* PEDRO BARAHONA
* ABEL CARRERA
* VALERIA GUTIERREZ

**10 de agosto de 2025**

Tarea03 - Patrones y Pruebas

HW03 - 6

Contenido

[Sección A : Patrones y Diagrama de Clases 3](#_Toc205749370)

[1. Factory Method (Creacional) 3](#_Toc205749371)

[Problema de diseño que ataca: 3](#_Toc205749372)

[Implementación en el código: 3](#_Toc205749373)

[Beneficios: 3](#_Toc205749374)

[2. Adapter (Estructural) 4](#_Toc205749375)

[Problema de diseño que ataca: 4](#_Toc205749376)

[Implementación en el código: 4](#_Toc205749377)

[Beneficios: 4](#_Toc205749378)

[3. State (Comportamental) 5](#_Toc205749379)

[Problema de diseño que ataca: 5](#_Toc205749380)

[Implementación en el código: 5](#_Toc205749381)

[Beneficios: 6](#_Toc205749382)

[4. Chain of Responsibility (Comportamental) 6](#_Toc205749383)

[Problema de diseño que ataca: 6](#_Toc205749384)

[Implementación en el código: 6](#_Toc205749385)

[Beneficios: 7](#_Toc205749386)

[Sección B: Plan de Pruebas 8](#_Toc205749387)

[Clase Usuario 8](#_Toc205749388)

[Clase: Cuenta 9](#_Toc205749389)

[Clase: Pronostico 9](#_Toc205749390)

[Clase: EventoDeportivo 10](#_Toc205749391)

[Clase: Incidencia 11](#_Toc205749392)

[Clase: Administrador 11](#_Toc205749393)

[Clase: MiembroQC 13](#_Toc205749394)

[Sección C: Implementación y Pruebas Unitarias 14](#_Toc205749395)

[1. CuentaTest (Clase: Cuenta) 14](#_Toc205749396)

[2. AdministradorTest (Clase: Administrador) 14](#_Toc205749397)

[3. EventoDeportivoTest (Clase: EventoDeportivo) 14](#_Toc205749398)

[4. IncidenciaTest (Clase: Incidencia) 15](#_Toc205749399)

[5. MiembroQCTest (Clase: MiembroQC) 15](#_Toc205749400)

[6. MiembroSoporteTest (Clase: MiembroSoporte) 15](#_Toc205749401)

[7. PronosticoTest (Clase: Pronostico) 16](#_Toc205749402)

[8. UsuarioTest (Clase: Usuario) 16](#_Toc205749403)

[Conclusión 16](#_Toc205749404)

# Sección A : Patrones y Diagrama de Clases

## Factory Method (Creacional)

### Problema de diseño que ataca:

* **Creación compleja de objetos**: El sistema necesita crear diferentes tipos de pronósticos (resultado del partido, marcador exacto, rendimiento de jugadores) y diferentes tipos de notificaciones (email, SMS, push notifications) de manera flexible.

### Implementación en el código:

* **Factory para Pronósticos**: Crear una interfaz IPronosticoFactory con métodos como crearPronosticoResultado(), crearPronosticoMarcador(), crearPronosticoRendimiento(). Esto permitiría crear pronósticos especializados según el tipo de evento deportivo.
* **Factory para Notificaciones**: Implementar INotifierFactory con métodos para crear diferentes tipos de notificadores (EmailNotifier, SMSNotifier, PushNotifier) según las preferencias del usuario o el tipo de mensaje.

Diagrama, Dibujo de ingeniería

El contenido generado por IA puede ser incorrecto.

### Beneficios:

* Flexibilidad para agregar nuevos tipos de pronósticos sin modificar el código existente
* Encapsulación de la lógica de creación
* Facilita testing

