Proyecto integrador » ARGBroker Demo

GRUPO THE CODE TEAM: INTEGRANTES

Equipo Base de datos

Nicolas Caceres DNI: 34840849

Email: ncaceres.cba@gmail.com GitHub: NicolasCaceres

Carolina Soledad Sanchez DNI: 40026893

Email: carolinasanchz20@gmail.com GitHub: CarolinaSanchez

Santiago Leonel Terragni DNI: 40403581

Email: terragnisantiago97@gmail.com GitHub: SantiagoTerragni

Equipo Ética

Tomás Quinteros DNI: 44077901

Email: tomasquinteros50@gmail.com GitHub: TomasQuinteros

Stella Maris Tita Mascarello DNI: 39623753

Email: stella.tita@mi.unc.edu.ar GitHub: StellaTita

Equipo Programación

Agustina López DNI: 43273918

Email: al6599522@gmail.com GitHub: AgustinaLopez

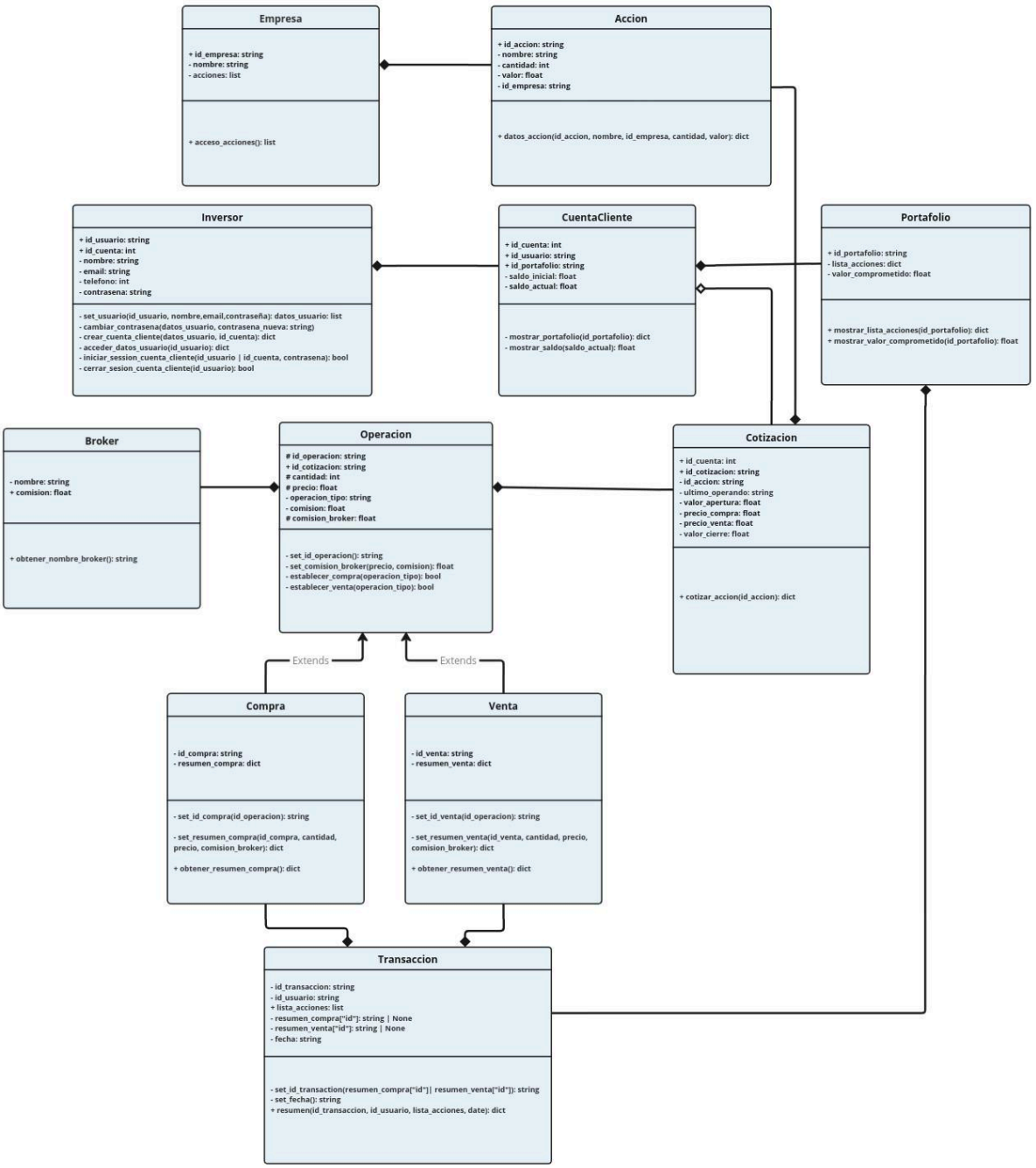
Leandro Matias Saig DNI: 32785685

Email: saig.matias@gmail.com GitHub: Matias-Saig

Programación 1: Diagrama de clases

Vista previa del diagrama de clases

Link al archivo original: **Diagrama_clases.jpg**



Documentación del desarrollo de diagrama de clases para ARGBroker Demo

Introducción

En este documento se describe el desarrollo del diagrama de clases para el área de programación

Tareas

- Identificación de clases
- Creación de clases
- Establecimiento de relaciones
- Realizar diagrama de clases
- Documentar diagrama

Las tareas se dividieron por igual entre el equipo

Nomenclatura

Las clases se nombran utilizando la notación PascalCase

Los atributos y métodos con la notación de snake_case

No se usan tildes, ñ o caracteres especiales en la definición de nombres

Clases, Atributos, Métodos y Relación

Clase Inversor

Atributos público

+ *id_usuario: string* » identificador del usuario

+ *id_cuenta: int* » identificador de la cuenta de cliente del usuario

Atributos privados

- nombre: string | - email: string | - telefono: int | - contrasena: string

Métodos privados

- *set_usuario(id_usuario, nombre,email,contraseña): datos_usuario: list* » Definición (setter) de los valores del usuario, se asignan por parámetros del método y la función retorna una lista con los valores nuevos

- *cambiar_contrasena(datos_usuario, contrasena_nueva: string)* » Se ingresan por parámetro los datos de usuario más la contraseña nueva

