




10 DE AGOSTO DE 2025

TAREA03 - PATRONES Y PRUEBAS

HW03 - 6

- PEDRO BARAHONA
 - ABEL CARRERA
 - VALERIA GUTIERREZ
- 

Contenido

Sección A : Patrones y Diagrama de Clases	3
1. Factory Method (Creacional)	3
Problema de diseño que ataca:	3
Implementación en el código:	3
Beneficios:	3
2. Adapter (Estructural)	4
Problema de diseño que ataca:	4
Implementación en el código:	4
Beneficios:	4
3. State (Comportamental)	5
Problema de diseño que ataca:	5
Implementación en el código:	5
Beneficios:	6
4. Chain of Responsibility (Comportamental)	6
Problema de diseño que ataca:	6
Implementación en el código:	6
Beneficios:	7
Sección B: Plan de Pruebas	8
Clase Usuario	8
Clase: Cuenta	9
Clase: Pronostico	9
Clase: EventoDeportivo	10
Clase: Incidencia	11
Clase: Administrador	11
Clase: MiembroQC	13
Sección C: Implementación y Pruebas Unitarias	14
1. CuentaTest (Clase: Cuenta)	14
2. AdministradorTest (Clase: Administrador)	14

3. EventoDeportivoTest (Clase: EventoDeportivo)	14
4. IncidenciaTest (Clase: Incidencia)	15
5. MiembroQCTest (Clase: MiembroQC)	15
6. MiembroSoporteTest (Clase: MiembroSoporte).....	15
7. PronosticoTest (Clase: Pronostico)	16
8. UsuarioTest (Clase: Usuario)	16
Conclusión.....	16

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

- Facilita testing

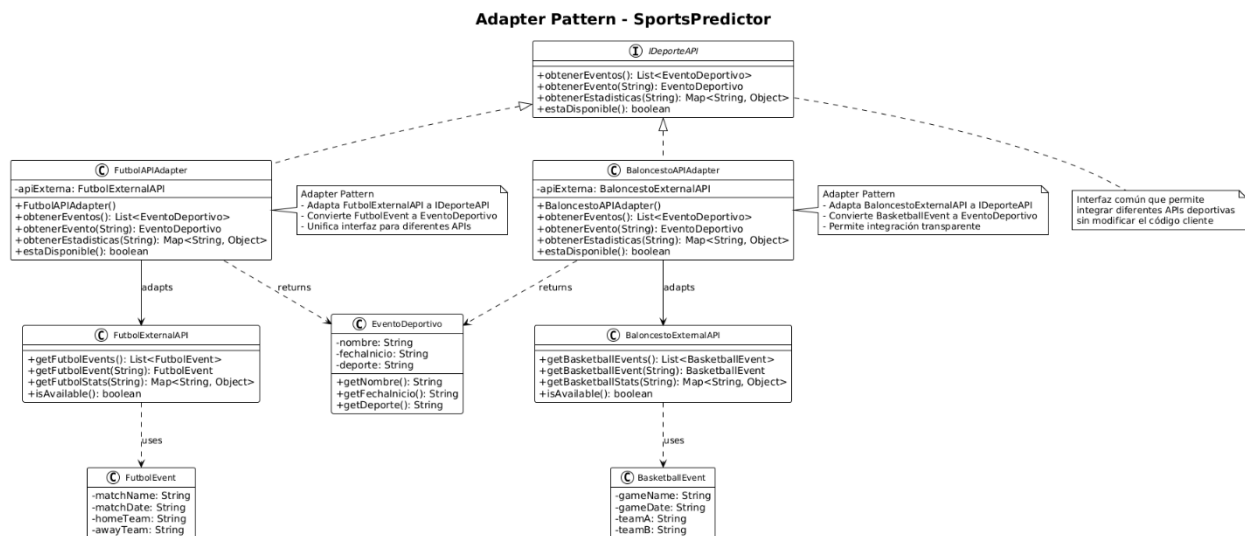
2. Adapter (Estructural)

