.

Caso de uso: Gestionar configuraciones.
Actores: Usuario administrador.
Propósito: Categorizar cada software por sus atributos.
Tipo: Primario.
Descripción: El usuario debe actualizar y categorizar cada tipo de software conforme los datos del mismo lo requieran.
Curso normal de los eventos:
1. El usuario categoriza los atributos del software.
2. Seleciona que configuración desea hacer.
3. Recurre a la ventana emergente para ingresar la configuración.
Cursos alternos: Ninguno.

Caso de uso: Configuración categorías.
Actores: Usuario administrador.
Propósito: Categorizar cada software.
Tipo: Secundario.
Descripción: El usuario debe categorizar cada tipo de software conforme el tipo de servicio que realice la plataforma.
Curso normal de los eventos:
1. El sistema verifica si el software pertenece a varias categorías de acuerdo a sus atributos.
2. El usuario selecciona las categorías que prefiera de las que el sistema proporciona.
3. Seleciona el orden más conveniente en el cual aparecerá el software.
4. El sistema informa al usuario que se ha guardado la configuración satisfactoriamente.
Cursos alternos: Si el sistema no pertenece a ninguna categoría predetermina, se solicitara al usuario ingresar la categoría a la que pertenece.

Caso de uso: Configuración métricas.
Actores: Usuario administrador.
Propósito: Realizar métricas.
Tipo: Secundario.
Descripción: El usuario debe crear cada tipo de métricas para que realicen las calificaciones otros usuarios.
Curso normal de los eventos:
1.El usuario llenara todos los campos para crear una métrica.
2.El sistema buscara entre las métricas si existe una métrica con el mismo identificador.
3. El sistema informa al usuario que se ha guardado la configuración satisfactoriamente.
Cursos alternos: El sistema notificara al usuario que ya existe una métrica con el mismo identificador en el sistema.

Caso de uso: Buscar.
Actores: Usuario básico, usuario premium.
Propósito: Buscar el software o categoría que el usuario desee.
Tipo: Primario.
Descripción: El usuario puede buscar cada tipo de software, por medio del nombre o categoría del software que desee.
Curso normal de los eventos:
1.El usuario introduce los datos del software.
2.El sistema utilizara los buscadores específicos para los datos ingresados por el software
3. El sistema desplegara las opciones de forma ordena.
Cursos alternos: El sistema informa al usuario que no existe un software o categoría con esas descripciones.

Caso de uso: Búsqueda software.
Actores: Usuario básico, usuario premium.
Propósito: Buscar el software que el usuario desee.
Tipo: Secundario.
Descripción: El usuario puede buscar todo tipo de software, apareciendo en forma ordenada.
Curso normal de los eventos:
1. El sistema buscara entre la base de softwares si existe un software con los mismos datos que el usuario introdujo.
2.El sistema desplegara las opciones de forma ordena.
Cursos alternos: El sistema informa al usuario que no existe un software con esas descripciones.

Caso de uso: Búsqueda categoría.
Actores: Usuario básico, usuario premium.
Propósito: Buscar la categoría que el usuario desee.
Tipo: Secundario.
Descripción: El usuario puede buscar cada tipo de software conforme a la categoría a la cual pertenezca, apareciendo en forma ordenada.
Curso normal de los eventos:
1. El sistema buscara entre la base de categorías si existe una categoría con los mismos datos que el usuario introdujo.
2. El sistema desplegara las opciones de forma ordena.
Cursos alternos: El sistema informa al usuario que no existe una categoría con esas descripciones.

Caso de uso: Realimentación del software.
Actores: Usuario básico, usuario premium.
Propósito: Buscar la categoría que el usuario desee.
Tipo: Primario.
Descripción: El usuario puede realizar calificaciones de cada tipo de software, como también una descripción del mismo.
Curso normal de los eventos:
1. El usuario seleccionara la métrica a calificar.
2. El usuario calificara el software del 1-5 estrellas.
3. El usuario comentara la aplicación, llenando cada uno de los campos.
4. El sistema notificara al usuario que se han guardado la realimentación exitosamente.
Cursos alternos: El sistema informa al usuario que debe de llenar todos los campos.

Caso de uso: Comparaciones software.
Actores: Usuario premium.
Propósito: Comparar los softwares que el usuario desee.
Tipo: Primario.
Descripción: El usuario puede realizar comparaciones de cada tipo de software, con una restricción de 2 a 5 aplicaciones.
Curso normal de los eventos:
1. El usuario seleccionara los softwares a comparar.
2. El sistema preguntara al usuario si está seguro de la elección.
3. El sistema desplegara los campos, gráficos y plataformas para la comparación realizada.
4. El usuario seleccionara publicar o volver a realizar una comparación.
5. El sistema notificara al usuario que la comparación se ha publicado con éxito.
6. El usuario podrá seleccionara visualizar comparaciones o realizar recomendaciones.
Cursos alternos:
1. El sistema informara al usuario que debe deben ser al menos 2 softwares para realizar la comparación.
2. El sistema informara al usuario que solo se permiten 5 softwares en una comparación.

Caso de uso: Configurar Cuenta

Actores: Usuario básico, usuario premium, usuario administrador.

Propósito: Realizar cambios en la cuenta personal del usuario.

Tipo: Primario

Descripción: El usuario puedo modificar su cuenta.

Curso normal de los eventos:

1. El sistema retornara los datos del usuario.
2. El usuario llenara los campos que desee modificar.
3. El sistema validara los campos.
4. El sistema retornara mensaje exitoso al realizar los cambios.

Cursos alternos:

1. El sistema retornara que campo en específico no puede modificarse.
2. El sistema retornara mensaje de error por distinto formato de campo.

Caso de uso: Eliminar Cuenta

Actores: Usuario básico, usuario premium, usuario administrador.

Propósito: Dejar de utilizar la aplicación.

Tipo: Secundario.