- *crear_cuenta_cliente(datos_usuario, id_cuenta): dict* » Definición de cuenta de cliente pasando por parámetros los datos de usuario y un id de la nueva cuenta, retorna un diccionario con los datos nuevos

- *acceder_datos_usuario(id_usuario): dict* » (getter) retorna los datos del usuario en un diccionarios

- *iniciar_session_cuenta_cliente(id_usuario | id_cuenta, contrasena): bool* » retorna true o false para el acceso de la sesión

- *cerrar_sesion_cuenta_cliente(id_usuario): bool* » retorna true o false para cerrar la sesión

Relación

Recepta como componente a la clase CuentaCliente a través del atributo público id_cuenta

Clase CuentaCliente

Atributos públicos

+ *id_cuenta: int* » identificador de la cuenta cliente

+ *id_usuario: string* » identificador del usuario (proveniente de clase Inversor)

+ *id_portafolio: string* » Identificador del portafolio (proveniente de clase Portafolio)

Atributos privados

- *saldo_inicial: float* » Dinero disponible a la apertura de la cuenta

- *saldo_actual: float* » Dinero disponible de la cuenta, sobreescribe saldo inicial

Métodos privados

- *mostrar_portafolio(id_portafolio): dict* » Acceso al portafolio de la cuenta cliente, retorna diccionario con los datos del portafolio

- *mostrar_saldo(saldo_actual): float* » Muestra el atributo privado saldo actual, retorna número con decimales

Relación

Es componente de la clase Inversor a través del atributo público id_usuario

Recepta como componente la clase Portafolio a través del atributo público id_portafolio

Recepta como agregado la clase Cotizacion a través del atributo público id_cuenta

Clase Portafolio

Atributos públicos

+ *id_portafolio: string* » Identificador de portafolio

Atributos privados

- *lista_acciones: dict* » Diccionario con las acciones pertenecientes al usuario, su cantidad y valor estimativo

- *valor_comprometido: float* » Total del valor estimativo de las acciones, número con decimales

Métodos públicos

+ *mostrar_lista_acciones(id_portafolio): dict* » Muestra atributo privado lista_acciones, retorna un diccionario con las acciones, su cantidad y precio

+ *mostrar_valor_comprometido(id_portafolio)*: float » Muestra atributo privado valor_comprometido, retorna un número con decimales

Relación

Es componente parte de clase CuentaCliente a través del atributo público id_portafolio

