TECNOLÓGICO SUPERIOR CORDILLERA

25 AÑOS

CIDE
Centro de Investigación y Desarrollo Ecuador

SEMINARIO INTERNACIONAL DE SEGURIDAD EN EL DESARROLLO DE SOFTWARE

Seguridad en Ingeniería de Software

Ricardo Botero Tabares

Ingeniero de Sistemas, Especialista en Didáctica Universitaria y Magíster en Ingeniería; doctorando en Gestión de la Tecnología y la Innovación.

Profesor de la Facultad de Ingeniería del Tecnológico de Antioquia Institución Universitaria en Medellín – Colombia, y docente en varias instituciones de educación superior, donde ha impartido cursos relacionados con algoritmos, estructuras de datos, matemáticas discretas e ingeniería de software.

Coordinador del Grupo de Investigación en Ingeniería de Software del TdeA - GIISTA. Autor de libros y artículos; expositor en encuentros y congresos de carácter nacional e internacional relacionados con ciencias de la computación.

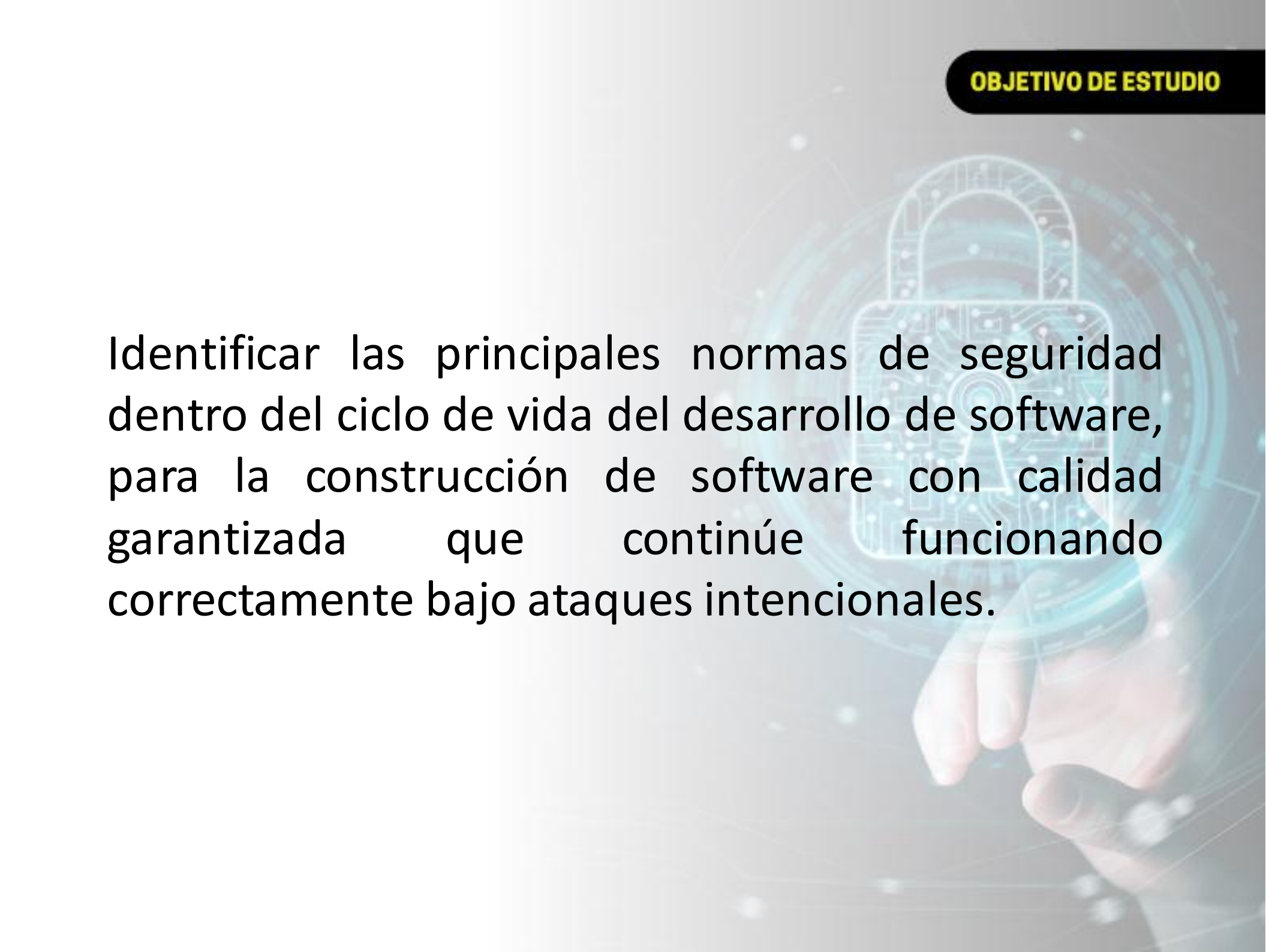
Seguridad en Ingeniería de Software

Yeiler Quintero Barco

Ingeniero de Sistemas, Especialista en Seguridad de la Información, Magister en Gestión de Tecnologías de la Información y candidato a Doctor en Tecnología Educativa.

Docente de la Facultad de Ingeniería del Tecnológico de Antioquia Institución Universitaria, la Escuela de Posgrado de la Corporación Universitaria Americana y docente de la Facultad de Ingeniería de la Fundación Universitaria María Cano, en Medellín – Colombia; con más de 12 años de experiencia en docencia, trabajando con temas de infraestructura de TI y seguridad informática.

OBJETIVO DE ESTUDIO



Identificar las principales normas de seguridad dentro del ciclo de vida del desarrollo de software, para la construcción de software con calidad garantizada que continúe funcionando correctamente bajo ataques intencionales.

La metodología conllevó las siguientes fases:

- Revisión sistemática de la literatura relacionada con Seguridad en el Ciclo de Vida de Desarrollo de Software (S-SDLC).

- Identificación de las normas de seguridad a tener en cuenta durante el ciclo de vida del software.

- Desarrollo de una aplicación donde se implementen las normas de seguridad identificadas (en desarrollo).

Seguridad en Ingeniería de Software

Los procesos de la ingeniería de software deben conllevar un conjunto de medidas preventivas y reactivas que permitan resguardar y proteger la información, buscando mantener la *confidencialidad*, *disponibilidad* e *integridad* de la misma.

Características intrínsecas del software en su ciclo de vida

- **Garantía**: el software debe garantizar confiabilidad durante su operación a pesar de la presencia de ataques intencionales (tolera y resiste ataques, recupera nivel normal de operación después de un ataque no fallido).