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

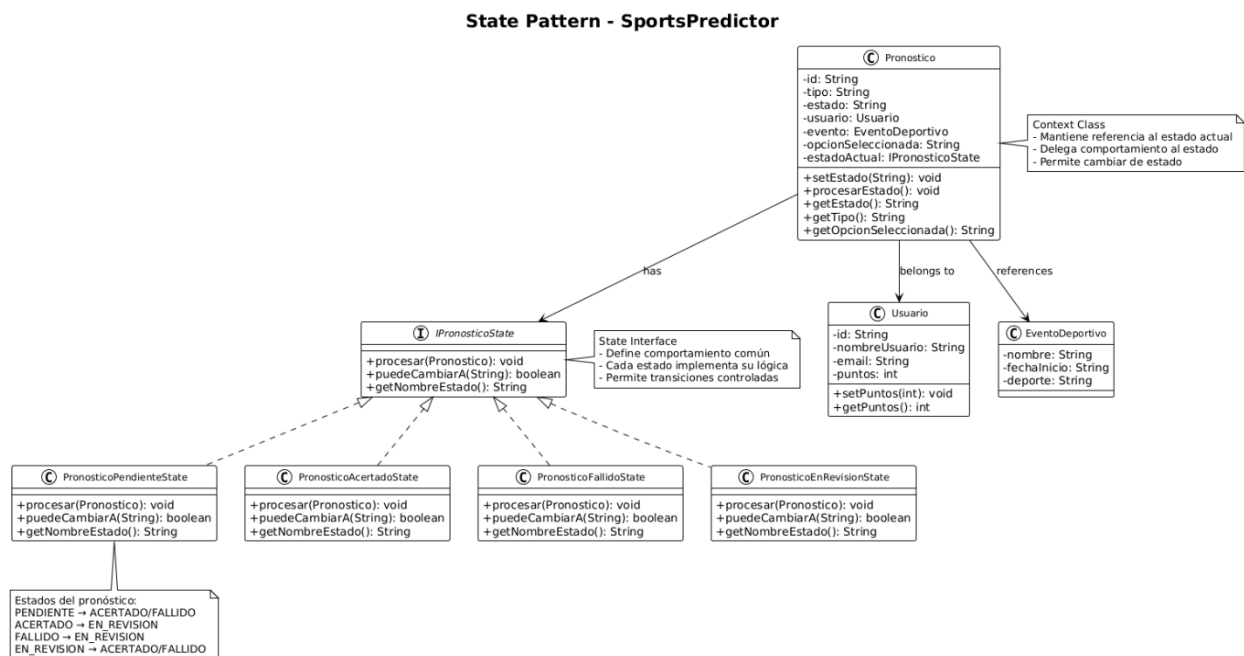
3. State (Comportamental)

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).



