Ciclo de vida del futbol.

Enunciado

Nuestra aplicación consiste en la gestión las finanzas de una serie de jugadores de equipos de futbol, las nacionalidades, sus representantes y patrocinadores principales.

PROPUESTA DE SOLUCIÓN.

Determinación y justificación del ciclo de vida que vamos a seguir. En principio, disponemos de 4 ciclos de vida:

- Cascada.
- Cascada con realimentación.
- Iterativo incremental.
- Espiral.

Cascada: es prácticamente imposible realizar un proyecto de software sin volver atrás en las fases del ciclo de vida, por lo que este modelo lo descartamos.

Nos inclinaremos por un modelo de cascada con retroalimentación ya que un modelo iterativo incremental no nos conviene debido a que no tendremos interacción con el cliente.

ANÁLISIS

En base a lo expuesto, la aplicación deberá satisfacer los siguientes:

Requisitos funcionales pedidos:

- R1: Generar un listado de los jugadores que pertenecen a un determinado equipo.
- R2: Generar un listado de los jugadores que son representados por el mismo representante.
- R3: Insertar un nuevo jugador.
- R4: Insertar un nuevo equipo.
- R5: Insertar un nuevo representante.
- R6: Insertar un nuevo patrocinador.
- R7: Insertar un nuevo país.

Requisitos no funcionales:

- Calculo de los fondos de un equipo teniendo en cuenta el salario del jugador a fichar, la comisión de su representante y el dinero que ofrece el patrocinador o patrocinadores.

Existen algunos requisitos funcionales no pedidos expresamente pero que el cliente podría asumir que ha pedido al trabajar con esas informaciones:

- R8: Generar listado de los jugadores que han sido internacionales alguna vez.
- R9: Calculo del número de equipos en los que aparece un patrocinador principal.

DISEÑO

Diseño de datos: A continuación indicamos los almacenes de información que necesitaremos para satisfacer los requisitos. En nuestro caso, necesitaremos almacenar la siguiente información:

- A1: Jugadores: identificador, nombre, dorsal, representante, equipo, nacionalidad, ha sido internacional, salario, etc.
- A2: Equipos: identificador, nombre, nacionalidad, fondos, patrocinador principal, etc.
- A3: Nacionalidades: identificador, nombre, etc.
- A4: Patrocinadores: identificador, nombre, dinero que inyecta, etc.
- A5: Representantes: identificador, nombre, comisión, etc.

Para comprobar si con estos almacenes de información podemos satisfacer todos los requisitos los representamos en una tabla (técnica matricial).

VERIFICACION CUMPLIMINETO REQUISITOS	R1: lista de jugadores de un determinado equipo	R2: lista de jugadores que son representados por el mismo representante.	R3: insertar un nuevo jugador	R4: insertar un nuevo equipo	R5: insertar un nuevo representante	R6: insertar un nuevo patrocinador
A1: Jugadores	X	X	X			
A2: Equipos	X			X		
A3: Nacionalidades						
A4: Patrocinadores						X
A5: Representantes		X			X	

VERIFICACION CUMPLIMINETO REQUISITOS	R7: insertar una nueva nacionalidad	R8: lista de jugadores que han sido internacionales	R9: cálculo del número de equipos que tienen el mismo patrocinador			
A1: Jugadores		X				
A2: Equipos			X			
A3: Nacionalidades	X					
A4: Patrocinadores			X			
A5: Representantes						

Diseño arquitectónico: establecemos los bloques que tendrá nuestra aplicación.

GESTION JUGADORES	GESTION EQUIPOS	INSERCCION DE DATOS
R1, R2, R8	R9	R3,R4,R5,R6,R7

Diseño de la interfaz: se deben diseñar las pantallas, intercambios de información,... de nuestra aplicación con otros sistemas de información y con los usuarios.

Diseño procedimental: se describen los programas que se deberían realizar.

Para cada programa indicar qué debe hacer, qué datos tiene de entrada y qué salidas (listados, comunicación con otros sistemas,...) debe generar.