- **Seguridad**: el software permanece correcto y predecible a pesar de los ataques que comprometan su confiabilidad.

Propiedades fundamentales del Software seguro

- **Confidencialidad**: las características, activos y contenidos que administra el software, son accesibles solo a entidades autorizadas.

- **Integridad**: resistencia y flexibilidad a modificaciones no autorizadas del código, o a configuraciones y actualizaciones del software por entidades autorizadas.

- **Disponibilidad**: el software debe estar accesible y operativo a los usuarios autorizados.

Propiedades del Software seguro relacionadas con los usuarios

- **Trazabilidad**: incluye el registro de las acciones relevantes relacionadas con la seguridad de una entidad que actúa como usuario, con el objetivo de establecer responsabilidades.

- **No repudio**: conlleva prevenir que una entidad (o usuario) pueda desmentir la responsabilidad de acciones que han sido ejecutadas.

Propiedades conducentes del Software

- **Fiabilidad**: el software siempre ha de operar de la manera esperada.
- **Correcto**: debe cumplir con los requerimientos del cliente incluso bajo condiciones no previstas.
- **Predecible**: las funcionalidades, propiedades y comportamientos del software siempre serán los esperados.
- **Protección**: cuando el sistema ha sufrido una falla o un ataque, debe detenerse o quedar parcialmente operativo en un estado seguro para impedir catástrofes (pérdida de vidas humanas o de activos valiosos, compromiso de la sustentabilidad del ambiente, etc.).

S-SDLC (Secure – Software Development Life Cycle)

La seguridad se ha venido incorporando al ciclo de vida del software:
Ing. de Software e **Ing. de Seguridad** deben ser paralelas

Modelos tradicionales de ciclo de vida del software

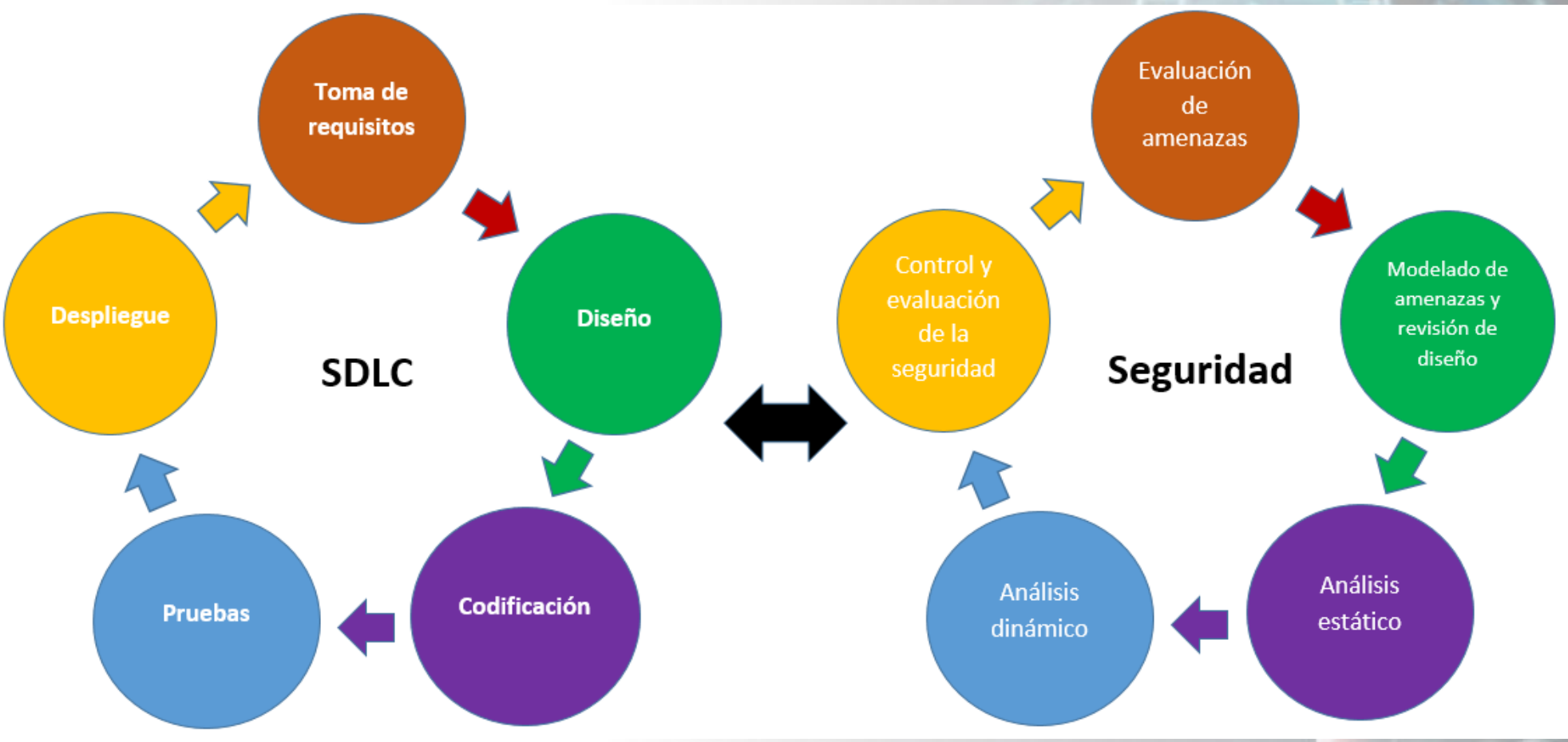
- Cascada
- Espiral
- Iterativo o por prototipos
- Unificado
- Ágiles
  - Scrum
  - Kambas
  - XP

Modelos de Ciclo de vida de Desarrollo de Software

Existen modelos nuevos definidos y publicados para el SDLC, que proporcionan marcos de trabajo articulados con las tradicionales:

- **Informática Confiable de Microsoft** (MS Trustworthy Computing SDL)
- **Proceso de Seguridad de Aplicación Comprensivo y Ligero** (Comprehesive, Lightweight Application Security Process - CLASP): disponible como un plug-in para Rational Unified Process.
- **Proceso de Garantía de Seguridad del Software Oracle** (Oracle Software Security Assurance Process).
- Otros: **Rational Unified Process Secure** (RUPSec), **Secure Software Development Model** (SSDM)

Paralelismo entre las fases del SDLC y la seguridad



Seguridad en la toma de requisitos

Seguridad y riesgos

Requisitos de seguridad

- Gestión del password y autenticación.
- Gestión de roles (Gerente de: interface, diseño, implementación, pruebas, etc).
- Sistema de logs.