Recepta como componente la clase Transaccion a través del atributo público lista_acciones (de Transacciones)

Clase Empresa

Atributos públicos

+ *id_empresa: string* » Identificador de la empresa a la que pertenece la acción

Atributos privados

- *nombre: string* » Nombre de la empresa

- *acciones: list* » Lista de acciones que oferta la empresa

Métodos públicos

+ *acceso_acciones(): list* » acceso al atributo privado acciones, retorna lista

Relación

Recepta clase acción a través de atributo público id_acción

Clase Accion

Atributos públicos

+ *id_accion: string* » Identificador de la acción

Atributos privados

- *nombre: string* » Nombre de la acción
- *cantidad: int* » Cantidad disponible de la acción
- *valor: float* » Precio de la acción, número con decimales
- *id_empresa: string* » Identificador de la empresa (proveniente de clase Empresa)

Métodos públicos

+ *datos_accion(id_accion, nombre, id_empresa, cantidad, valor): dict* » retorna un diccionario con los datos de la acción

Relación

Es componente de la clase Empresa a través del atributo público id_acción

Es componente de la clase Cotizacion a través del atributo público id_acción

Clase Cotizacion

Atributos públicos

+ *id_cuenta: int* » Identificador de cuenta (proveniente de clase CuentaCliente)
+ *id_cotizacion: string* » Identificador de la cotización

Atributos privados

- *id_accion: string* » Identificador de la acción (proveniente de la clase Acción)
- *ultimo_operando: string »* Precio de la última transacción de la acción
- *valor_apertura: float* » Precio de apertura de la acción
- *precio_compra: float* » Precio de compra de la acción
- *precio_venta: float* » Precio de venta de la acción
- *valor_cierre: float »* Precio de la última transacción de la acción al momento del cierre de mercado

Métodos públicos

+ *cotizar_accion(id_accion): dict* » Retorna en un diccionario los datos de la cotización

Relación

Es componente de la clase Operación a través del atributo público id_cotizacion

Clase Operacion

Atributos públicos

+ *id_cotizacion: string* » Identificador de la cotización (proveniente de clase Cotización)

Atributos privados

- *operacion_tipo: string* » Tipo de operación compra/venta

- *comision: float* » Porcentaje de la comisión del broker (proveniente de la clase Broker)

Atributos protegido

id_operacion: string » Identificador de la operación

cantidad: int » Cantidad de acciones a comprar

precio: float » Precio total

comision_broker: float » Costo en dinero de la comisión del brroker

Métodos privados

- *set_id_operacion(): string* » Establece identificador de la operación, retorna cadena de carácteres

- *set_comision_broker(precio, comision): float* » Cálculo de la comisión del broker sobre el precio total, retorna un número con decimales

- *establecer_compra(operacion_tipo): bool* » determina sí operacion_tipo es compra, retorna true o false

- *establecer_venta(operacion_tipo): bool* » determina sí operacion_tipo es compra, retorna true o false

Relación

Recepta como componente la clase Cotizacion a través de atributo público id_cotizacion

Recepta como componente la clase Broker a través del atributo público comision

Es superclase de clases Compra y Venta

Clase Broker

Atributos públicos

+ *comision: float* » Porcentaje de la comisión del Broker

Atributos privados

- *nombre: string* » nombre del broker

Métodos públicos

+ *obtener_nombre_broker(): string* » Obtiene el atributo privado nombre

Relación

Es componente la clase Operación a través del atributo público comision

Clase Compra

Atributos privados

- *id_compra: string* » Identificador de la compra
- *resumen_compra: dict* » atributos heredados de superclase Operacion

Métodos privados

- *set_id_compra(id_operacion): string* » Define identificador de la compra usando como parámetro el atributo protegido id_operacion de la superclase Operacion
- *set_resumen_compra(id_compra, cantidad, precio, comision_broker): dict* » Toma como parámetro los atributos heredados de superclase Operacion más el atributo privado id_compra, retorna un diccionario con ellos

Métodos públicos

+ *obtener_resumen_compra(): dict* » Retorna todos los datos de la compra en un diccionario

Relación

Subclase de clase Operacion

Es componente de la clase Transaccion a través del atributo privado resumen_compra["id"]

Clase Venta

Atributos privados

- *id_venta: string* » Identificador de la venta
- *resumen_venta: dict* » atributos heredados de superclase Operacion