Modelo Entidad/Relacion:

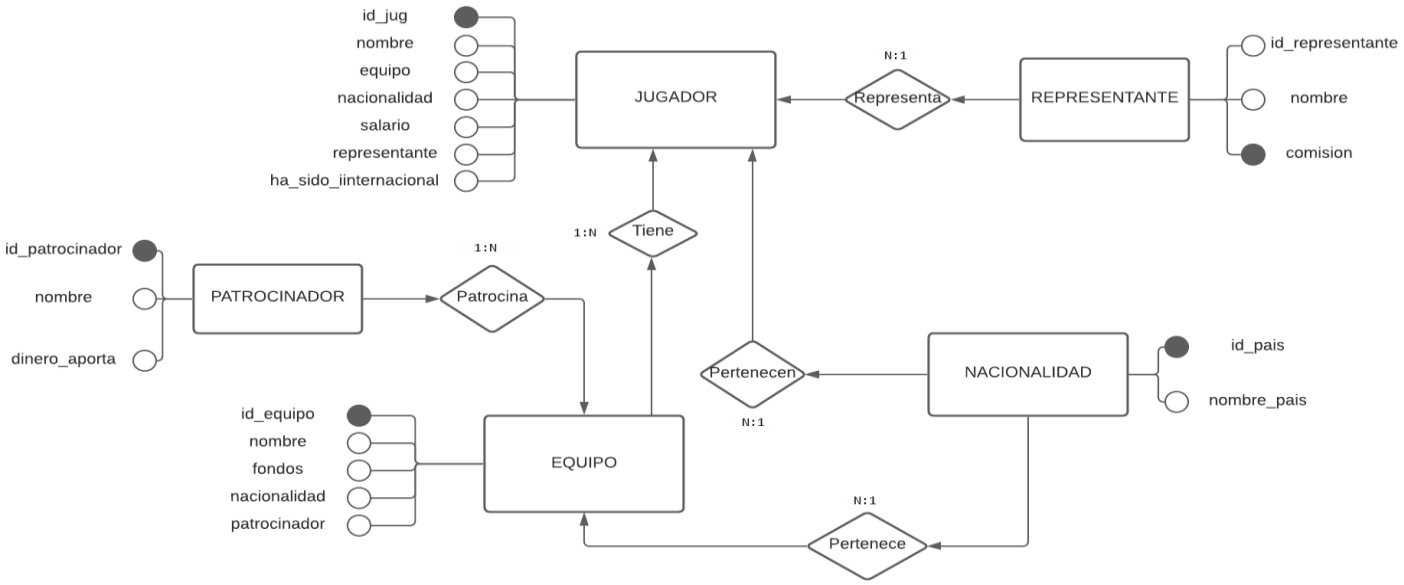
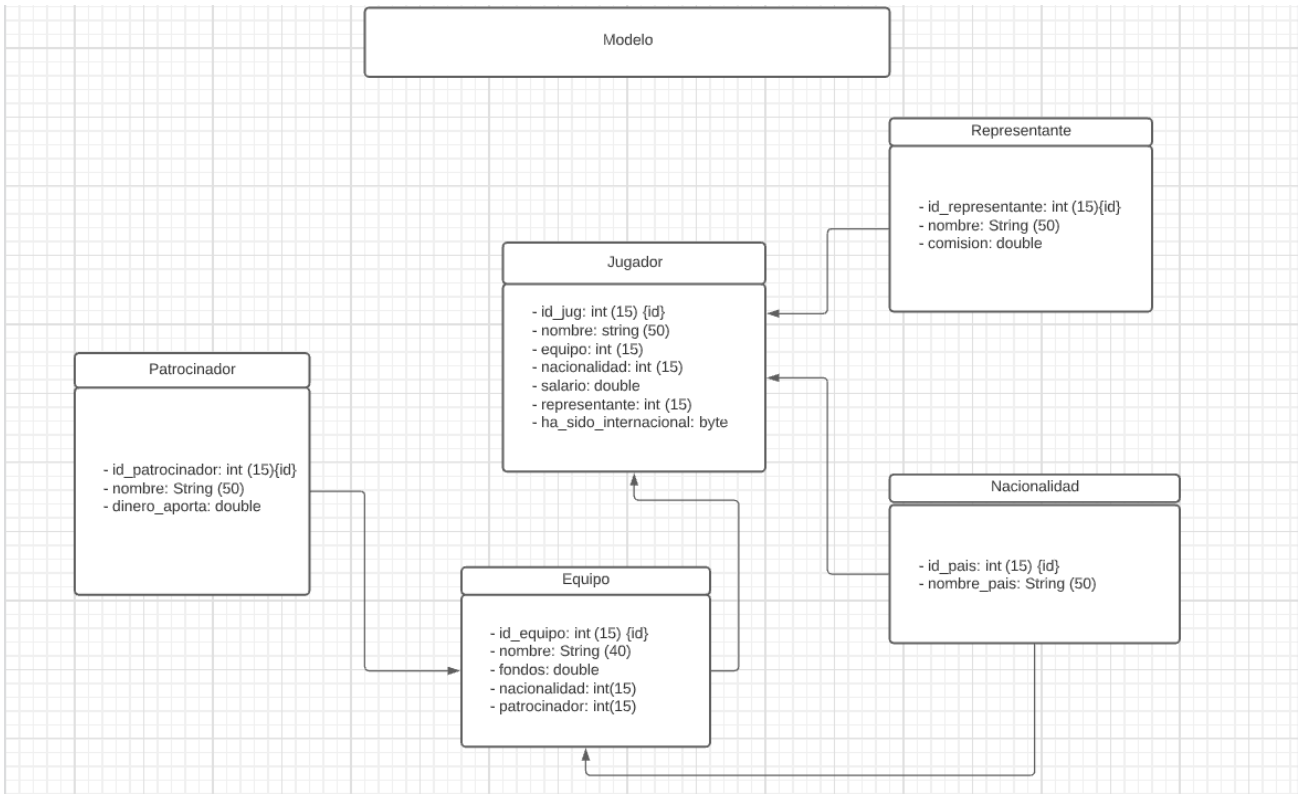