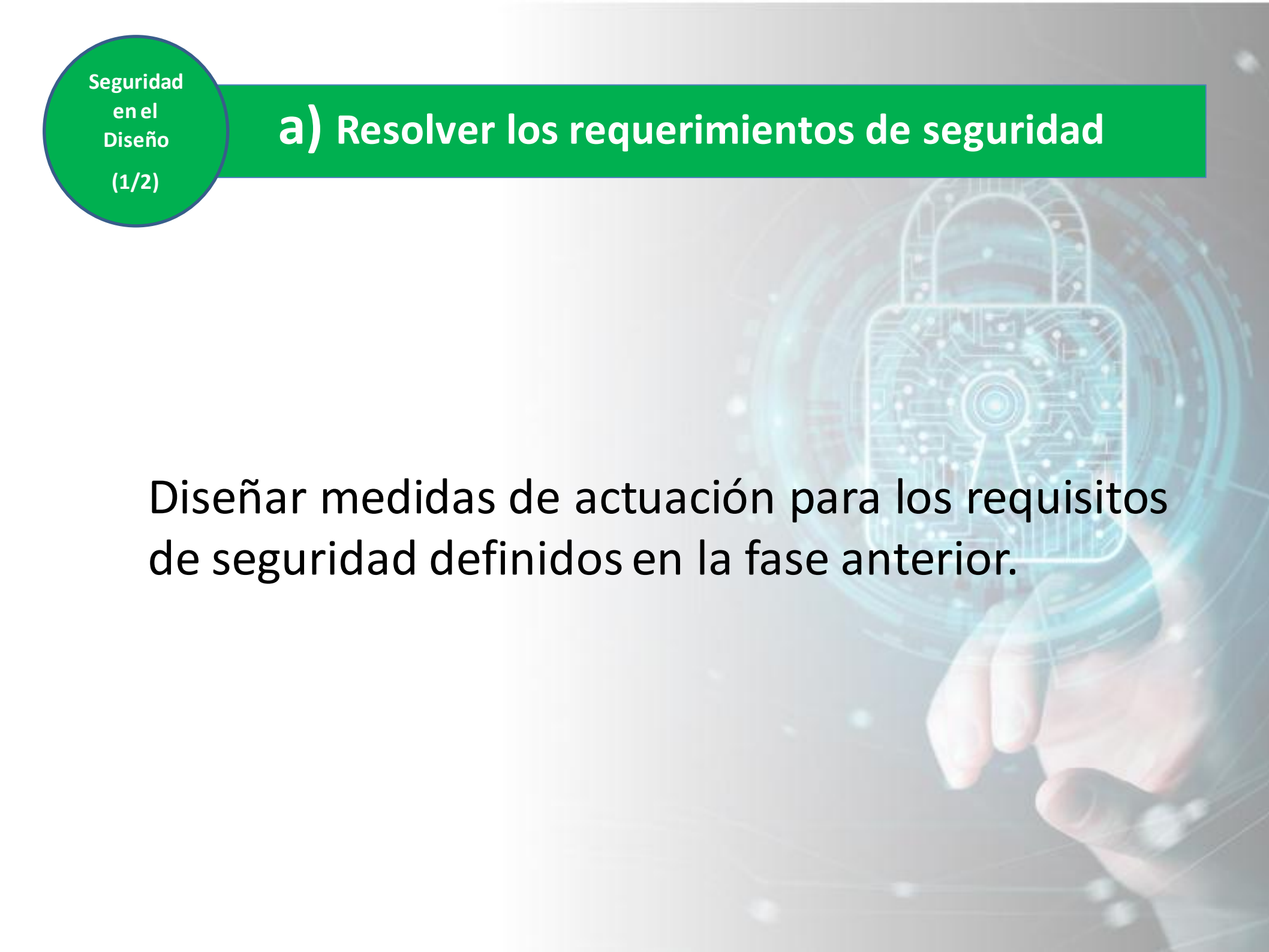
Evaluación de riesgos

- Cesión de datos a terceros.
- Política de confidencialidad.
- Planes de recuperación ante desastres.
- Seguridad de los datos de los usuarios.

Seguridad en el Diseño

(1/2)