Descripción: El usuario puedo eliminar su cuenta.

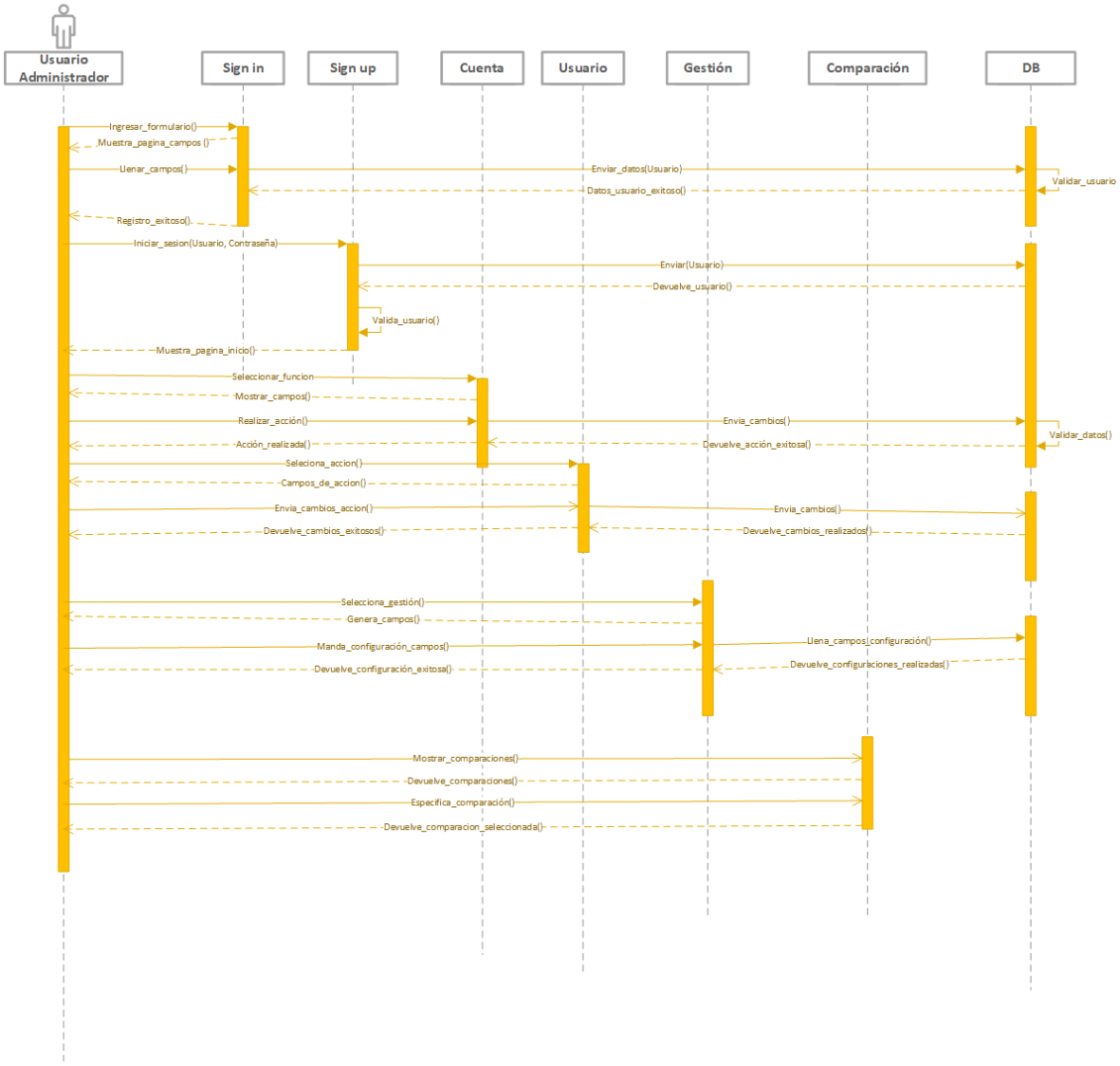
Curso normal de los eventos:

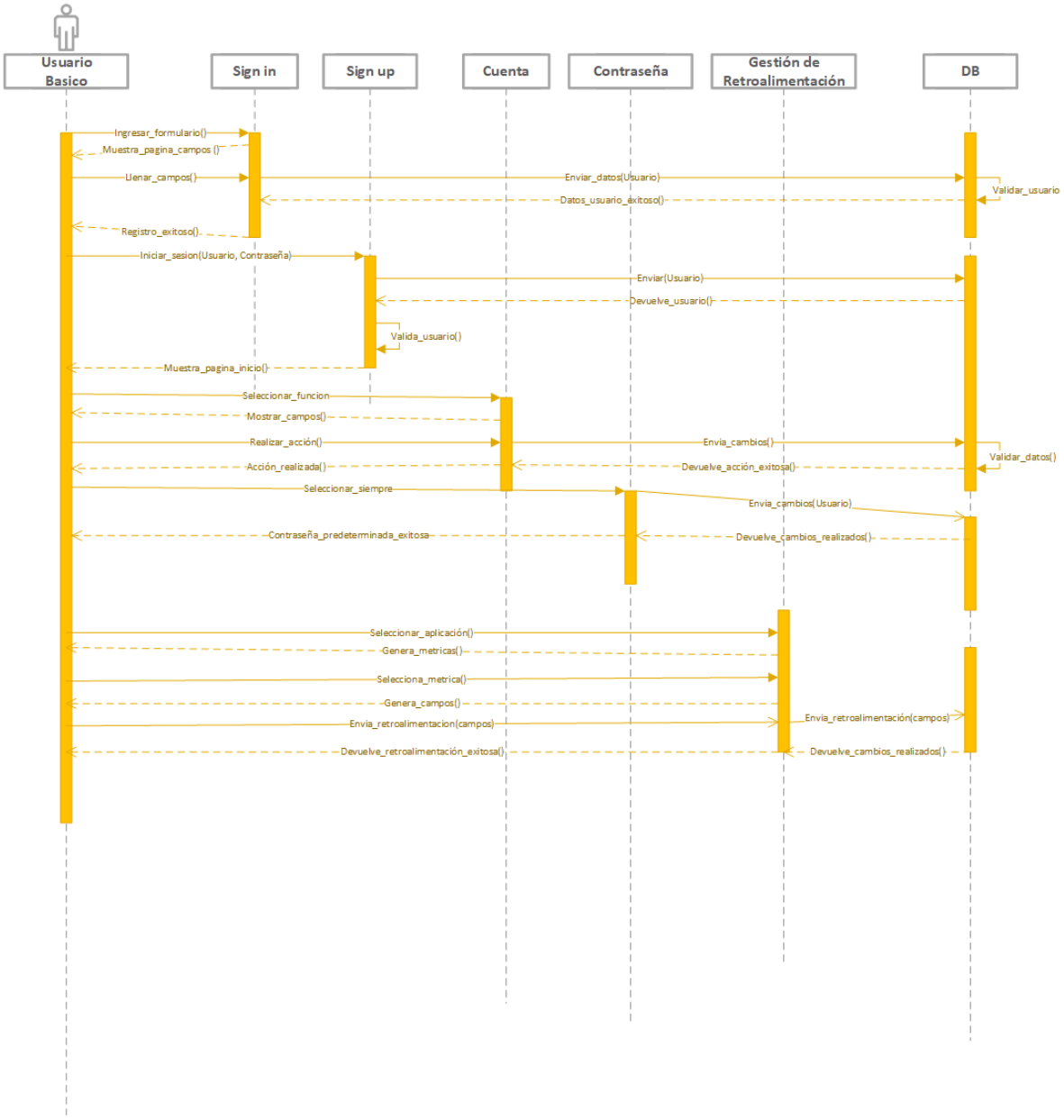
1. Seleccionar configurar cuenta.
2. Seleccionar eliminar cuenta.
3. El sistema eliminara la cuenta.
4. El sistema retornara al usuario al inicio de sesión.

Cursos alternos:

1. El sistema retornara falló en el comando a realizar.

Diagrama de secuencia




Usuario Basico
Sign in
Sign up
Cuenta
Contraseña
Gestión de Retroalimentación
DB
Ingresar_formulario()
Muestra_pagina_campos ()
Llenar_campos()
Enviar_datos(Usuario)
Validar_usuario
Datos_usuario_exitoso()
Registro_exitoso()
Iniciar_sesion(Usuario, Contraseña)
Enviar(Usuario)
Devuelve_usuario()
Valida_usuario()
Muestra_pagina_inicio()
Seleccionar_funcion
Mostrar_campos()
Realizar_acción()
Envia_cambios()
Validar_datos()
Acción_realizada()
Devuelve_acción_exitosa()
Seleccionar_siempre
Envia_cambios(Usuario)
Contraseña_predeterminada_exitosa
Devuelve_cambios_realizados()
Seleccionar_aplicación()
Genera_metricas()
Selecciona_metrica()
Genera_campos()
Envia_retroalimentacion(campos)
Envia_retroalimentación(campos)
Devuelve_retroalimentación_exitosa()
Devuelve_cambios_realizados()