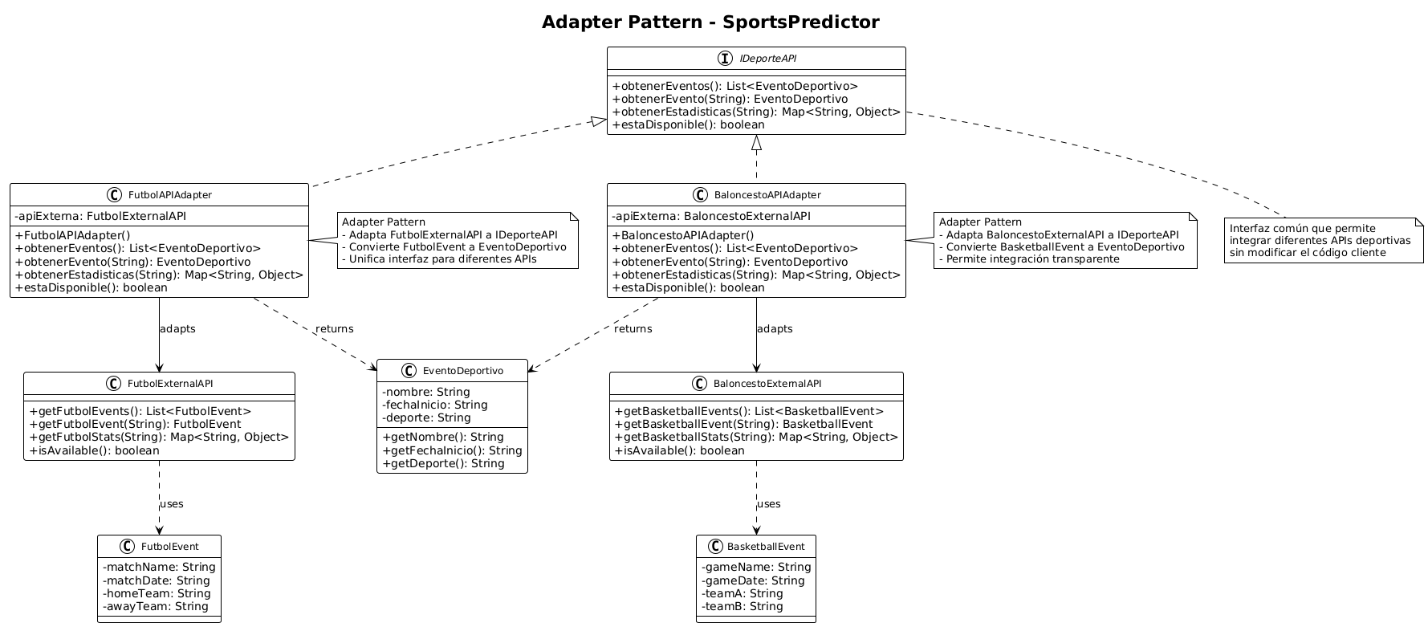
## Adapter (Estructural)

### Problema de diseño que ataca:

* **Integración de sistemas externos**: El sistema necesita integrar diferentes APIs de deportes, servicios de notificación, y sistemas de puntuación que tienen interfaces incompatibles.
* **Cambios en APIs externas**: Cuando cambien las APIs de proveedores de datos deportivos, no se debería afectar el código principal.

### Implementación en el código:

* **Adapter para APIs deportivas**: Crear adaptadores como FutbolAPIAdapter, BaloncestoAPIAdapter que implementen una interfaz común IDeporteAPI para normalizar los datos de diferentes deportes.
* **Adapter para notificaciones**: El EmailNotifier ya actúa como un adaptador básico, pero se podría extender para adaptar diferentes servicios de email (Gmail, Outlook, SendGrid).
* **Adapter para sistemas de puntuación**: Crear adaptadores para diferentes algoritmos de cálculo de puntos según el deporte o tipo de pronóstico.



### Beneficios:

* Permite integrar nuevos proveedores de datos sin cambiar el código principal
* Facilita el testing con implementaciones mock
* Mantiene la consistencia en la interfaz interna del sistema

## State (Comportamental)

### Problema de diseño que ataca:

* **Estados complejos de pronósticos**: Los pronósticos tienen múltiples estados (PENDIENTE, ACERTADO, FALLIDO, EN\_REVISION) con comportamientos diferentes en cada estado.
* **Transiciones de estado**: Las reglas de cuándo y cómo cambiar de un estado a otro están dispersas en el código, creando lógica compleja y difícil de mantener.

### Implementación en el código:

* **Estados de Pronóstico**:Crear clases como PronosticoPendienteState, PronosticoAcertadoState, PronosticoFallidoState, PronosticoEnRevisionState que implementen una interfaz IPronosticoState.
* **Contexto del Estado**: La clase Pronostico actuaría como contexto, delegando el comportamiento a su estado actual.
* **Transiciones**: Cada estado definiría las transiciones válidas y la lógica asociada (ej: solo se puede cambiar de PENDIENTE a ACERTADO cuando el evento finaliza).

Diagrama, Dibujo de ingeniería

El contenido generado por IA puede ser incorrecto.

### Beneficios:

* Elimina las cadenas de if-else para manejar estados
* Facilita agregar nuevos estados y comportamientos
* Centraliza la lógica de transiciones de estado
* Mejora la legibilidad y mantenibilidad del código

## Chain of Responsibility (Comportamental)

### Problema de diseño que ataca:

* **Escalamiento de incidencias**: El sistema actual tiene una lógica rígida donde MiembroSoporte decide si escalar a MiembroQC, pero esto no es flexible para agregar nuevos niveles de soporte o cambiar las reglas de escalamiento.
* **Procesamiento secuencial**: Las incidencias necesitan pasar por diferentes niveles de revisión con reglas específicas en cada nivel.