a) Resolver los requerimientos de seguridad

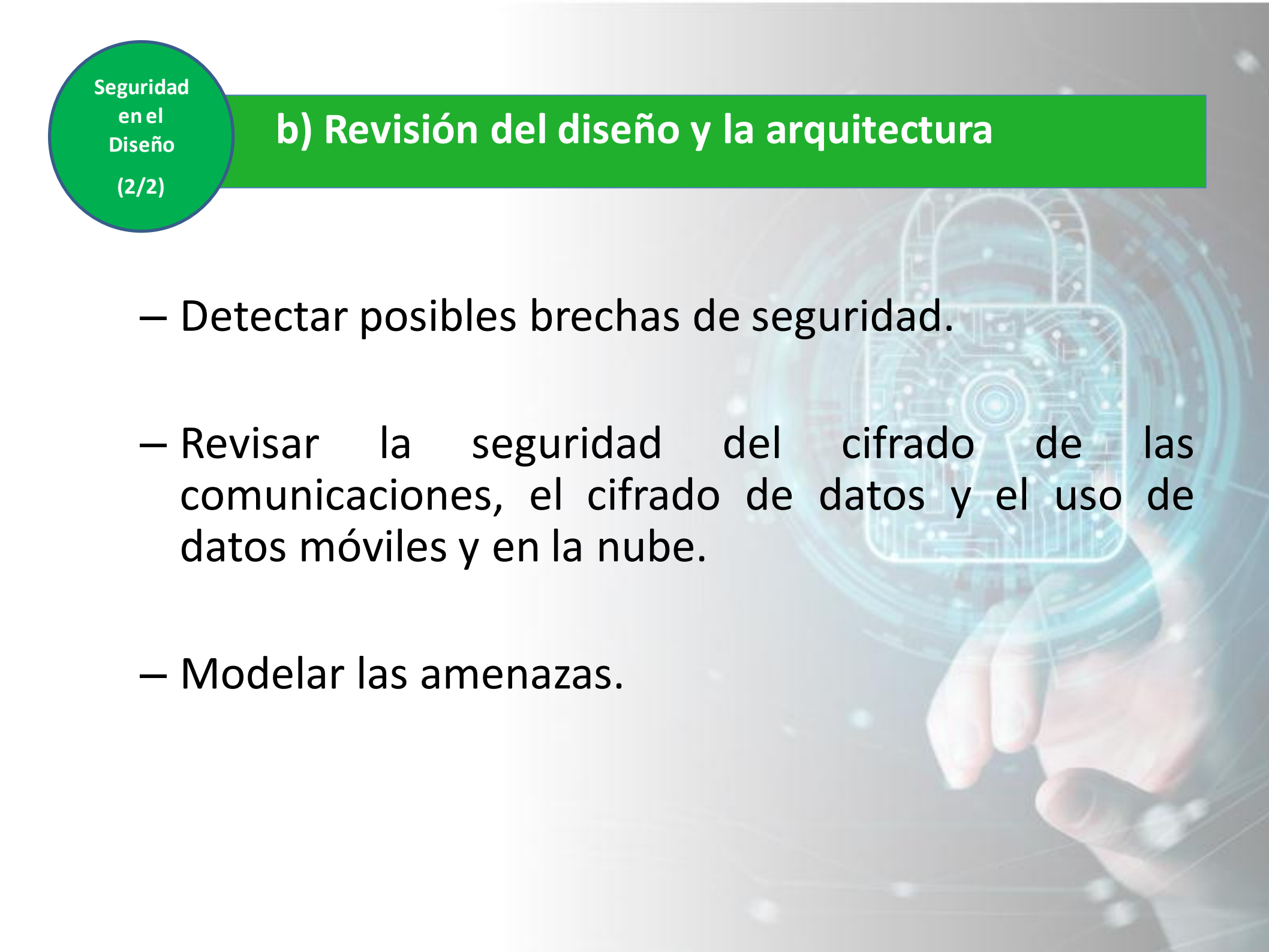


Diseñar medidas de actuación para los requisitos de seguridad definidos en la fase anterior.

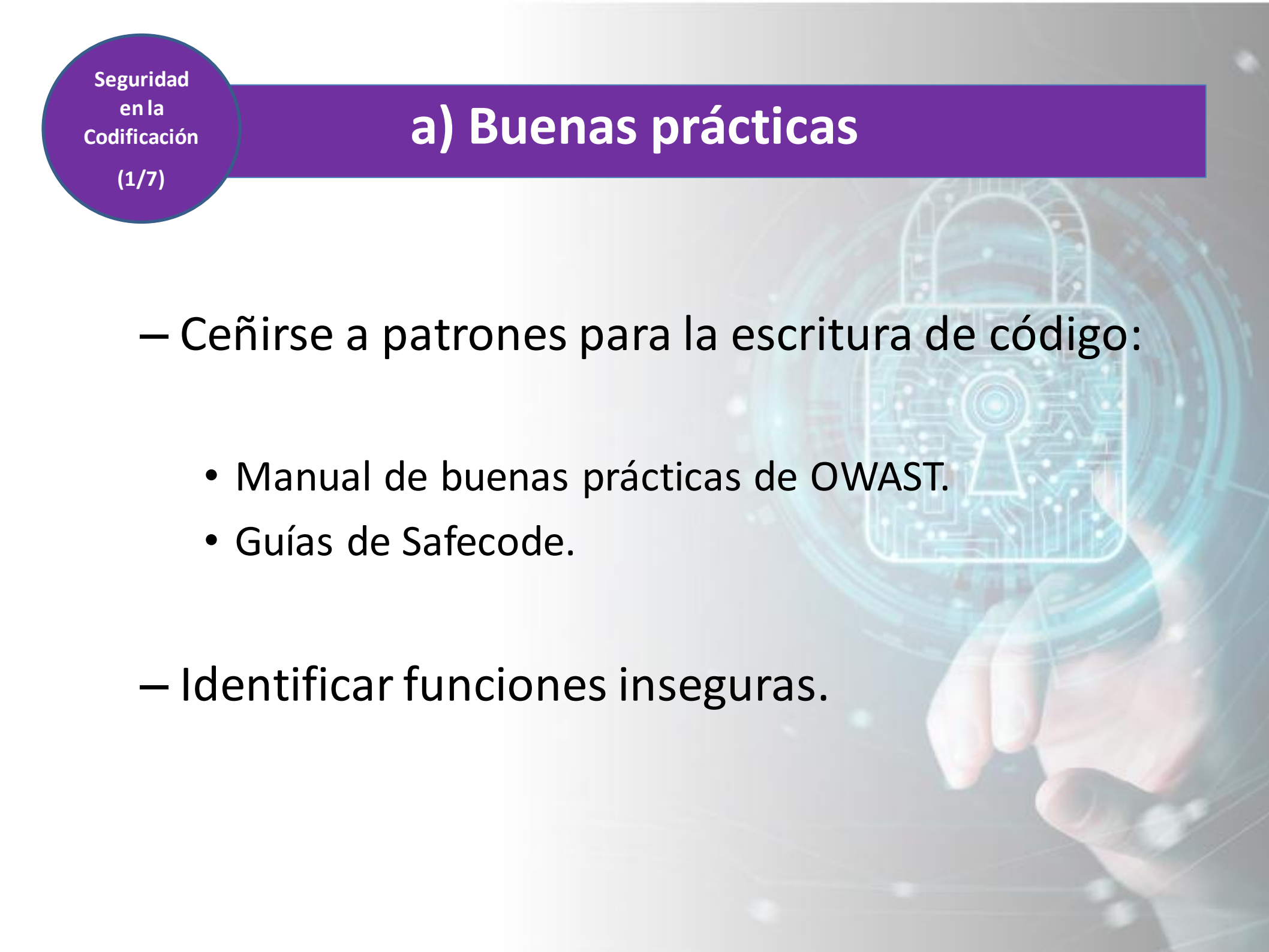
Seguridad en el Diseño (2/2)

b) Revisión del diseño y la arquitectura

- Detectar posibles brechas de seguridad.
- Revisar la seguridad del cifrado de las comunicaciones, el cifrado de datos y el uso de datos móviles y en la nube.
- Modelar las amenazas.



a) Buenas prácticas



– Ceñirse a patrones para la escritura de código:

  - Manual de buenas prácticas de OWAST.
  - Guías de Safecode.

– Identificar funciones inseguras.

Seguridad en la Codificación (2/7)

Ejemplo 1:

Primera buena práctica de seguridad en POO: **encapsulación de datos**.