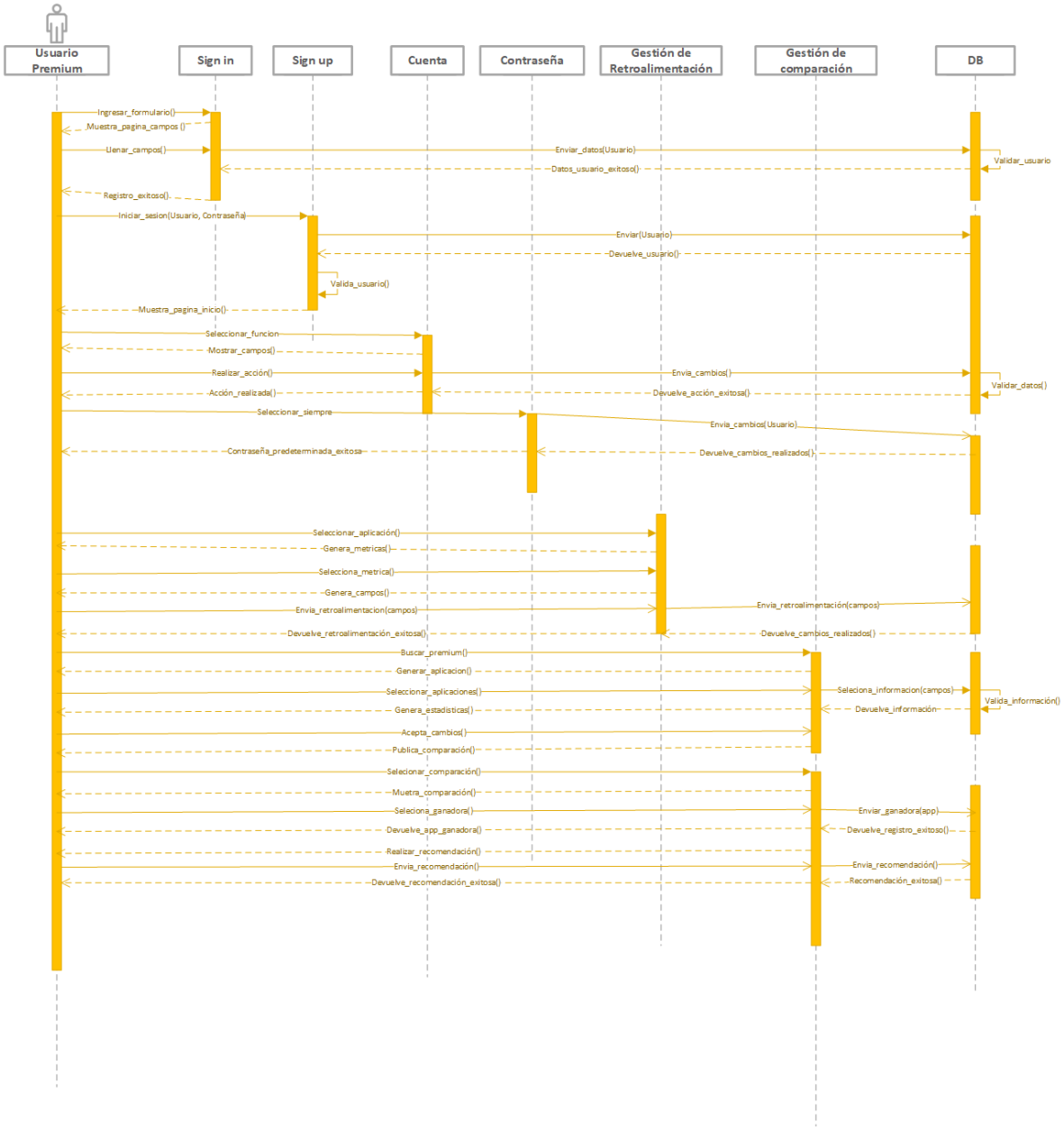
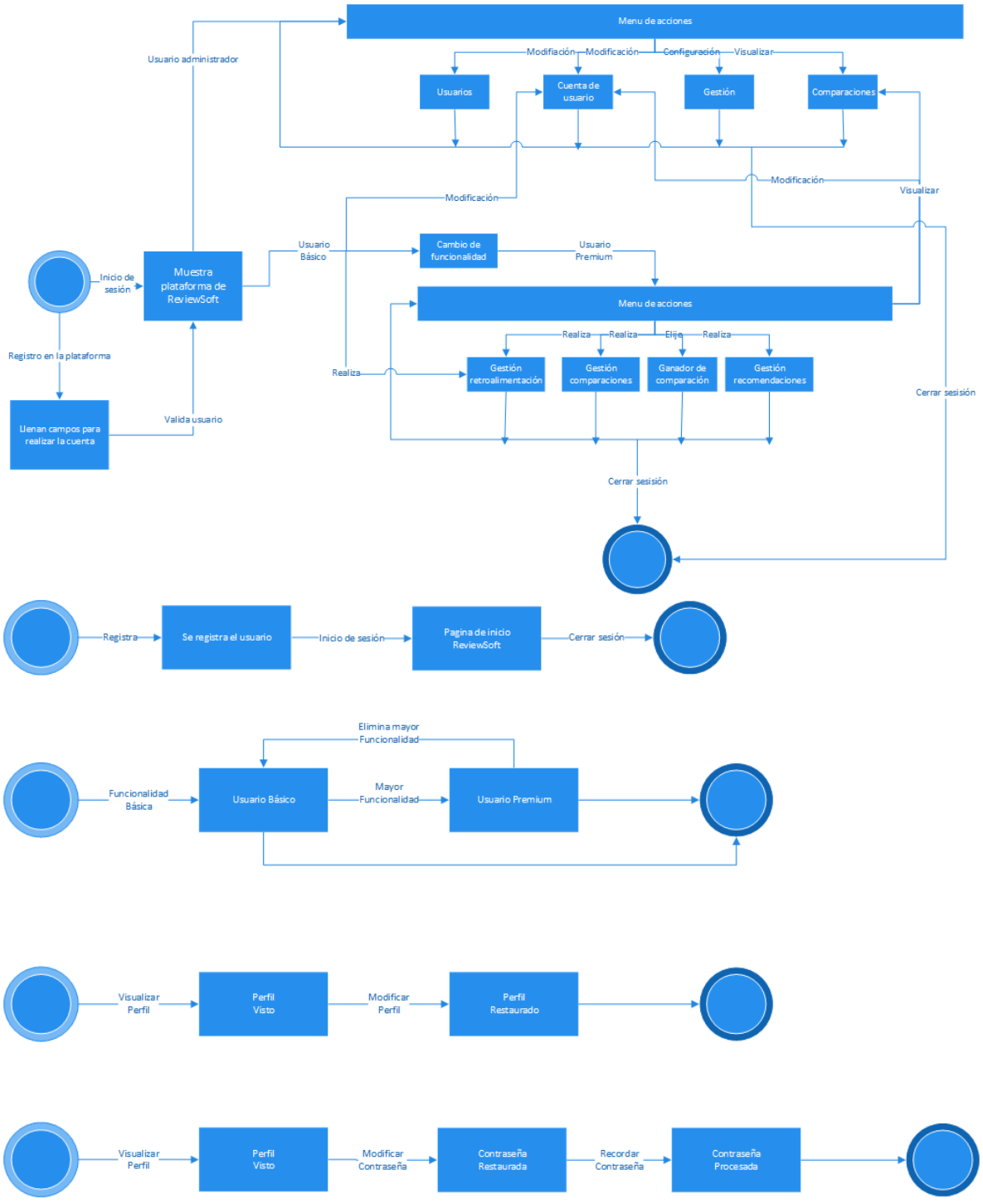
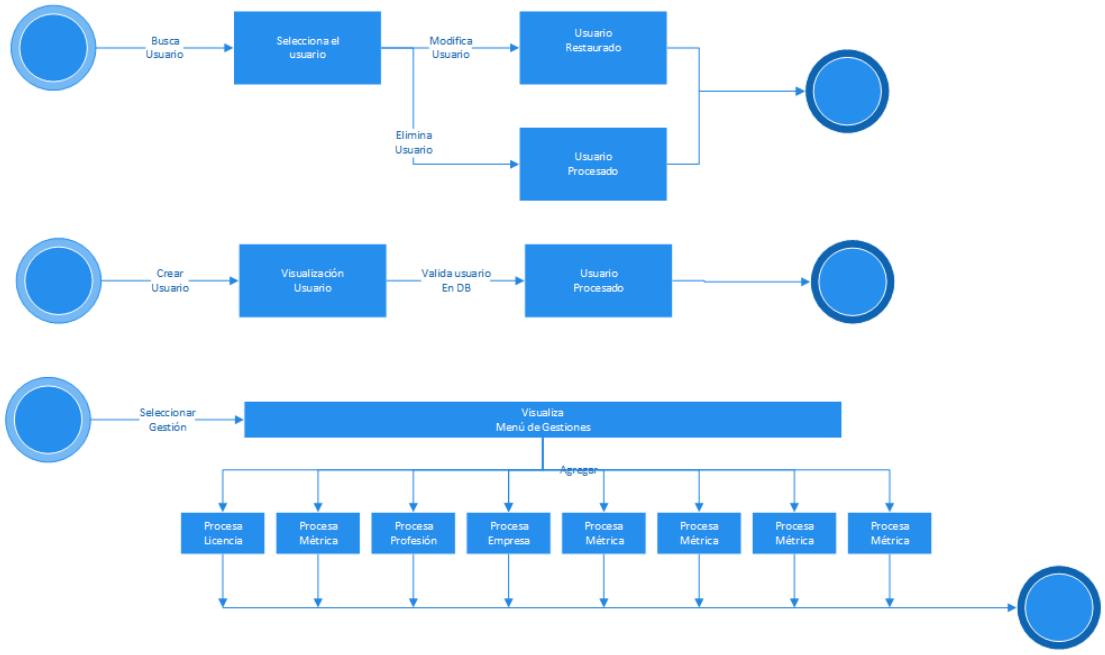