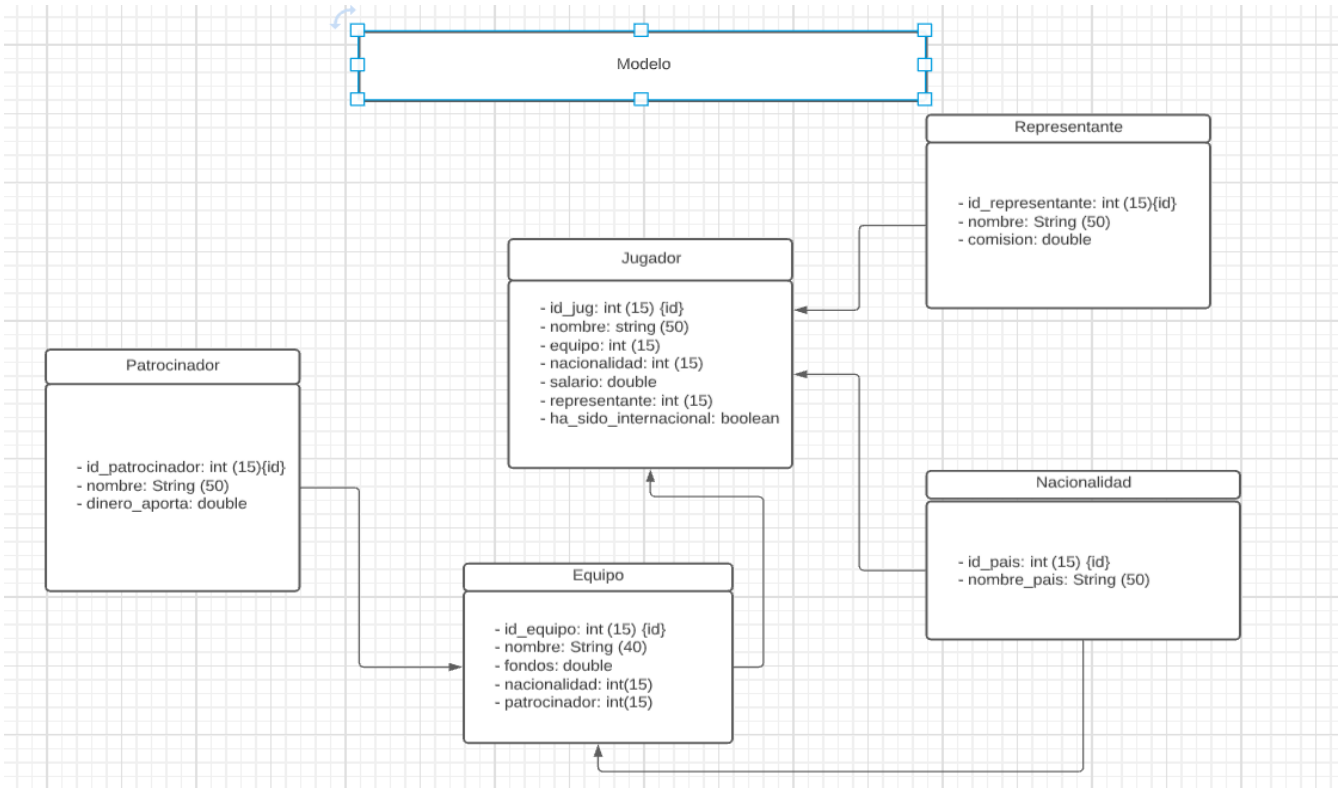
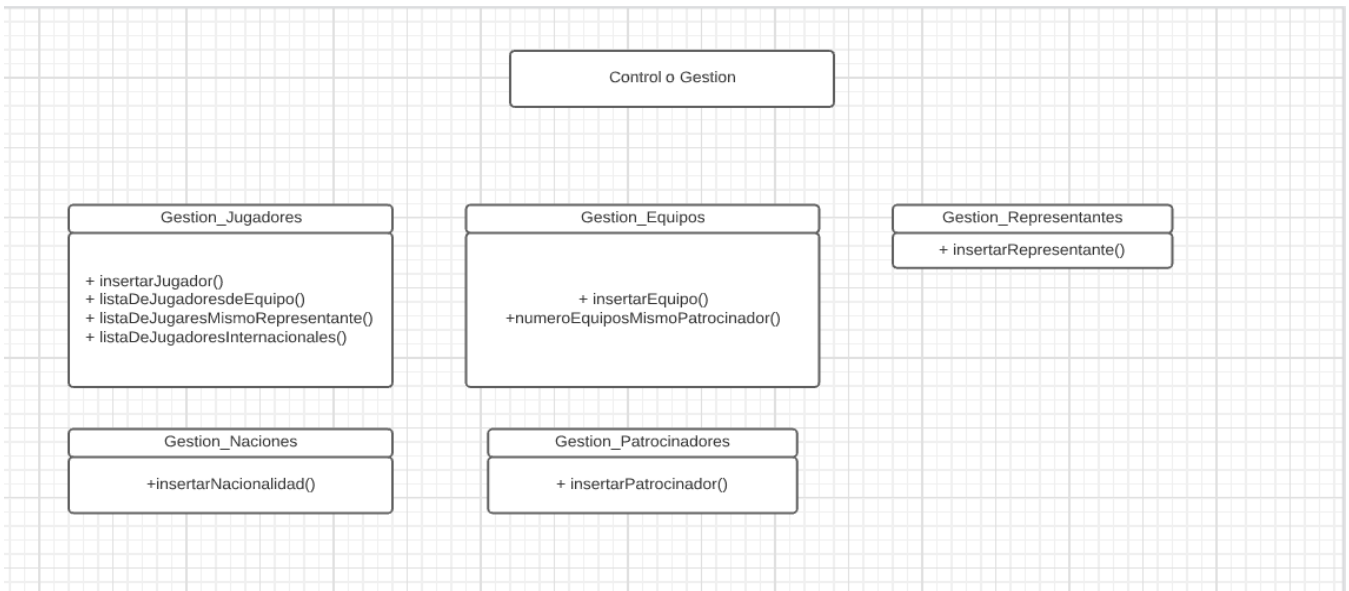