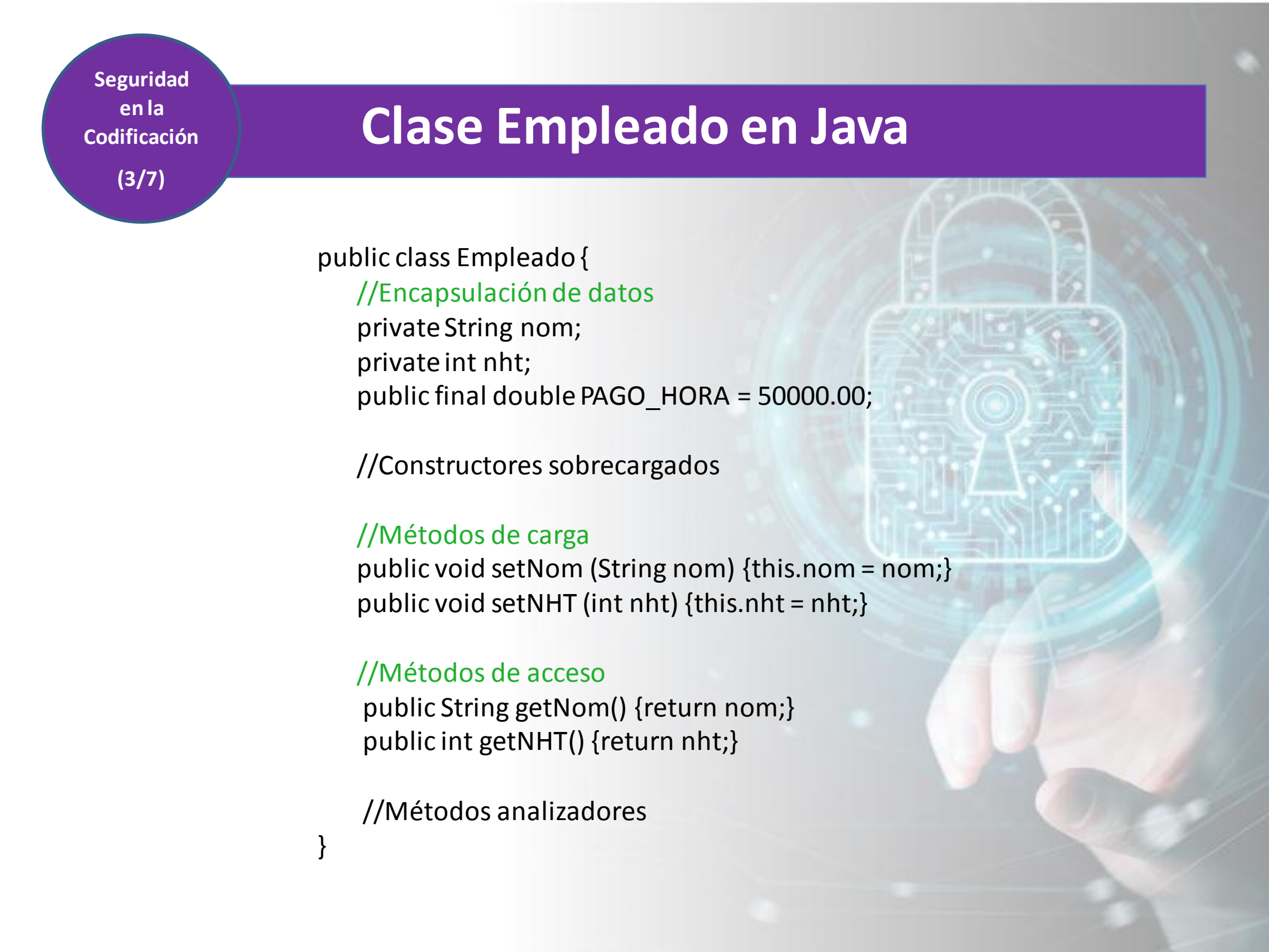
- Definir datos con visibilidad privada o protegida.
- Para procesar los datos privados utilizar los métodos de carga (set) y de acceso (get).

Seguridad en la Codificación

(3/7)

Clase Empleado en Java

```
public class Empleado {
    //Encapsulación de datos
    private String nom;
    private int nht;
    public final double PAGO_HORA = 50000.00;

    //Constructores sobrecargados

    //Métodos de carga
    public void setNom (String nom) {this.nom = nom;}
    public void setNHT (int nht) {this.nht = nht;}

    //Métodos de acceso
     public String getNom() {return nom;}
     public int getNHT() {return nht;}

     //Métodos analizadores
}
```

