

Seguridad en el desarrollo de aplicaciones

•I. Principios de codificación seguridad


•Desde la otra perspectiva, la industria del desarrollo de software evoluciona de forma muy rápida, dada la demanda y las oportunidades que ofrece el mercado en forma de necesidades no cubiertas. Hay muchísima competencia, ya que muchas empresas están intentando desarrollar sus propias soluciones para llevarse su parte del pastel. Desgraciadamente, muchos esfuerzos concienzudos y minuciosos se van al cubo de la basura en el desarrollo de software. Varias fuentes del sector señalan que aproximadamente un 80% de los proyectos de software no tienen éxito por malas previsiones, ejecuciones de proyecto muy mejorables, presupuestos y recursos limitados, o funcionalidades inapropiadas.

1. Unidad de aprendizaje	I. Principios de codificación segura
2. Horas Teóricas	8
3. Horas Prácticas	12
4. Horas Totales	20
5. Objetivo de la Unidad de Aprendizaje	El alumno elaborará un plan de desarrollo para implementar mecanismos de seguridad en aplicaciones.

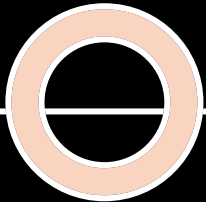

Temas	Saber	Saber hacer	Ser
Buenas prácticas en el desarrollo de software seguro.	Identificar las buenas prácticas en el desarrollo de software orientadas a seguridad. Identificar los estándares de codificación del lenguaje elegido.	Seleccionar las buenas prácticas y estándares aplicables al desarrollo de aplicaciones.	Ética Responsabilidad Proactivo Analítico Sentido de la planificación Capacidad de autoaprendizaje Uso de procesos cognitivos Razonamiento lógico
Protección de vulnerabilidades.	Describir el ciclo de vida de seguridad en el desarrollo de software (S-SDLC). Identificar las técnicas y mecanismos de protección de vulnerabilidades.	Planear aplicaciones seguras tomando en cuenta las etapas del ciclo de vida de seguridad en el desarrollo de software (S-SDLC).	Ética Responsabilidad Proactivo Analítico Sentido de la planificación Capacidad de autoaprendizaje Uso de procesos cognitivos Razonamiento lógico

#1

Seleccionar el talento y los recursos apropiados

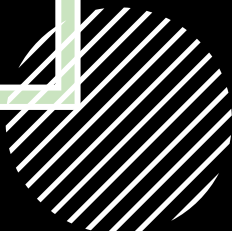



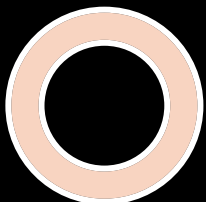

- Captar y seleccionar el talento humano con las destrezas necesarias y experiencia relevante es vital para garantizar el éxito del proyecto. Es importante asignar el trabajo apropiado a la persona indicada. Por otro lado, invertir en herramientas que aumentan la productividad y eficiencia del equipo de desarrollo es muy importante. Buenos equipos, hardware moderno, software y plataformas de desarrollo y de pruebas actualizado, y herramientas automatizadas ayudan a que el equipo pueda imprimir todo su conocimiento y buenas prácticas para garantizar un producto sólido, fiable y robusto.



#2 Escoger el proceso de desarrollo apropiado


•El ciclo de vida del desarrollo del software tiene una fuerte dependencia del proceso elegido. El modelo en cascado, la metodología ágil, el enfoque iterativo en espiral, son todas formas contrastadas de alcanzar el éxito. La dificultad está en elegir bien qué metodología le conviene más a cada tipo de proyecto. Esto es de suma importancia. La adhesión efectiva y el hecho aplicar a rajatabla la metodología elegida es lo que determina que el proyecto llegue a buen puerto. En ocasiones es bueno hacer un pequeño prototipo para evaluar la viabilidad o investigar nuevas tecnologías.





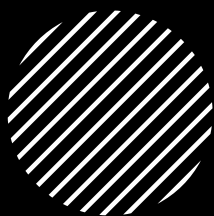
#3 Hacer presupuestos y estimaciones razonables

• Muchos proyectos fracasan o se prolongan en plazos por hacer estimaciones poco realistas. Una planificación razonable depende de fijar bien los tiempos, el presupuesto, los recursos y los esfuerzos. Lo mejor es usar técnicas de estimación y presupuestarias contrastadas. Intentar apretar las estimaciones para intentar acortar un proyecto normalmente termina en catástrofe.





#4 Fijar hitos más pequeños

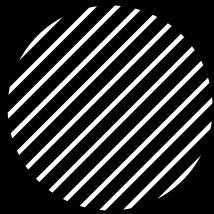


- Los grandes proyectos e hitos deben complementarse con mini-hitos para poder hacer mejor seguimiento, más control y mejor gestión de riesgos, y en general para mitigar incidencias de una forma más controlada. Los miembros del equipo deben reunirse para fijar estos mini-hitos y alinearlos con los grandes hitos para cumplir plazos y reducir los retrasos que pueden surgir por las interdependencias de las tareas que tienen asignadas.





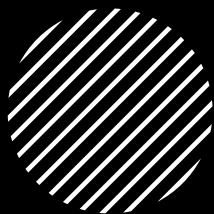
#5 Definir bien los requisitos



- Documentar de manera efectiva los requisitos es la columna vertebral para poder alinear el producto final con los objetivos empresariales. Es imperativo que se reúnan todas las partes (clientes, los responsables de empresa y los líderes de los equipos) para documentar los requisitos de forma clara y concisa, sin dejar lugar a lagunas o a la improvisación.
- Es necesario definir los requisitos básicos, los derivados y los implícitos, tanto funcionales como no funcionales. La funcionalidad se puede obtener mediante escenarios de casos de uso. Los requisitos en torno al rendimiento, funcionamiento a prueba de fallos, de sistema, diseño y arquitectura, todos deben estar bien documentados y tenidos en cuenta.