Usuario Premium
Sign in
Sign up
Cuenta
Contraseña
Gestión de Retroalimentación
Gestión de comparación
DB
Ingresar_formulario()
Muestra_pagina_campos ()
Llenar_campos()
Enviar_datos(Usuario)
Validar_usuario
Datos_usuario_exitoso()
Registro_exitoso().
Iniciar_sesion(Usuario, Contraseña)
Enviar(Usuario)
Devuelve_usuario()
Valida_usuario()
Muestra_pagina_inicio()
Seleccionar_funcion
Mostrar_campos()
Realizar_acción()
Envia_cambios()
Validar_datos()
Acción_realizada()
Devuelve_acción_exitosa()
Seleccionar_siempre
Envia_cambios(Usuario)
Contraseña_predeterminada_exitosa
Devuelve_cambios_realizados()
Seleccionar_aplicación()
Genera_metricas()
Selecciona_metrica()
Genera_campos()
Envia_retroalimentacion(campos)
Envia_retroalimentación(campos)
Devuelve_retroalimentación_exitosa()
Devuelve_cambios_realizados()
Buscar_premium()
Generar_aplicacion()
Seleccionar_aplicaciones()
Seleciona_informacion(campos)
Valida_información()
Genera_estadisticas()
Devuelve_información
Acepta_cambios()
Publica_comparación()
Selecionar_comparación()
Muetra_comparación()
Seleciona_ganadora()
Enviar_ganadora(app)
Devuelve_app_ganadora()
Devuelve_registro_exitoso()
Realizar_recomendación()
Envia_recomendación()
Envia_recomendación()
Devuelve_recomendación_exitosa()
Recomendación_exitosa()

