**CLASE 2 - MODELOS DE DESARROLLO DE SOFTWARE**

**✨ Objetivos de la clase:**

* Comprender los principales modelos de desarrollo de software.
* Diferenciar los enfoques secuenciales, iterativos y ágiles.
* Analizar ventajas, desventajas y contextos de aplicación de cada modelo.
* Aplicar los modelos en situaciones simuladas de desarrollo.
* Fomentar el trabajo en equipo y el pensamiento crítico.

**🔄 1. Repaso de la clase anterior**

* Breve exposición oral de cada grupo sobre el modelo investigado como tarea.
* El docente ordena las ideas en una pizarra digital (Miro/Jamboard).
* Preguntas orientadoras: ¿Qué caracteriza al modelo? ¿En qué tipo de proyecto se usaría?

**🔬 2. Desarrollo teórico y práctico de los modelos**

**✏️ Actividad disparadora**

Ver video breve: "Waterfall vs Agile" (YouTube).

https://www.youtube.com/watch?v=Z9QbYZh1YXY

* Pregunta al grupo:
  + ¿Cuál creen que se usaría en un proyecto escolar?
  + ¿Y en uno para una empresa de transporte?

**🎓 Presentación guiada por el docente**

**Modelos Secuenciales:**

* **Waterfall (Cascada)**:
  + Etapas: Requisitos → Diseño → Desarrollo → Pruebas → Mantenimiento.
  + Ventajas: claridad de etapas, adecuado para proyectos bien definidos.
  + Desventajas: difícil adaptación ante cambios, retroalimentación tardía.
  + Ejemplo: Software de facturación con requisitos cerrados.

**📘 1. Modelo Waterfall – "La fábrica de tortas"**

Había una vez un pastelero llamado Don Cascada. Él hacía las mejores tortas de bodas de toda la ciudad, pero era muy estructurado:  
Primero tomaba el pedido (requisitos), luego hacía el diseño del pastel en una hoja (diseño), después lo horneaba (desarrollo), lo decoraba y lo probaba (pruebas), y por último, lo entregaba (mantenimiento).

Todo seguía ese orden. Si en medio del horneado el cliente pedía cambiar el sabor, Don Cascada decía: “¡Ya es tarde, eso no se puede!”

**🎯 Actividad: “La receta inalterable”**

**Consigna:**  
Imaginen que son una empresa que desarrolla una app de facturación con requisitos ya definidos por una entidad x.

1. Armen una línea de tiempo con las 5 etapas del modelo Cascada.
2. Escriban 1 problema que ocurriría si al llegar al final alguien pide cambiar algo del diseño.

RESPUESTA:

Requisitos → Diseño → Desarrollo → Pruebas → Mantenimiento.

1 mes → 4 meses → 6 meses → 1 mes → sin tiempo.

Si el cliente quiere agregar un campo más a los datos de facturación

* **Modelo en V:**
  + Similar a cascada pero con validación en cada fase.
  + Útil para sistemas críticos donde se requiere trazabilidad.

**📘 2. Modelo en V – "El mecánico meticuloso"**

Juan era mecánico. Cada vez que reparaba un auto, hacía una verificación tras cada arreglo.  
Si ajustaba los frenos, luego los probaba. Si cambiaba el aceite, luego hacía una revisión.  
Su trabajo formaba una “V”: por cada paso de desarrollo, tenía un paso de validación.

**🎯 Actividad: “Verifica todo”**

**Consigna:**  
Analicen cómo el Modelo en V sería útil en el desarrollo del software de un avión.

1. ¿Qué etapa debería validarse inmediatamente después del diseño técnico?
2. ¿Por qué sería riesgoso no validar cada etapa?

RESPUESTA

Requisitos → Diseño → Desarrollo → Producción → Mantenimiento.

Pruebas Pruebas Pruebas Pruebas

* **Modelo Incremental:**
  + El sistema se construye por módulos funcionales.
  + Mejora la respuesta al cambio, se entrega valor desde el inicio.
  + Ejemplo: App con login → menú → carrito → pago.

**📘 3. Modelo Incremental – "El castillo modular"**

Lola era arquitecta de castillos de juguete. Pero no los hacía de una sola vez.  
Primero construía la torre, luego el puente levadizo, después los jardines y por último la muralla.  
Cada módulo funcionaba por separado, pero todos juntos formaban un castillo completo.

**🎯 Actividad: “App por partes”**

**Consigna:**  
Tu equipo está creando una app tipo MercadoLibre.

1. Enumeren 4 funcionalidades principales.
2. Dibujen cómo podrían entregarse en 4 versiones incrementales.

RESPUESTA

Login-– Representar informacion (catalogo) – carrito (anotaciones)

* **Modelo Espiral:**
  + Enfoque iterativo con análisis de riesgos.
  + Ideal para sistemas complejos con alto riesgo.

**📘 4. Modelo Espiral – "El valiente explorador"**

Nico era un explorador que cruzaba una selva peligrosa. En cada viaje, avanzaba un poco más, pero antes analizaba qué podría salir mal.  
Llevaba mapas, linternas, y planeaba rutas alternativas por si algo fallaba. Así, cada vuelta por la selva lo acercaba más a su destino, evitando errores fatales.