#6 Definir la arquitectura del sistema

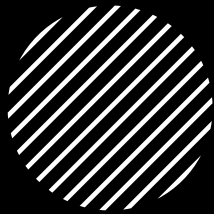


- Un buen arquitecto de aplicaciones garantizará una elección de arquitectura del sistema apropiada, teniendo en cuenta tanto los requisitos como las limitaciones y restricciones, si las hubiera. Buenas prácticas, tales como la identificación de amenazas y anti-patrones dentro del sistema, son muy útiles.





#7 Optimizar el diseño de la aplicación

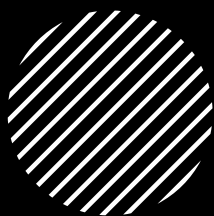


- El diseño debe ser modular y estar optimizado.
- Equilibrar y distribuir funcionalidad entre varios módulos puede hacer que el proyecto funcione bien o que rompa.
- Un enfoque orientado a objetos es una técnica que garantiza modularidad. Depende de los diseñadores garantizar que el enfoque elegido se aplique bien para poder lograr la máxima cohesión con un acoplamiento mínimo. La reutilización del código es un aspecto muchas veces infrutilizado en el diseño de la aplicación, y si está bien implementado, puede ahorrar mucho esfuerzo y reducir costes a la larga en cualquier proyecto.




8

Implementar el código de manera efectiva

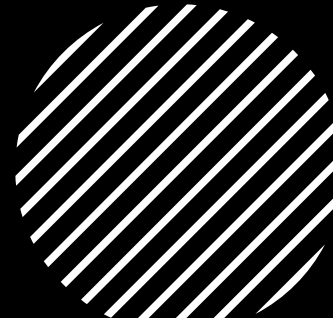


- El uso de módulos más pequeños que están autoprobadados, probados unitariamente y que se integran continuamente es una buena práctica muy extendida. La automatización de herramientas *build* y la ejecución automatizada de pruebas de regresión para cada funcionalidad incluida se recomienda para garantizar que la funcionalidad ya implementada no rompa.





#9 Pruebas rigurosas y validación



- La planificación de pruebas, la creación de conjuntos de pruebas y la ejecución de las mismas son muy importantes con el fin de validar la funcionalidad desarrollada.

- De hecho, la planificación de las pruebas debe hacerse en paralelo a la fase de desarrollo. Igual de importante es la documentación que hagamos de las pruebas, informar de forma efectiva los errores, el rastreo de los errores y la corrección de los mismos.

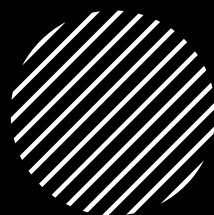
- El uso de herramientas automatizadas al igual que procesos contrastados que aseguren que los errores se identifiquen en la fase más temprana posible y resueltos con el menor coste.

- Las pruebas unitarias, las de integración, las de funcionalidades, las del sistema y las del rendimiento son algunos tipos de pruebas. Cada nivel de prueba requiere su pericia, planificación y ejecución.



#10

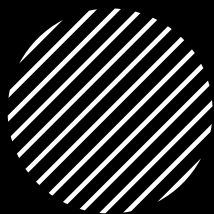
Documentación



•Aún siendo importante el propio software, igual de importante es toda la documentación sobre el que se apoya –el plan del proyecto, requisitos y especificaciones, Diseño de Alto Nivel (HLD), Diseño de Bajo Nivel (LLD), planes de pruebas, informes de las pruebas, informes de estado y la documentación para los usuarios. Muchas veces estos documentos son parte de los entregables especificados por el cliente o las partes interesadas en un proyecto determinado. Estos documentos ayudan a garantizar el entendimiento del software, trazabilidad y eliminar la dependencia del equipo de desarrollo original. Pueden usarse como referencia en el futuro por otras personas que necesiten mantener, mejorar o usar el software.



#11 Instalación y despliegue eficaz

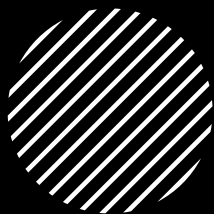


•En multitud de ocasiones cuando ya hemos probado el software de puertas a dentro y todo va bien, de repente el proyecto fracasa en casa del cliente o cuando estamos en fase de implementación y despliegue. Es muy importante tener un buen plan de despliegue y hacer una lista a modo de “checklist” para evitar desastres.





#12 DRY – Don repeat yourself



- No repitas código. Repetir partes de código a lo largo de un desarrollo solo sirve para dificultar el mantenimiento y aumentar la probabilidad de cometer errores.

- Es mejor agrupar en funciones las operaciones que se repitan, y aíslala del resto del código, el esfuerzo necesario para el mantenimiento del código va a disminuir.



4 Principios para el Diseño y Desarrollo de Software

Aplícalos para una codificación más eficiente y sostenible



KISS

Keep it Simple, Stupid!
Evita complicar
el código



DRY

Don't Repeat Yourself
Procura no
duplicar código



YAGNI

You Aren't Gonna Need It
Desarrolla solo
el código necesario

S

Single
Responsibility

O

Open/
Closed

L

Liskov
Substitution

I

Interface
Segregation

D

Dependency
Inversion

Juego de participación – khoot

Actividad de clase

- Mapa Conceptual
- Infografía
- Cuadro informativo

Mencionando las buenas prácticas en el desarrollo de software