### Implementación en el código:

* **Cadena de procesamiento**: Crear una cadena donde MiembroSoporte → MiembroQC → Administrador procesen las incidencias en secuencia.
* **Handlers de incidencias**: Cada miembro implementaría IIncidenciaHandler con métodos puedeProcesar() y procesar().
* **Reglas de escalamiento**: Cada handler decidiría si puede resolver la incidencia o debe pasarla al siguiente en la cadena.

Diagrama

El contenido generado por IA puede ser incorrecto.

### Beneficios:

* Flexibilidad para agregar nuevos niveles de soporte
* Cada handler tiene una responsabilidad específica
* Fácil modificación de las reglas de escalamiento
* Desacoplamiento entre los diferentes niveles de soporte

# Sección B: Plan de Pruebas

## Clase Usuario

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Método a probar | Datos de entrada | Salida esperada | Propósito de la prueba |
| U1 | Constructor | id=1, nombre="Juan", email="juan@test.com", telefono="123456789" | Objeto Usuario creado correctamente | Verificar creación válida del objeto |
| U2 | realizarPronostico | evento válido, opcion="Local" | Pronóstico creado y puntos actualizados | Validar creación de pronóstico exitosa |
| U3 | canjearPuntos | puntos=100, puntosACanjear=50 | Puntos reducidos a 50 | Verificar canje de puntos exitoso |
| U4 | canjearPuntos | puntos=30, puntosACanjear=50 | Excepción por puntos insuficientes | Validar validación de puntos insuficientes |
| U5 | consultarHistorial | - | Lista de pronósticos del usuario | Verificar retorno del historial de pronósticos |

## Clase: Cuenta

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Método a probar | Datos de entrada | Salida esperada | Propósito de la prueba |
| C1 | Constructor | id=1, nombre="Admin", email="admin@test.com" | Objeto Cuenta creado correctamente | Verificar creación válida del objeto |
| C2 | login | email="admin@test.com", password="123" | true (login exitoso) | Validar autenticación correcta |
| C3 | logout | - | false (sesión cerrada) | Verificar cierre de sesión |
| C4 | setTelefono/getTelefono | telefono="987654321" | "987654321" | Validar setter y getter de teléfono |
| C5 | Herencia | - | Usuario es instancia de Cuenta | Verificar herencia correcta de Usuario |

## Clase: Pronostico

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Método a probar | Datos de entrada | Salida esperada | Propósito de la prueba |
| P1 | Constructor | id=1, usuario válido, evento válido | Objeto Pronóstico creado correctamente | Verificar creación válida del objeto |
| P2 | setEstado | estado="ACERTADO" | Estado actualizado correctamente | Validar cambio de estado |
| P3 | setTipo | tipo="RESULTADO" | Tipo actualizado correctamente | Verificar cambio de tipo |
| P4 | procesarEstado | - | Estado procesado sin errores | Validar procesamiento del estado |
| P5 | Relaciones | - | Usuario y evento asociados correctamente | Verificar asociaciones del pronóstico |

## Clase: EventoDeportivo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Método a probar | Datos de entrada | Salida esperada | Propósito de la prueba |
| E1 | Constructor | id=1, nombre="Barcelona vs Madrid", fecha=2024-01-15 | Objeto Evento creado correctamente | Verificar creación válida del objeto |
| E2 | Estadísticas iniciales | - | Estadísticas vacías al crear | Validar inicialización de estadísticas |
| E3 | Propiedades | nombre="Partido", fecha=2024-01-15 | Propiedades correctamente asignadas | Verificar asignación de propiedades |
| E4 | Formato nombre | nombre="Barcelona vs Madrid" | Nombre no está vacío | Validar que el nombre tenga contenido |
| E5 | Fecha no nula | fecha=2024-01-15 | Fecha no es null | Verificar que la fecha sea válida |

## 

## Clase: Incidencia

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Método a probar | Datos de entrada | Salida esperada | Propósito de la prueba |
| I1 | Constructor | id=1, descripcion="Error en pronóstico", pronostico válido | Objeto Incidencia creado correctamente | Verificar creación válida del objeto |
| I2 | setEstado | estado="RESUELTA" | Estado actualizado correctamente | Validar cambio de estado |
| I3 | Incidencia compleja | isCompleja=true | isCompleja retorna true | Verificar marcado de incidencia compleja |
| I4 | Propiedades | descripcion="Error", pronostico válido | Propiedades correctamente asignadas | Validar asignación de propiedades |
| I5 | Relación con Pronóstico | - | Pronóstico asociado correctamente | Verificar asociación con pronóstico |