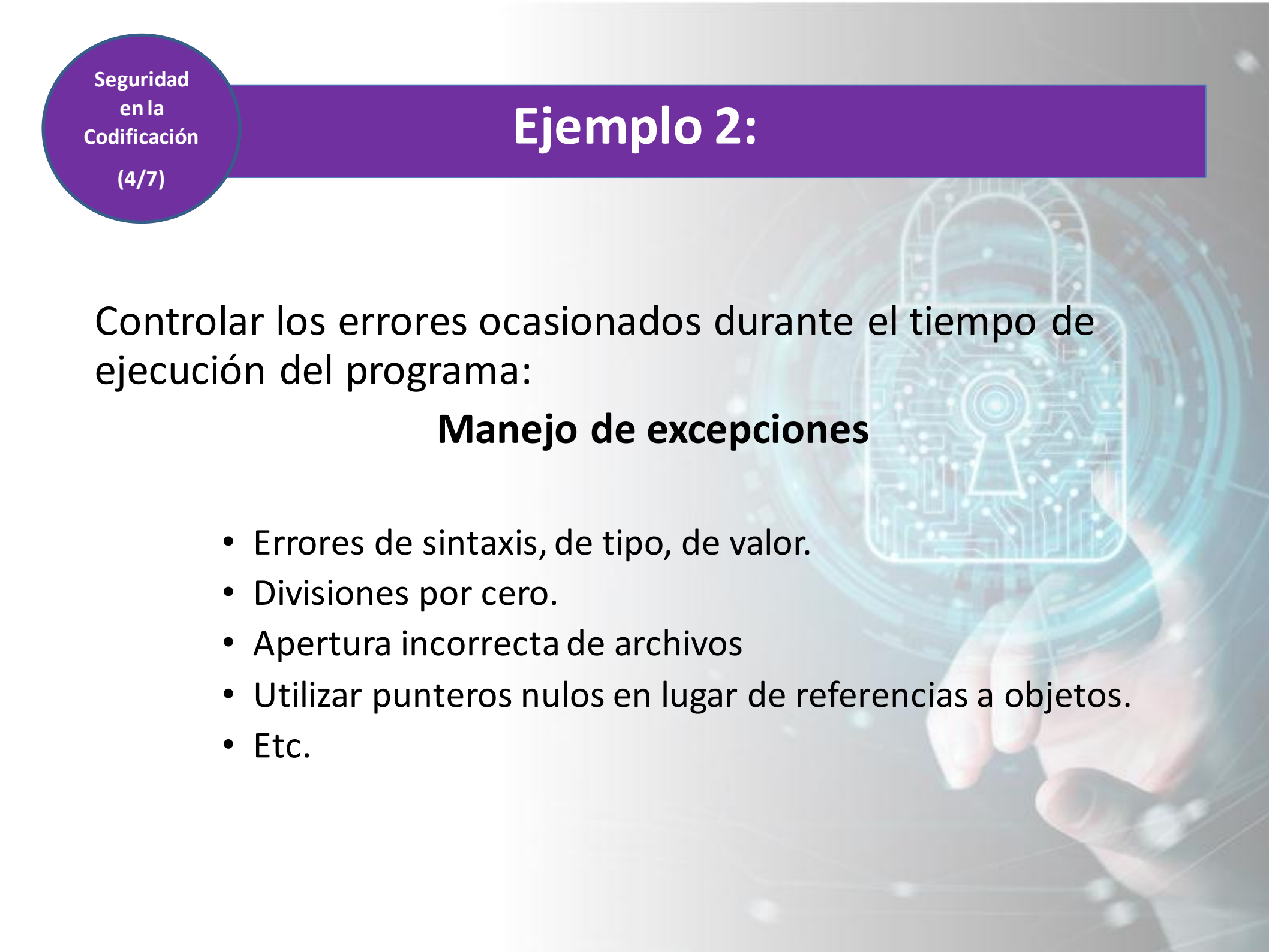


Ejemplo 2:

Controlar los errores ocasionados durante el tiempo de ejecución del programa:

Manejo de excepciones

- Errores de sintaxis, de tipo, de valor.
- Divisiones por cero.
- Apertura incorrecta de archivos
- Utilizar punteros nulos en lugar de referencias a objetos.
- Etc.



Seguridad en la Codificación

(5/7)

Manejo de una excepción

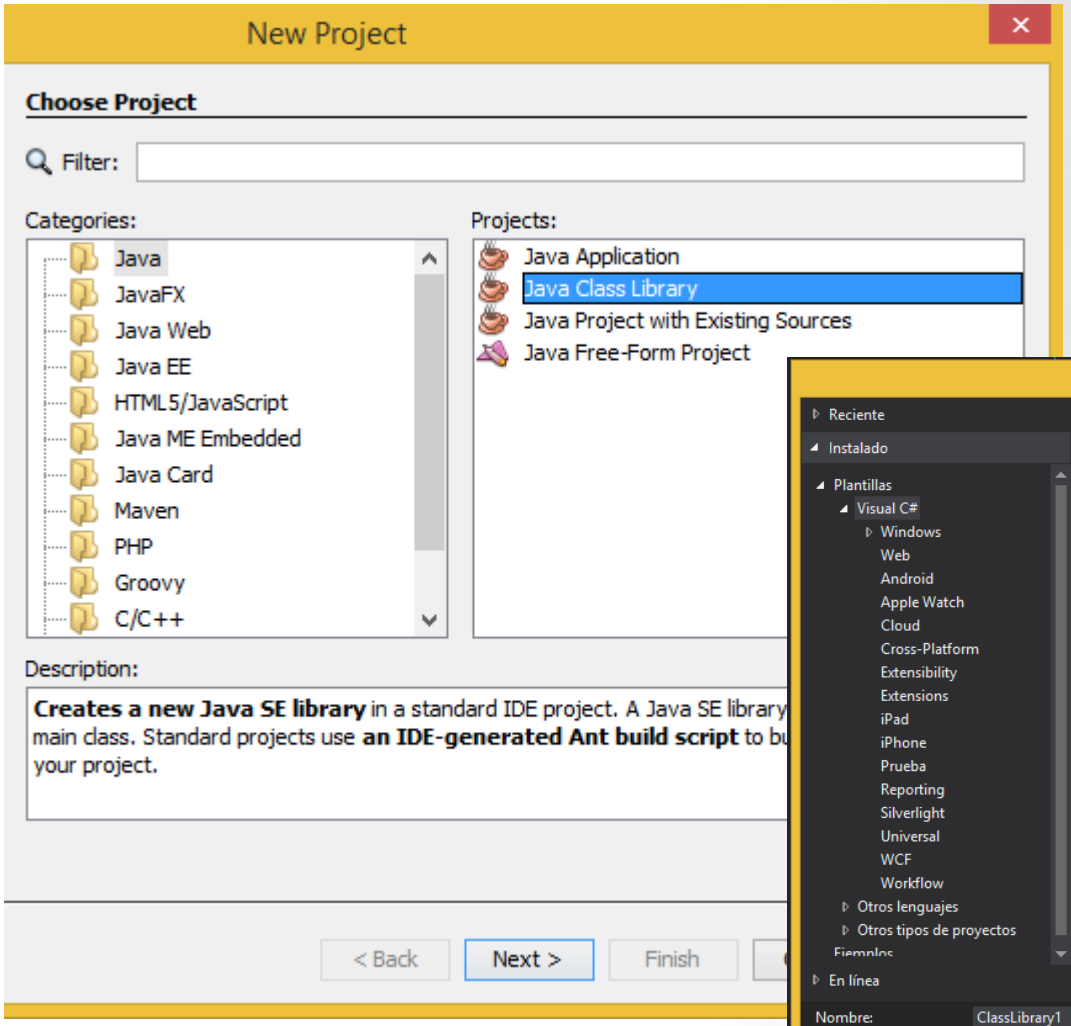
```
public class EjemploExcepcion {
    public static void main(String[] args) {
        String respuesta;
        int numerador, denominador, cociente;

        try {
                numerador = Consola.leerEntero();
                denominador = Consola.leerEntero();
                cociente = numerador/denominador;
                respuesta = String.valueOf(cociente);
        } catch(NumberFormatException ex){
                respuesta="Se han introducido caracteres no numéricos";
        } catch(ArithmeticException ex){
                respuesta = "División entre cero";
        }
        System.out.println(respuesta);
    }
}
```