Métodos privados

- *set_id_venta(id_operacion): string* » Define identificador de la venta usando como parámetro el atributo protegido id_operacion de la superclase Operacion

- *set_resumen_venta(id_venta, cantidad, precio, comision_broker): dict* » Toma como parámetro los atributos heredados de superclase Operacion más el atributo privado id_venta, retorna un diccionario con ellos

Métodos públicos

+ *obtener_resumen_venta(): dict* » Retorna todos los datos de la venta en un diccionario

Relación

Subclase de clase Operacion

Es componente de la clase Transaccion a través del atributo privado resumen_venta["id"]

Clase Transaccion

Atributos públicos

+ *lista_acciones: list* » Lista de acciones compradas o vendidas en la transacción

Atributos privados

- *id_transaccion: string* » Identificador de la transacción

- *id_usuario: string* » Identificador del usuario

- *resumen_compra["id"]: string* | *None* » Obtiene identificador de la compra a partir de la clave “id” del diccionario resumen_compra, sí el diccionario existe retorna un cadena de caracteres, sí no existe retorna None

- *resumen_venta["id"]: string* | *None* » Obtiene identificador de la venta a partir de la clave “id” del diccionario resumen_venta, sí el diccionario existe retorna una cadena de caracteres, sí no existe retorna None

- *fecha: string* » fecha de la transacción

Métodos privados

- *set_id_transaction(resumen_compra["id"]* | *resumen_venta["id"]): string* » Establece el identificador de la transacción tomando los valores de la clave “id” del diccionario correspondiente a la transacción realizada, retorna una cadena de caracteres

- *set_fecha(): string* » Establece la fecha

Métodos públicos

+ *resumen(id_transaccion, id_usuario, lista_acciones, date): dict* » Crea un resumen de la transacción realizada

Relación

Recepta como componentes la clase Compra a través del atributo resumen_compra[“id”] obtenido a partir del diccionario resumen_compra con método obtener_resumen_compra()

Recepta como componentes la clase Venta a través del atributo resumen_venta[“id”] obtenido a partir del diccionario resumen_venta con método obtener_resumen_venta()

Es un componente de la clase Portafolio a través del atributo público lista_acciones

Base de datos: modelo relacional

Vista previa del modelo relacional

Link al archivo original: **diagrama_entidad_relacion.jpg**

Documentación del Desarrollo de la Base de Datos

Introducción

Este documento presenta la documentación del desarrollo de un programa de Base de Datos diseñado para el proyecto ARGBroker Demo. El contenido describe detalladamente las entidades, sus atributos, relaciones y su funcionalidad dentro del programa.

Descripción de las Entidades

Usuario/Inversor

La entidad "Usuario/Inversor" almacena información relevante sobre los inversores, incluyendo su nombre, correo electrónico y contraseña de acceso al programa. La clave primaria de esta entidad es ID_usuario.

Relaciones:

- Relacionada con Portafolio (uno a muchos): Cada usuario puede tener varios portafolios, pero cada portafolio pertenece a un solo usuario.

- Relacionada con Cuenta (uno a muchos): Cada usuario puede tener varias cuentas, pero cada cuenta pertenece a un solo usuario.

Empresa

En esta entidad se almacena información descriptiva sobre las empresas, como su símbolo identificatorio y nombre. La clave primaria es ID_empresa.

Relación uno a muchos:

- Una empresa puede tener muchas acciones disponibles, pero cada acción corresponde a una empresa.

Acciones

Esta entidad almacena datos relacionados con la cantidad y valor de las acciones, junto con su fecha correspondiente. La clave primaria es ID_accion.

Relaciones:

- Relacionada con Cotizaciones (uno a muchos): Una acción puede tener múltiples cotizaciones, pero una cotización corresponde a una única acción.

- Relacionada con Empresa (muchos a uno): Una empresa puede tener múltiples acciones asociadas, pero cada acción pertenece a una sola empresa.

Cotizaciones

La entidad "Cotizaciones" contiene información sobre el valor de apertura y cierre de las acciones, así como los precios de compra y venta con sus respectivas fechas. La clave primaria es ID_cotizacion.

Relaciones:

- Relacionada con Acciones (muchos a uno): Cada acción puede tener varias cotizaciones, pero una cotización corresponde a una acción.