## Clase: Administrador

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Método a probar | Datos de entrada | Salida esperada | Propósito de la prueba |
| A1 | Constructor | id=1, nombre="Admin", email="admin@test.com" | Objeto Administrador creado correctamente | Verificar creación válida del objeto |
| A2 | gestionarOpciones | - | Opciones gestionadas sin errores | Validar gestión de opciones del sistema |
| A3 | gestionarReglas | - | Reglas gestionadas sin errores | Verificar gestión de reglas del sistema |
| A4 | Herencia | - | Administrador es instancia de Cuenta | Validar herencia correcta de Cuenta |
| A5 | login | email="admin@test.com", password="123" | true (login exitoso) | Verificar autenticación del administrador |

**Clase: MiembroSoporte**

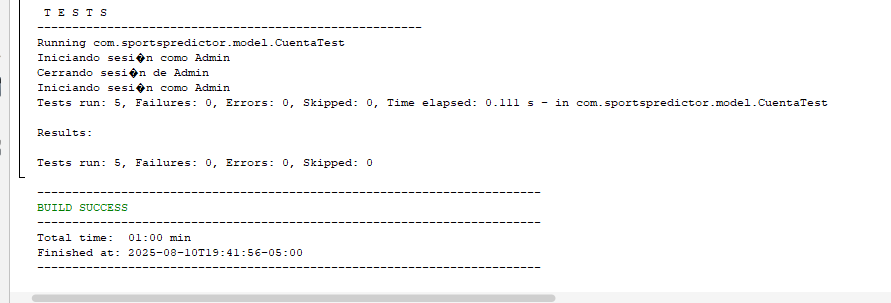
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Método a probar | Datos de entrada | Salida esperada | Propósito de la prueba |
| MS1 | Constructor | id=1, nombre="Soporte", email="soporte@test.com" | Objeto MiembroSoporte creado correctamente | Verificar creación válida del objeto |
| MS2 | gestionarIncidencia | incidencia simple | Incidencia gestionada sin errores | Validar gestión de incidencia simple |
| MS3 | gestionarIncidencia | incidencia compleja | Incidencia escalada correctamente | Verificar escalamiento de incidencia compleja |
| MS4 | Herencia | - | MiembroSoporte es instancia de Cuenta | Validar herencia correcta de Cuenta |
| MS5 | login | email="soporte@test.com", password="123" | true (login exitoso) | Verificar autenticación del miembro de soporte |

## Clase: MiembroQC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Método a probar | Datos de entrada | Salida esperada | Propósito de la prueba |
| MQ1 | Constructor | id=1, nombre="QC", email="qc@test.com" | Objeto MiembroQC creado correctamente | Verificar creación válida del objeto |
| MQ2 | revisarIncidenciaDetallada | incidencia válida | Revisión detallada completada | Validar revisión detallada de incidencia |
| MQ3 | Herencia | - | MiembroQC es instancia de MiembroSoporte | Verificar herencia correcta de MiembroSoporte |
| MQ4 | login | email="qc@test.com", password="123" | true (login exitoso) | Validar autenticación del miembro QC |
| MQ5 | Formato ID | id=1 | ID es un entero positivo | Verificar formato válido del ID |

# Sección C: Implementación y Pruebas Unitarias

## 1. CuentaTest (Clase: Cuenta)

****

## 2. AdministradorTest (Clase: Administrador)

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.**

## 3. EventoDeportivoTest (Clase: EventoDeportivo)

**Texto, Tabla

El contenido generado por IA puede ser incorrecto.**

## 4. IncidenciaTest (Clase: Incidencia)

**Texto, Tabla

El contenido generado por IA puede ser incorrecto.**

## 5. MiembroQCTest (Clase: MiembroQC)

**Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.**

## 6. MiembroSoporteTest (Clase: MiembroSoporte)

**Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.**

## 7. PronosticoTest (Clase: Pronostico)

**Texto, Tabla

El contenido generado por IA puede ser incorrecto.**

## 8. UsuarioTest (Clase: Usuario)

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.**

## Conclusión

La implementación de las pruebas unitarias para el sistema SportsPredictor ha demostrado ser exitosa, validando la funcionalidad del código. Con un total de 40 tests ejecutándose sin fallos. Si bien este conjunto de pruebas cubre los aspectos fundamentales del modelo, representa un primer paso importante en el proceso de aseguramiento de calidad, estableciendo una base confiable para futuras implementaciones en el código. Respetando a su vez la estructura del mismo y los patrones de diseño usados.