Diagrama de estado

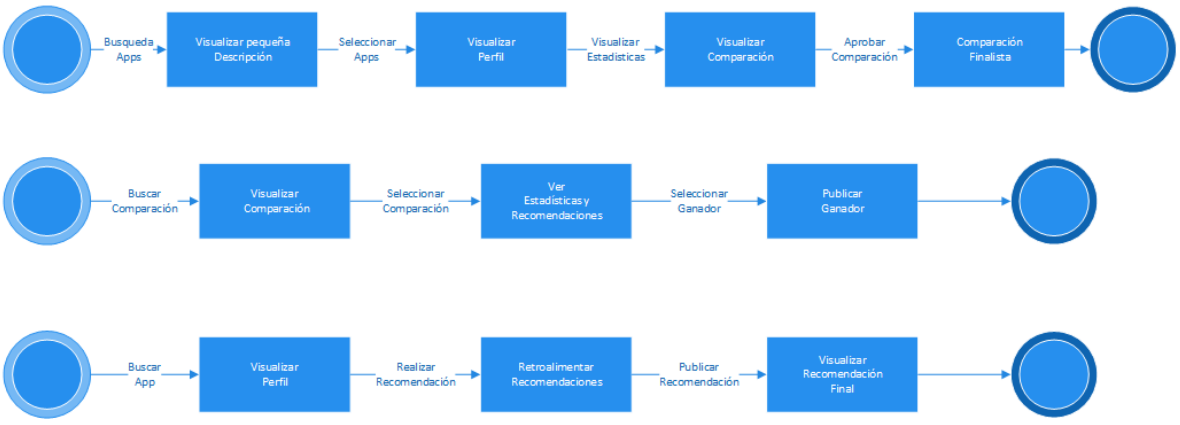


Usuario Administrador

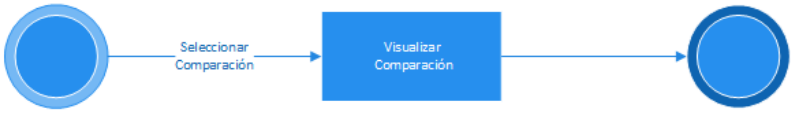


Usuario Básico y Premium

Usuario Premium



Usuario Premium Y
Administrador



DB

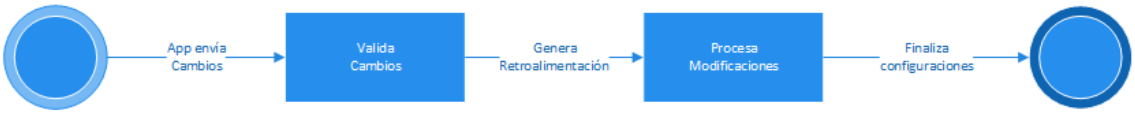
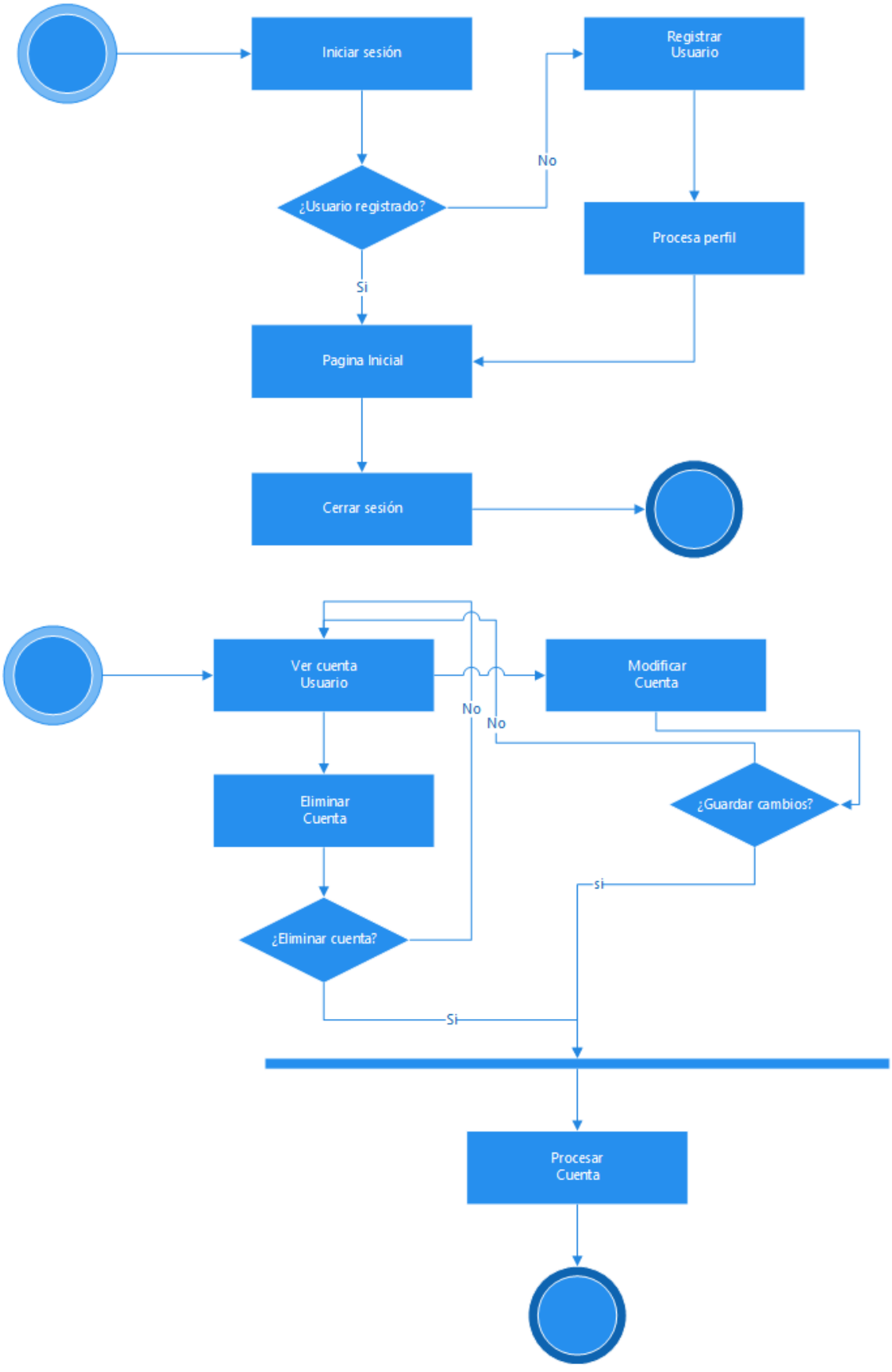