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

4. Chain of Responsibility (Comportamental)

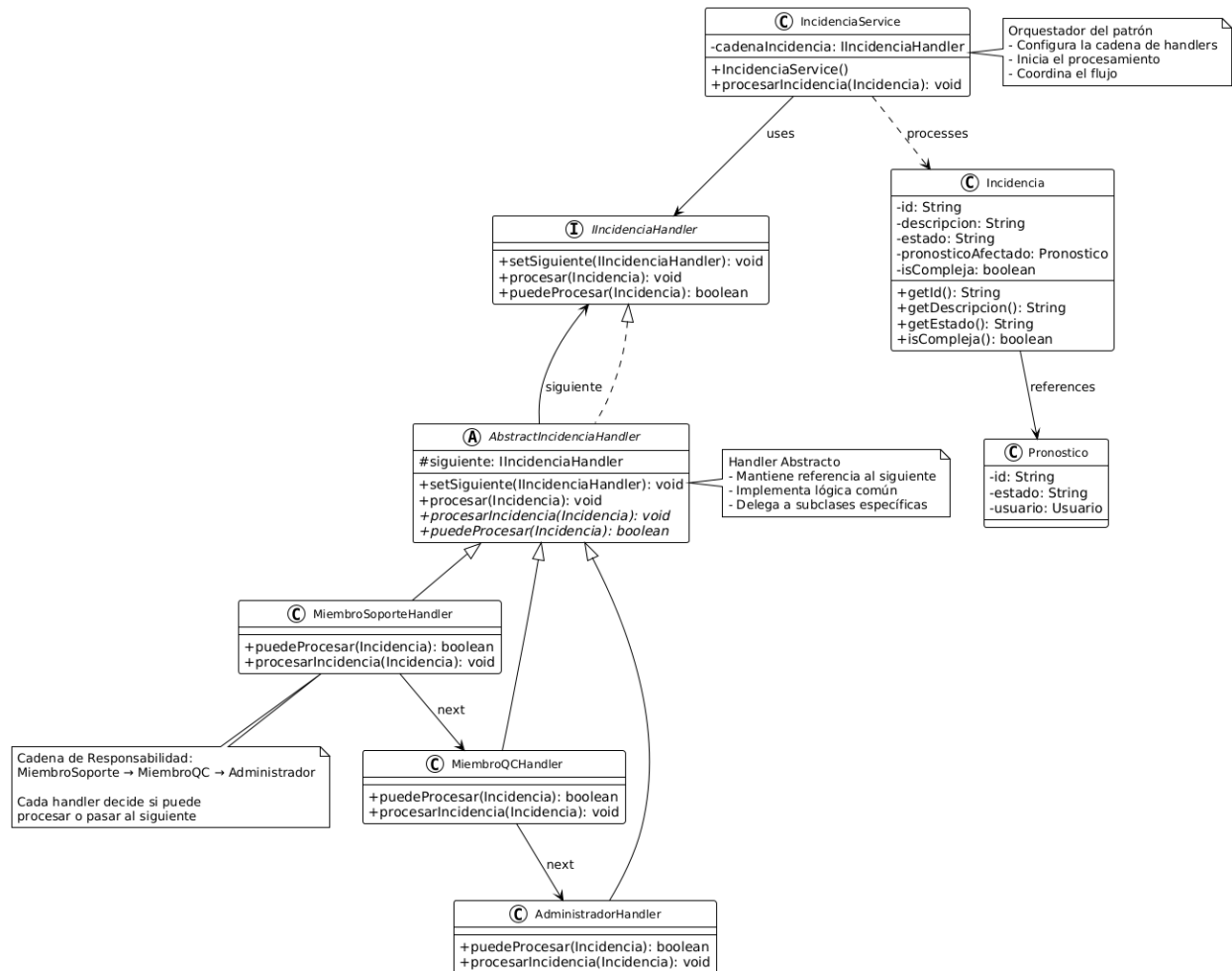
Problema de diseño que ataca:

- **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.

Chain of Responsibility Pattern - SportsPredictor



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

A 3	gestionarReglas	-	Reglas gestionadas sin errores	Verificar gestión de reglas del sistema
A 4	Herencia	-	Administrador es instancia de Cuenta	Validar herencia correcta de Cuenta
A 5	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
MS 1	Constructor	id=1, nombre="Soporte", email="soporte@test.com"	Objeto MiembroSoporte creado correctamente	Verificar creación válida del objeto
MS 2	gestionarInciden- cia	incidencia simple	Incidencia gestionada sin errores	Validar gestión de incidencia simple
MS 3	gestionarInciden- cia	incidencia compleja	Incidencia escalada correctamente	Verificar escalamiento de incidencia compleja
MS 4	Herencia	-	MiembroSoporte es instancia de Cuenta	Validar herencia correcta de Cuenta
MS 5	login	email="soporte@test.com", password="123"	true (login exitoso)	Verificar autenticación

				n del miembro de soporte
--	--	--	--	--------------------------------

Clase: MiembroQC

ID	Método a probar	Datos de entrada	Salida esperada	Propósito de la prueba
MQ 1	Constructor	id=1, nombre="QC", email="qc@test.com"	Objeto MiembroQC creado correctamente	Verificar creación válida del objeto
MQ 2	revisarIncidenciaDetallada	incidencia válida	Revisión detallada completada	Validar revisión detallada de incidencia
MQ 3	Herencia	-	MiembroQC es instancia de MiembroSopORTE	Verificar herencia correcta de MiembroSopORTE
MQ 4	login	email="qc@test.com", password="123"	true (login exitoso)	Validar autenticación del miembro QC
MQ 5	Formato ID	id=1	ID es un entero positivo	Verificar formato válido del ID

Sección C: Implementación y Pruebas Unitarias

1. CuentaTest (Clase: Cuenta)

```
-----
T E S T S
-----
Running com.sportspredictor.model.CuentaTest
Iniciando sesión como Admin
Cerrando sesión de Admin
Iniciando sesión como Admin
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.111 s - in com.sportspredictor.model.CuentaTest

Results:

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0

-----
BUILD SUCCESS
-----
Total time: 01:00 min
Finished at: 2025-08-10T19:41:56-05:00
-----
```