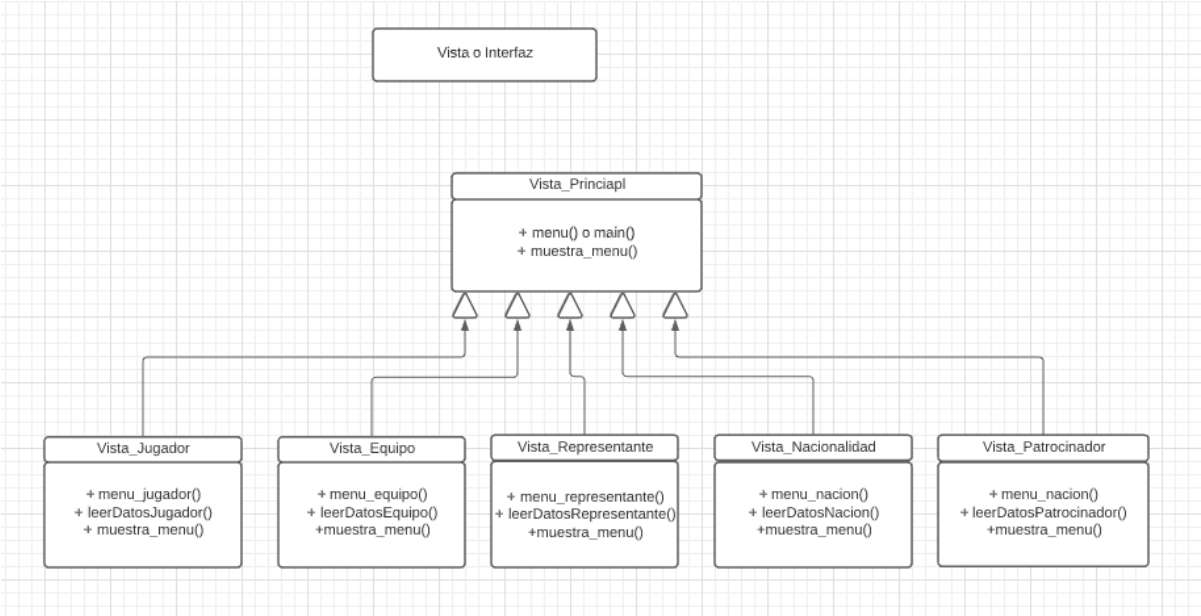
Diagrama de clases UML:

