Capítulo 6
Repositorio de datos

Objetivo

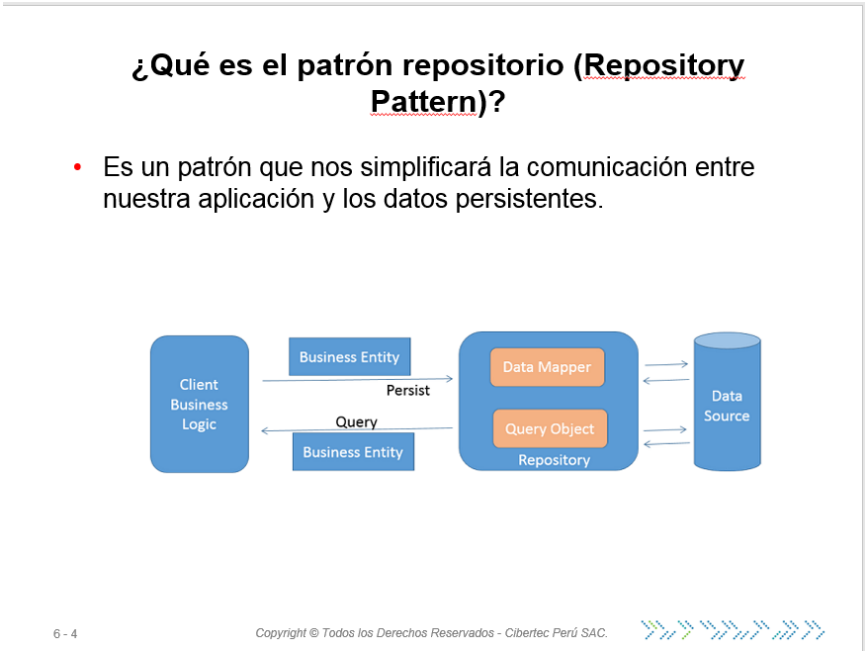
Al finalizar el capítulo, el alumno:

- Comprender la utilidad del patrón repositorio y unit of work.
- Implementar el patrón repositorio y unit of work.

Temas

1. ¿Qué es el patrón repositorio?
2. ¿Qué es el patrón unit of work?
3. Implementando el patrón repositorio:
   Entity Framework

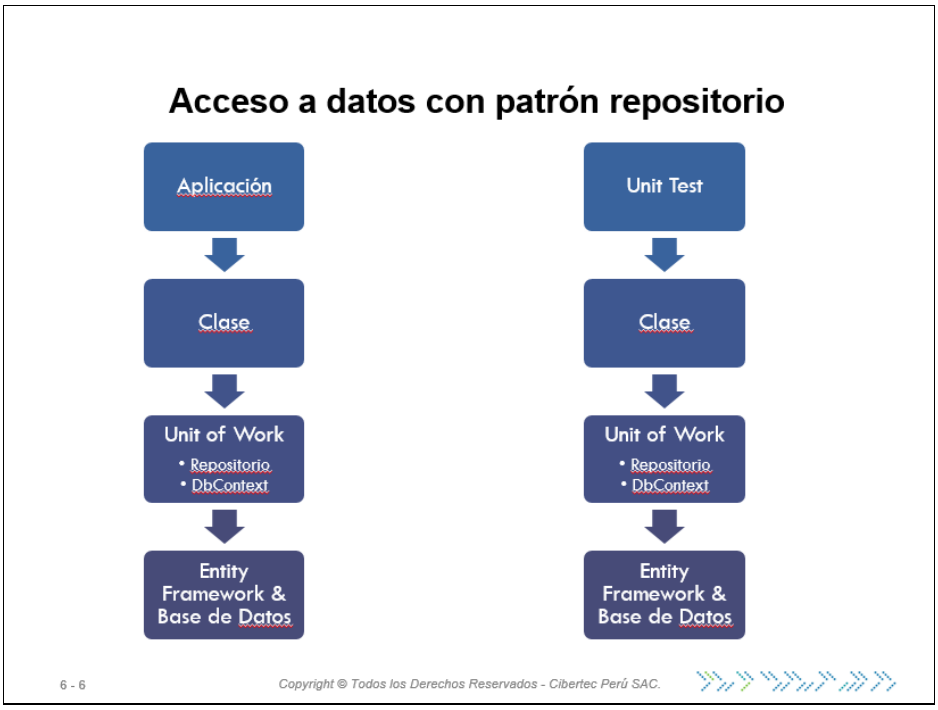
1. ¿Qué es el patrón repositorio (Repository Pattern)?



Es habitual en desarrollo encontrarnos una y otra vez con los mismos problemas de un proyecto a otro e incluso en el mismo proyecto. Tendemos a solucionarlos siempre de una misma manera, conocida y cuyos resultados y efectos conocemos también. A finales de los años 80, Ward Cunningham y Kent Beck decidieron aplicar la idea de los patrones de diseño que Christopher Alexander aportó al mundo de la arquitectura.

Pensaron que podrían abstraerse del problema concreto y expresar la solución proporcionada en términos de patrones genéricos que pudieran ser reusados una y otra vez. Quizás el sector no estaba preparado para aplicar patrones de diseño en su código. Habría que esperar a 1994 para que Gamma, Helm, Johnson y Vlissides, conocidos como The Gans of Four [GoF] escribieran Design Patterns, Elements of reusable Object-Oriented Software. En él describen 23 patrones comunes y su influencia en la programación es equivalente a la del método científico moderno de Galileo para la ciencia actual. Sin embargo, no es hasta año 2003 que no oímos hablar por primera vez del patrón Repositorio de mano de Martin Fowler en su libro Patterns of Enterprise Application Arrchitecture [Fowler, PEAA], y cuya autoría adjudica a Edward Hieatt y Rob Mee.

2. Implementando el patrón repositorio



Es un patrón para aislar los tecnicismos de la persistencia de las capas superiores y en especial del modelo de Dominio. Gracias a ello, podremos concentrarnos en implementar en el modelo las reglas de negocio. El cliente ahora podrá concentrarse en cómo presentar la información, en cómo aplicar filtros, algoritmos, relaciones y reglas que hagan cumplir las reglas de empresa especificadas. Otra ventaja de usar este patrón es que podemos usar distintos tipos de persistencia, ya sea a la vez o por sustitución. Por ejemplo, podemos usar SQL Server para almacenar los usuarios, Oracle para los productos, y Excel para las órdenes de Compra, o mañana sustituir la base de datos de Oracle por Azure SQL. Derivado de esto podemos implementar un Repositorio en memoria, que nos sirva para hacer pruebas de test unitarias (**Test Driven Development, TDD**) sobre nuestro modelo de negocio sin que afecte a los datos reales, y así probar una y otra vez las capas superiores sin que por ello afecte a las capas inferiores.