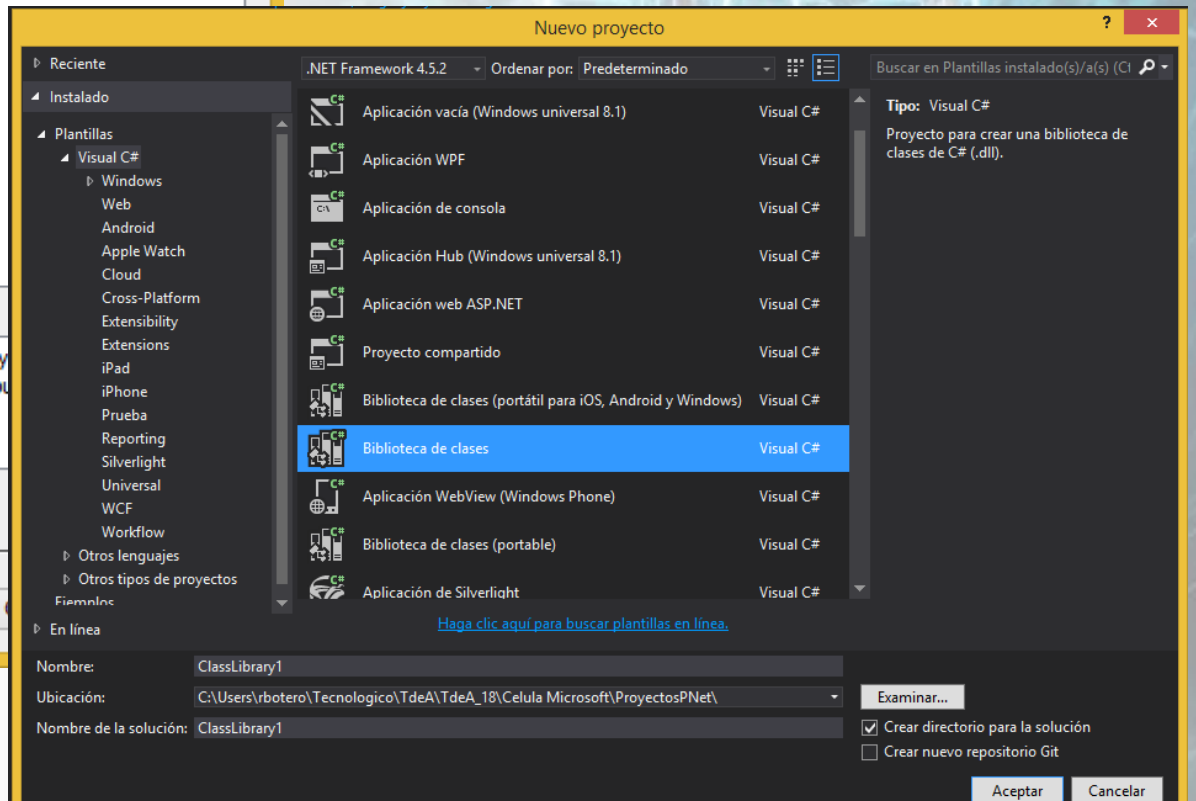
Seguridad en la Codificación

(6/7)

Ejemplo 3

Manejo de Librerías o bibliotecas de clases.

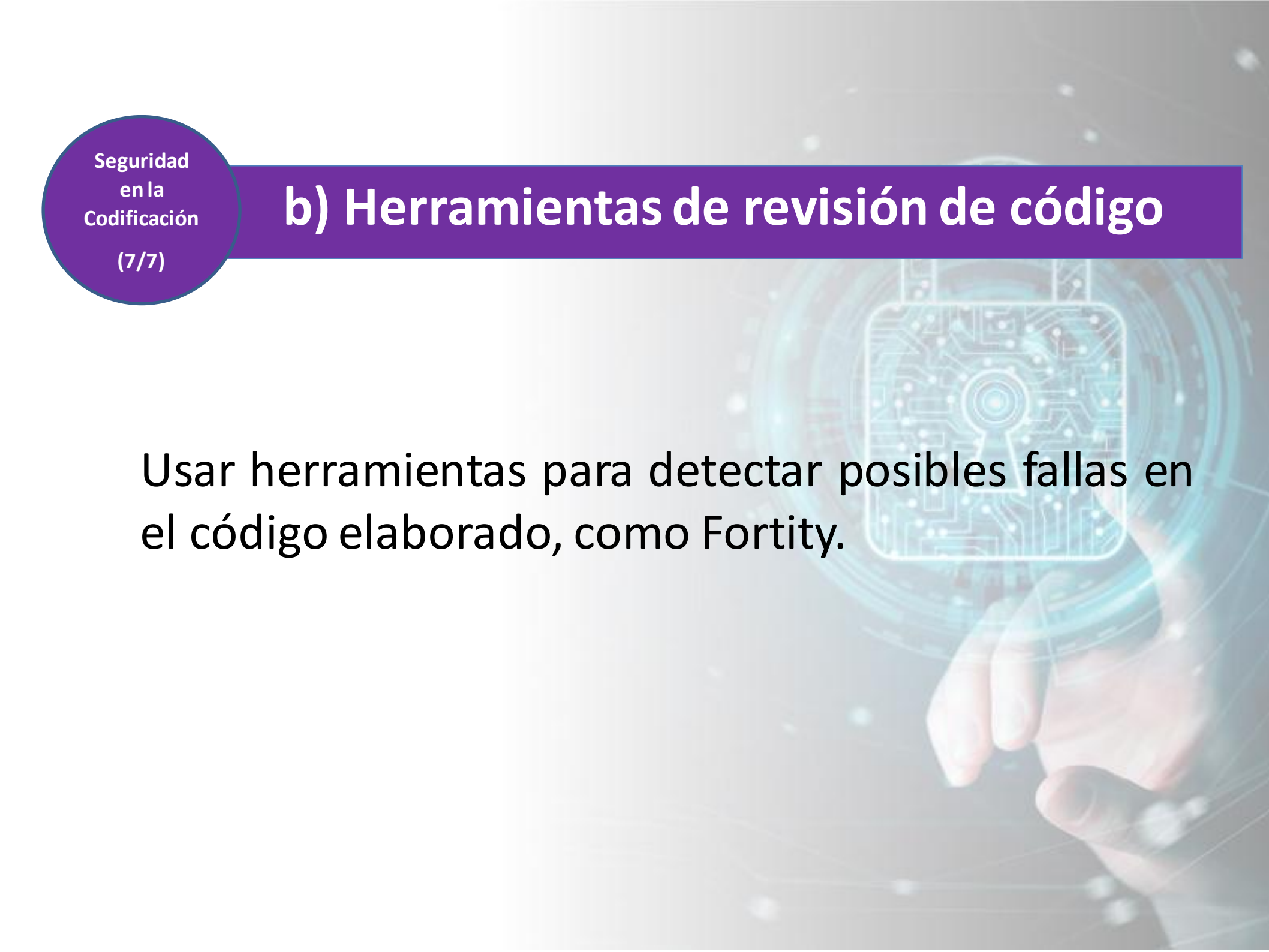




Seguridad en la Codificación (7/7)

b) Herramientas de revisión de código

Usar herramientas para detectar posibles fallas en el código elaborado, como Fortity.