2. AdministradorTest (Clase: Administrador)

```
-----
T E S T S
-----
Running com.sportspredictor.model.AdministradorTest
El administrador Admin está gestionando las reglas de puntuación.
El administrador Admin está gestionando opciones para el evento: Barcelona vs Madrid
Iniciando sesión como Admin
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.079 s - in com.sportspredictor.model.AdministradorTest

Results:

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0

-----
BUILD SUCCESS
-----
Total time: 5.408 s
Finished at: 2025-08-10T19:54:16-05:00
-----
```

3. EventoDeportivoTest (Clase: EventoDeportivo)

```
--- surefire:3.0.0:test (default-cli) @ sportspredictor-system ---
Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider

-----
T E S T S
-----
Running com.sportspredictor.model.EventoDeportivoTest
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.07 s - in com.sportspredictor.model.EventoDeportivoTest

Results:

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0

-----
BUILD SUCCESS
-----
Total time: 2.207 s
Finished at: 2025-08-10T19:55:45-05:00
-----
```

4. IncidenciaTest (Clase: Incidencia)

```
--- Surefire3.0.0-test (default-cli) @sportspredictor-system ---
Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider

-----
T E S T S
-----
Running com.sportspredictor.model.IncidenciaTest
El estado de la incidencia I001 ha cambiado a: RESUELTA
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.083 s - in com.sportspredictor.model.IncidenciaTest

Results:

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0

-----
BUILD SUCCESS
-----
Total time: 2.844 s
Finished at: 2025-08-10T19:56:09-05:00
-----
```

5. MiembroQCTest (Clase: MiembroQC)

```
-----
T E S T S
-----
Running com.sportspredictor.model.MiembroQCTest
Iniciando sesi3n como QC
El miembro de QC QC est3 realizando una revisi3n detallada de la incidencia: I001
El estado de la incidencia I001 ha cambiado a: RESUELTA
La incidencia I001 ha sido resuelta por el equipo de Calidad.
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.078 s - in com.sportspredictor.model.MiembroQCTest

Results:

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0

-----
BUILD SUCCESS
-----
Total time: 2.299 s
Finished at: 2025-08-10T19:56:27-05:00
-----
```

6. MiembroSoporteTest (Clase: MiembroSoporte)

```
-----
T E S T S
-----
Running com.sportspredictor.model.MiembroSoporteTest
El miembro de soporte Soporte est3 gestionando la incidencia: I001
El estado de la incidencia I001 ha cambiado a: RESUELTA
La incidencia I001 ha sido resuelta.
Iniciando sesi3n como Soporte
El miembro de soporte Soporte est3 gestionando la incidencia: I002
La incidencia I002 es compleja y ser3 escalada.
El miembro de QC QualityControlUser est3 realizando una revisi3n detallada de la incidencia: I002
El estado de la incidencia I002 ha cambiado a: RESUELTA
La incidencia I002 ha sido resuelta por el equipo de Calidad.
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.082 s - in com.sportspredictor.model.MiembroSoporteTest

Results:

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0
```


7. PronosticoTest (Clase: Pronostico)

```
put - test (PronosticoTest) x
Using auto detected provider: org.apache.maven.surefire.junitplatform.JUnit4PlatformProvider

-----
T E S T S
-----
Running com.sportspredictor.model.PronosticoTest
El estado del pronóstico P001 ha cambiado a: ACERTADO
Pronóstico P001 está pendiente de resolución
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.101 s - in com.sportspredictor.model.PronosticoTest

Results:

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0

-----
BUILD SUCCESS
-----
Total time: 2.441 s
Finished at: 2025-08-10T19:57:16-05:00
-----
```

8. UsuarioTest (Clase: Usuario)

```
put - test (UsuarioTest) x
-----
T E S T S
-----
Running com.sportspredictor.model.UsuarioTest
Juan está realizando un pronóstico para el evento Barcelona vs Madrid
Puntos insuficientes para el canje.
Canje exitoso. Puntos restantes: 50
Juan está realizando un pronóstico para el evento Barcelona vs Madrid
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.086 s - in com.sportspredictor.model.UsuarioTest

Results:

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0

-----
BUILD SUCCESS
-----
Total time: 2.555 s
Finished at: 2025-08-10T19:57:43-05:00
-----
```

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.