Diagrama de actividades



Usuario Administrador

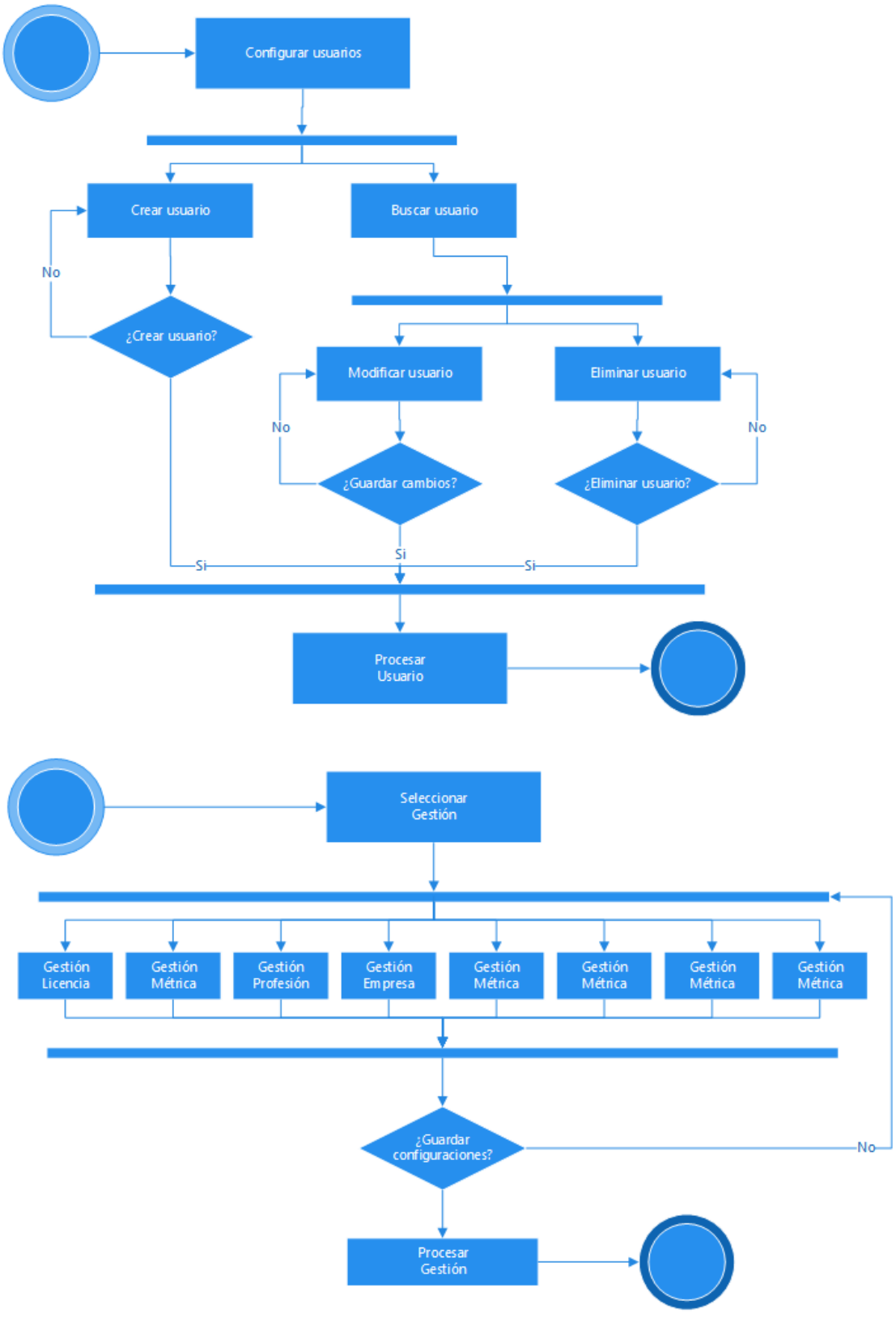


Diagrama de clases de diseño

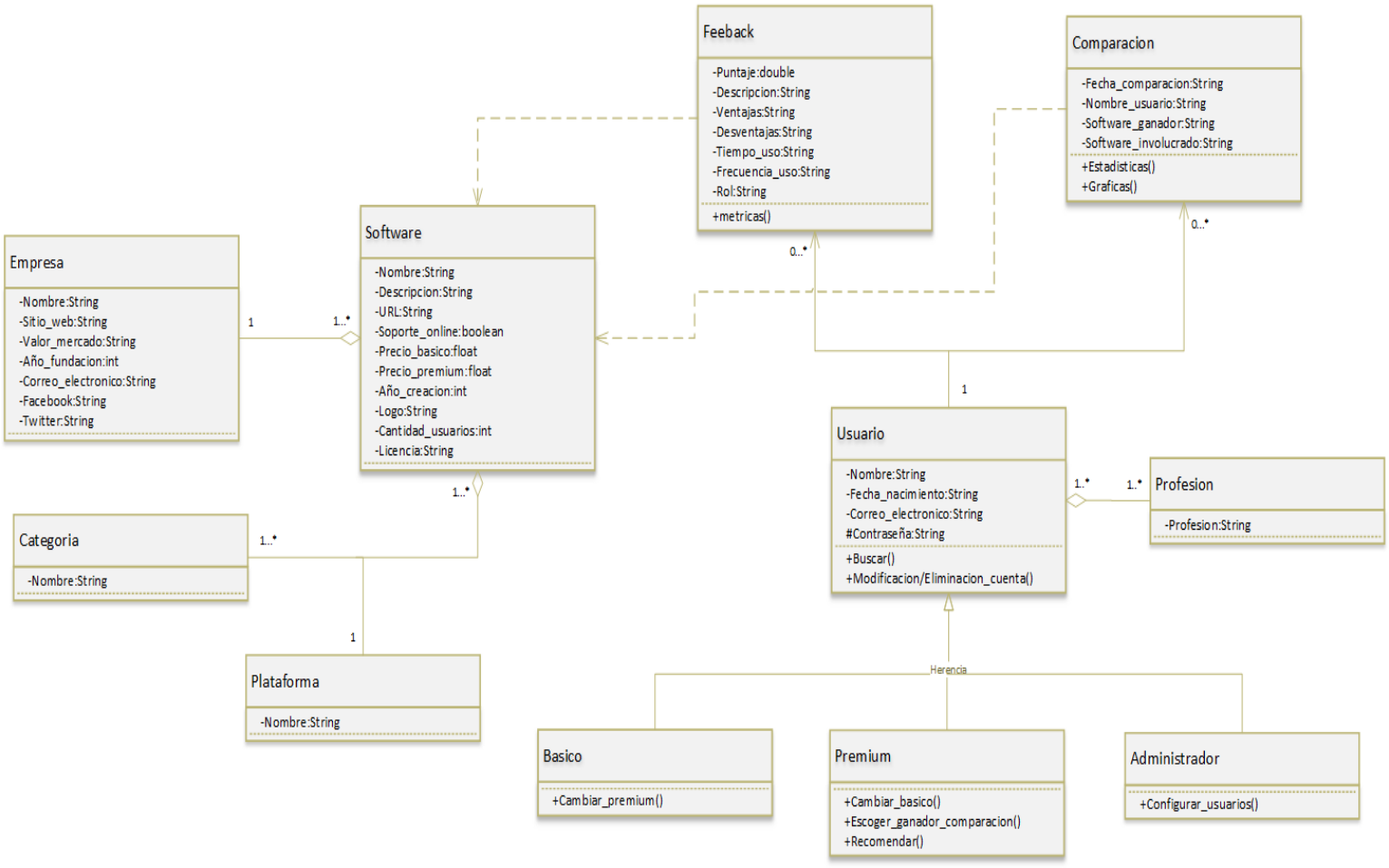
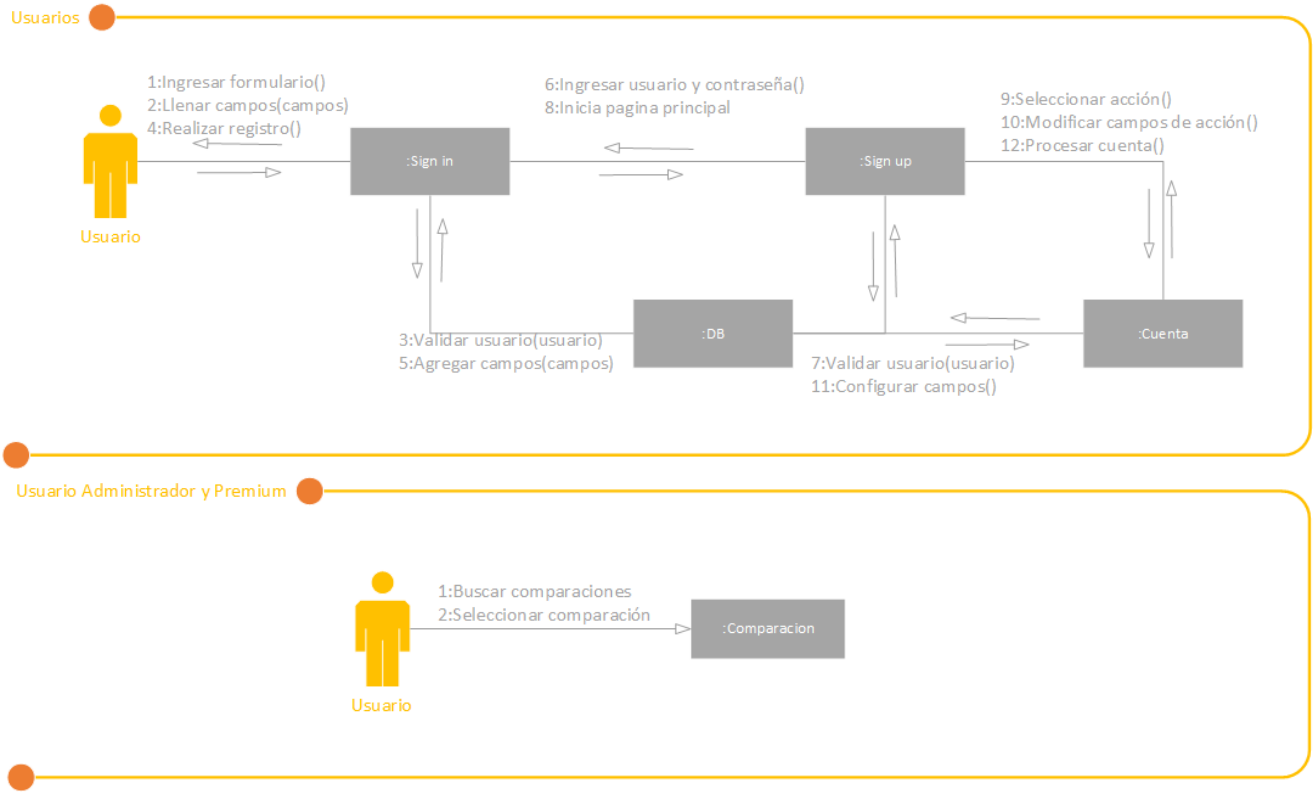