Tengo dos diagramas UML de Modelo debido a que una variable (ha sido internacional) podremos implementarla de tipo byte o de tipo boolean.




Modelo
Representante
- id_representante: int (15){id}
- nombre: String (50)
- comision: double
Jugador
- id_jug: int (15) {id}
- nombre: string (50)
- equipo: int (15)
- nacionalidad: int (15)
- salario: double
- representante: int (15)
- ha_sido_internacional: boolean
Patrocinador
- id_patrocinador: int (15){id}
- nombre: String (50)
- dinero_aporta: double
Nacionalidad
- id_pais: int (15) {id}
- nombre_pais: String (50)
Equipo
- id_equipo: int (15) {id}
- nombre: String (40)
- fondos: double
- nacionalidad: int(15)
- patrocinador: int(15)


Control o Gestion
Gestion_Jugadores
+ insertarJugador()
+ listaDeJugadoresdeEquipo()
+ listaDeJugaresMismoRepresentante()
+ listaDeJugadoresInternacionales()
Gestion_Equipos
+ insertarEquipo()
+numeroEquiposMismoPatrocinador()
Gestion_Representantes
+ insertarRepresentante()
Gestion_Naciones
+insertarNacionalidad()
Gestion_Patrocinadores
+ insertarPatrocinador()



REQUERIMIENTOS

Java, con JRE: JavaSE-14 y jdk-15.0.1 y usaremos MySQL para la base de datos.

Además necesitaremos un conector a la base de datos de SQL que será: mysql-connector-java-8.0.26.jar.

CODIFICACIÓN

El lenguaje de programación debe ser uno que esté generalizado y de uso libre. Usaremos java.

Determinaremos los algoritmos más apropiados para los módulos que se han definido en el diseño. En esta fase los implementaremos/programaremos.

Los compilamos y eliminamos los errores de sintaxis obteniendo el código objeto (bytecodes) para su ejecución con la máquina virtual java.

PRUEBAS

Tipos de pruebas unitarias:

- De caja negra: cada elemento de software lo consideramos como una caja negra a la que entran datos y se obtienen resultados. Se hacen pruebas fijándonos sólo en los datos que entran y los resultados que se obtienen.

- De caja blanca: se realizan pruebas en base a la estructura interna de cada elemento de software, considerando los bucles y selecciones que tiene.

Para cada módulo/programa de software haremos pruebas unitarias de caja blanca y de caja negra.

Una vez superadas, haremos pruebas de integración.

DOCUMENTACIÓN

Se generará:

- Guía técnica.
- Guía de uso.
- Guía de instalación.

EXPLOTACIÓN

Se instala la aplicación y configura en los equipos del cliente. Por último realizaremos la beta test en la instalación del cliente

MANTENIMIENTO

Se pacta con el cliente un contrato de mantenimiento. No obstante, suele haber un periodo de garantía durante el cual nos comprometemos a asumirlos errores que aparezcan en el normal funcionamiento de la aplicación.