Seguridad en las Pruebas (1/2)

a) Evaluación de vulnerabilidades

Realizar pruebas automatizadas para identificar puntos vulnerables del software.

Herramientas:

- **Rational Functional Tester**
- **Visual Studio Test Proffessional**
- **Watir**
- **HP Quicktest Professional (QTP)**
- **Selenium**

Seguridad en las Pruebas (2/2)

b) Fuzzing

Se obtienen excepciones que pueda devolver el equipo, como llenados de pila (stack overflow) o respuestas del programa al recibir campos incorrectos (un *integer* en lugar de un *float*).

Seguridad en el Despliegue (1/2)

a) Revisión de la configuración del servidor

Antes de subir la aplicación a producción, se debe revisar la configuración de la seguridad, evitando errores comunes, como devolver más información de la debida en caso de error.

Seguridad en el Despliegue (2/2)

b) Revisión de la configuración de la red

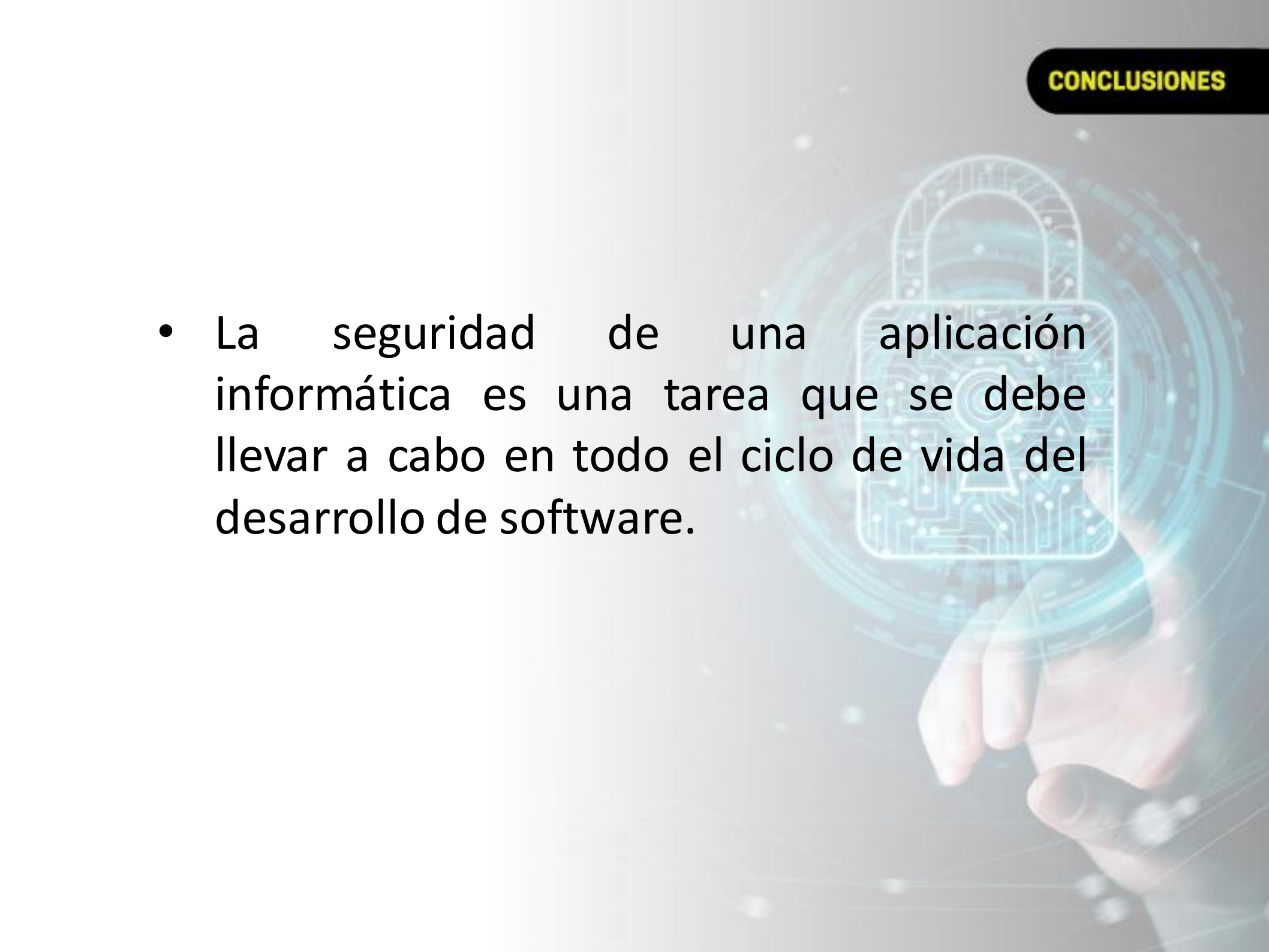
Antes de subir la aplicación a producción, se debe revisar que la configuración de la red sea segura. Por ej.:

- Si se va a colocar la aplicación en una DMZ, hay que configurarla de manera correcta.

- Si se va a usar un servidor en la nube, hay que configurar los parámetros de conexión.

CONCLUSIONES

- La seguridad de una aplicación informática es una tarea que se debe llevar a cabo en todo el ciclo de vida del desarrollo de software.



CONCLUSIONES

- Se deben reducir las vulnerabilidades potenciales tan temprano como sea posible dentro del SDLC, mediante la adopción de procesos y prácticas mejoradas para la seguridad.

  Esto resulta más adecuado que la modalidad tradicional de desarrollar el software, y luego liberar parches para el software durante su despliegue y operación.

CONCLUSIONES

- La garantía de seguridad en el software brinda solidez tecnológica con altos niveles de competitividad empresarial: forma parte de una **cultura de calidad y mejora continua y conjunta**.

¡Muchas Gracias!

Ing. – MSc Ricardo Botero Tabares

rbotero@tdea.edu.co

Ing. – MSc Yeiler Quintero Barco

yeonet@hotmail.com

Tecnológico de Antioquia
Institución Universitaria

www.tdea.edu.co

TECNOLÓGICO SUPERIOR CORDILLERA

25 años

CIDE
Centro de Investigación y Desarrollo Ecuador

SEMINARIO INTERNACIONAL DE SEGURIDAD EN EL DESARROLLO DE SOFTWARE

WWW.CIDECUADOR.COM

Una vez finalizado el evento esta presentación sera publicada en su respectiva página web