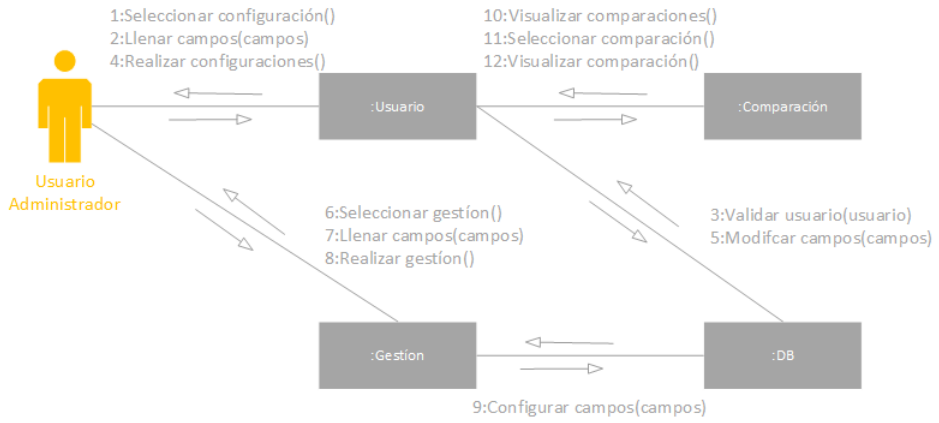


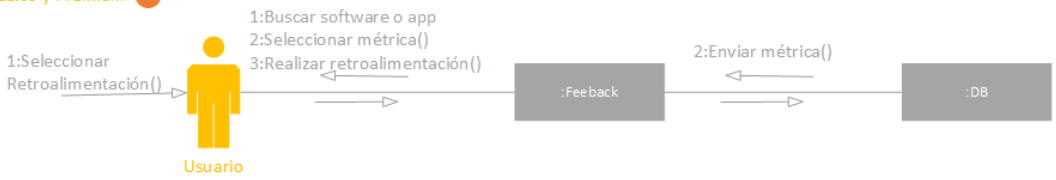
Diagrama de colaboración



Usuario administrador



Usuario Básico y Premium



Usuario Premium