- Relacionada con Compras (uno a muchos): Una cotización puede tener varias compras, pero una compra corresponde a una cotización.

- Relacionada con Ventas (uno a muchos): Una cotización puede tener varias ventas, pero una venta corresponde a una cotización.

Portafolio

La entidad "Portafolio" detalla la cantidad de acciones que posee un usuario junto con su valor comprometido. La clave primaria es ID_portafolio.

Relaciones:

- Relacionada con Usuario (muchos a uno): Un usuario puede tener varios portafolios, pero cada portafolio pertenece a un solo usuario.

- Relacionada con Acciones (muchos a muchos): Un portafolio puede contener varias acciones y una acción puede estar en varios portafolios.

- Relacionada con Stock_acciones (uno a uno): Un portafolio tiene un stock de acciones y un stock de acciones corresponde a un solo portafolio.

Stock_acciones

Esta entidad representa la cantidad de acciones que un usuario tiene dentro de su portafolio. La clave primaria es ID_accion_portafolio.

Relaciones:

- Relacionada con Portafolio (uno a uno): Un portafolio tiene un stock de acciones y un stock de acciones corresponde a un solo portafolio.

- Relacionada con Acciones (uno a muchos): Un stock de acciones contiene muchas acciones, pero una acción pertenece a un stock.

Cuenta

La entidad "Cuenta" proporciona información sobre los saldos iniciales y actuales que un usuario posee en su cuenta. La clave primaria es ID_cuenta.

Relaciones:

- Relacionada con Usuario (muchos a uno): Un usuario puede tener varias cuentas, pero cada cuenta pertenece a un solo usuario.

Transacción

Esta entidad registra las operaciones de compra y venta realizadas por un usuario, junto con sus respectivas fechas. La clave primaria es ID_transaccion.

Relaciones:

- Relacionada con Usuario (muchos a uno): Un usuario puede realizar varias transacciones, pero cada transacción es realizada por un solo usuario.

- Relacionada con Acción (muchos a uno): Una acción puede estar involucrada en varias transacciones, pero cada transacción involucra una sola acción.

- Relacionada con Compra (uno a uno): Una compra está asociada a una transacción y una transacción a una compra.

- Relacionada con Venta (uno a uno): Una venta está asociada a una transacción y una transacción a una venta.

Compra

La entidad "Compra" registra los detalles de las compras efectuadas, incluyendo la cantidad, precio y comisión correspondiente para el Broker. La clave primaria es ID_compras.

Relación uno a muchos:

- Relacionada con Cotizaciones: Una misma cotización puede tener varias compras, pero una compra solo corresponde a una cotización.

Venta

La entidad "Venta" registra los detalles de las ventas efectuadas, incluyendo la cantidad, precio y comisión correspondiente para el Broker. La clave primaria es ID_ventas.

Relación uno a muchos:

- Relacionada con Cotizaciones: Una misma cotización puede tener varias ventas, pero una venta solo corresponde a una cotización.

Conclusiones

La documentación anterior detalla las entidades, atributos y relaciones de la base de datos desarrollada para el proyecto ARGBroker Demo. Esta información es fundamental para comprender la estructura y funcionalidad del sistema, facilitando su desarrollo, mantenimiento y uso por parte de los usuarios.

Ética: conformación legal del grupo

Estructura de la Empresa

- **Socios**: Fundadores del proyecto que aportan capital y toman decisiones estratégicas.
- **CEO**: Responsable de la gestión y dirección de la empresa.
- **Empleados**: Personal contratado para desarrollar y operar la aplicación.

Contrato de Socios

Cláusulas:

- *Objeto del Contrato*: Definir el propósito y las actividades de la empresa.
- *Aportaciones de Capital*: Especificar la contribución de cada socio.
- *Distribución de Ganancias y Pérdidas*: Cómo se repartirán las ganancias y se asumirán las pérdidas.
- *Responsabilidades y Funciones:* Asignación de roles y responsabilidades.
- *Resolución de Conflictos*: Mecanismos para resolver disputas entre socios.
- *Confidencialidad:* Proteger la información confidencial de la empresa.
- *Duración y Terminación:* Condiciones para la disolución de la sociedad.

Matriculación Profesional

Requisitos de Matriculación

- **Profesionales**: Ingenieros en Sistemas, Programadores.
- **Entidad Matriculadora**: Colegio de Profesionales en Ciencias Informáticas de la Provincia de Córdoba (CPCIPC).

Pasos para Matriculación

1. **Solicitud de Inscripción**: Completar y presentar el formulario de inscripción.
2. **Documentación**: Presentar título profesional, DNI y antecedentes.
3. **Pago de Tasa**: Abonar la tasa de inscripción y la cuota anual.

Requerimientos Legales del Código y Base de Datos

Legislación Vigente

Ley de Protección de Datos Personales (Ley 25.326):

- *Consentimiento*: Obtener consentimiento explícito de los usuarios para el tratamiento de sus datos.
- *Seguridad*: Implementar medidas de seguridad adecuadas para proteger los datos personales.
- *Acceso y Rectificación*: Permitir a los usuarios acceder y corregir sus datos.

Ley de Sociedades Comerciales (Ley 19.550):

- *Constitución Legal*: Formalizar la constitución de la sociedad y su inscripción en el Registro Público de Comercio.
- *Obligaciones Fiscales*: Cumplir con las obligaciones impositivas y presentar balances contables.

Registro Nacional de Software

Pasos para el Registro

1. **Inscripción en el Registro Nacional de Propiedad del Software**: Presentar el software en la Dirección Nacional de Derecho de Autor.
2. **Documentación**: Proveer la documentación técnica y descriptiva del software.
3. **Pago de Tasa**: Abonar la tasa correspondiente.
4. **Certificación**: Obtener el certificado de registro de software.

Consecuencias de No Matriculación

Pena por No Matriculación

- **Sanciones**: Multas de dinero entre diez y quinientas veces el valor del derecho de Inscripción a la matrícula y prohibición de ejercer la profesión legalmente.
- **Responsabilidad Penal**: En caso de ejercicio ilegal, puede haber consecuencias penales.

Protección de Propiedad Intelectual

Ley de Propiedad Intelectual (Ley 11.723)

- Registro de Software: Protege el software registrado contra copias no autorizadas.
- Acción Legal: Permite acciones legales contra infractores, como demandas por daños y perjuicios.

Divulgación No Autorizada de Datos

Acción Legal en Caso de Divulgación

- **Denuncia**: Presentar una denuncia ante la Agencia de Acceso a la Información Pública (AAIP).
- **Acción Civil**: Demanda por violación de confidencialidad y daños.

Reutilización No Autorizada del Código

Legislación Provincial y Código Penal Argentino

- **Ley 25.326**: Protección de datos personales.
- **Código Penal**: Sanciona la divulgación no autorizada de secretos de empresa (Art. 153-155).

Protección Internacional del Código

Tratados Internacionales

- **Convenio de Berna**: Protección de obras literarias y artísticas, incluyendo software.
- **OMPI**: Organización Mundial de la Propiedad Intelectual ofrece protección global de derechos de autor.

Adulteración Externa de la Base de Datos

Protección Nacional

- **Ley 26.388**: Delitos informáticos, sanciona la adulteración de bases de datos.
- **Acción Penal**: Posibilidad de presentar denuncia penal por ataques cibernéticos.

Adulteración Internacional de la Base de Datos

Protección Internacional

- **Convenio de Budapest**: Convenio sobre Cibercrimen, protege contra delitos informáticos internacionales.

Negligencia de los Programadores

- Instrumentos legales para abordar y remediar la situación.
- Reclamar daños y perjuicios por incumplimiento del contrato de servicios profesionales.

Responsabilidad Legal

- **Ley de Defensa del Consumidor (Ley 24.240)**: Protección al usuario final en caso de negligencia.
- **Acción Legal**: Posible demanda por daños y perjuicios.

Implementación de Seguridad

Ley de Protección de Datos Personales

- **Medidas de Seguridad**: Implementar medidas técnicas y organizativas adecuadas.
- **Auditorías**: Realizar auditorías periódicas para asegurar el cumplimiento de la ley.

Reemplazo de Proveedor

Actuar Ético

- **Transparencia**: Informar al cliente de manera clara y transparente.
- **Continuidad del Servicio**: Asegurar una transición ordenada y minimizar el impacto en el cliente.

Denuncia por Divulgación de Datos Personales

Acción Jurídica

- **Defensa**: Demostrar el cumplimiento de la Ley 25.326 y las medidas de seguridad implementadas.
- **Acción Correctiva**: Implementar medidas correctivas y compensar al afectado si es necesario.