**🎯 Actividad: “Misión en terreno hostil”**

**Consigna:**  
Tenés que desarrollar un software médico con mucha regulación.

1. Dibujá un espiral con 3 ciclos (etapas).
2. En cada uno, escribí qué riesgos podrías evaluar antes de continuar.

RESPUESTA:

Registro – Info personal / Info clínica || Carga de Estudios Médicos (Diagnostico) || Error de diagnostico || Seguridad de datos

=====================================================

**Modelos Ágiles:**

* **Scrum:**
  + Roles: Product Owner, Scrum Master, Dev Team.
  + Conceptos: Sprint, backlog, daily meeting, sprint review.
  + Herramientas: Trello, Jira, ClickUp.

**📘 5. Scrum – "La cocina colaborativa"**

Un equipo quería abrir un restaurante nuevo. Cada semana, se reunían y decidían qué plato agregar al menú.  
Cocinaban, probaban, corregían. El Dueño (Product Owner), el organizador (Scrum Master) y los cocineros (Dev Team) hacían reuniones todos los días para ver avances.  
Y cada viernes, mostraban al público el nuevo plato (Sprint Review).

**🎯 Actividad: “Tu app en mini sprints”**

**Consigna:**

1. Imaginé que sos el equipo de desarrollo de una app para encontrar mascotas perdidas.
2. Dividí el trabajo en 3 sprints con tareas posibles (como mapa, publicación, contacto).
3. ~~Asignen un Scrum Master y hagan una Daily simulada de 2 minutos.~~

RESPUESTA

Sprint 1=> Crear un formulario de publicación de perdida con datos y fotos.

Sprint 2=> Agregar funcionalidad para ver mascotas reportadas zona de desaparición

Sprint 3=> Relacionar con datos personales del dueño si se registra encuentro

* **Kanban:**
  + Flujo visual continuo.
  + No hay iteraciones, se controla el WIP (work in progress).
  + Muy usado en mantenimiento, soporte y mejoras.

**📘 6. Kanban – "El taller organizado"**

En el taller de Sofía, había un gran pizarrón dividido en tres: “Por hacer”, “En proceso”, “Hecho”. (TODO – DOING – DONE)  
Cada tarea era una ficha que se movía a medida que avanzaban.  
No tenían tiempos fijos, pero nada se acumulaba gracias a su control del trabajo en curso.

**🎯 Actividad: “Tablero en vivo”**

**Consigna:**

1. Simulen un equipo de mantenimiento IT.
2. Usen post-its en una pizarra dividida en 3 columnas y coloquen tareas reales.
3. Organicen las tareas priorizando que no haya más de 3 tareas “En proceso”.

CASA

|  |  |  |
| --- | --- | --- |
| “Por hacer” | “En proceso” | “Hecho” |
|  |  | SACAR LA BASURA(Agustin) |
|  |  | COCINAR(Emanuel) |
|  | LAVAR LOS PLATOS(Gustavo) |  |
| BARRER(Esteba) |  |  |

* **XP (Extreme Programming):**
  + Prácticas clave: Programación en parejas, TDD, integración continua.

**📘 7. XP – "La dupla de oro"**

Ana y Luis eran programadores que siempre trabajaban en pareja.  
Mientras uno escribía código, el otro revisaba y testeaba en simultáneo.  
Cada mañana hacían pruebas automáticas antes de subir los cambios.  
Eran rápidos, eficientes y evitaban errores a largo plazo.

**🎯 Actividad: “Código a dos manos”**

**Consigna:**

1. En parejas, uno escribe una función sencilla (ej: convertir Celsius a Fahrenheit) y el otro hace pruebas con distintos datos.
2. Luego cambian roles.
3. ¿Qué errores encontraron al testear en vivo?

num = 4

num1 = 3

num2 = 4

fac = num \* num-1 \* num-2 \* mun-3

* **DevOps:**
  + Integración de desarrollo y operaciones.
  + Automatización, CI/CD, herramientas: Docker, Jenkins, GitHub Actions.

**📘 8. DevOps: El puente entre dos mundos**

🔎 **Cuento: “El puente invisible”**

Había una vez dos reinos separados por un río caudaloso. A un lado vivía el Reino del Desarrollo, donde los ingenieros creaban software, escribían líneas de código y soñaban con sistemas perfectos. Al otro lado del río estaba el Reino de las Operaciones, donde los guardianes de los servidores, las redes y la seguridad se encargaban de que todo funcionara sin caídas.

Durante años, estos dos reinos trabajaban por separado. Cada vez que los desarrolladores terminaban su producto, lo arrojaban al río con la esperanza de que las operaciones lo recogieran y lo pusieran a funcionar. Pero muchas veces el software se hundía, o llegaba dañado. Las peleas eran constantes: “¡Funciona en mi máquina!” gritaban desde el lado del desarrollo. “¡Pero no en producción!” respondían desde operaciones.

Un día, surgió una idea nueva: construir un puente. Pero no cualquier puente, sino uno que se mantuviera solo, que se reparara al instante y que conectara a ambos reinos sin errores. Así nació **DevOps**, una cultura y conjunto de prácticas que unió desarrollo y operaciones, automatizando todo lo posible, asegurando calidad desde el inicio y eliminando la eterna lucha entre ambos mundos.

Gracias al puente DevOps, los despliegues eran continuos, las fallas se detectaban antes de llegar a producción y ambos equipos trabajaban como uno solo, con herramientas compartidas como **Docker**, **Jenkins**, **GitHub Actions** y muchas otras.

📚 **Actividad: "Construyendo el puente DevOps"**

📝 **Consigna:**

Imaginá que sos parte de un equipo que trabaja en una app de pedidos de comida. El equipo de desarrollo ya tiene todo el código, pero hay constantes errores cuando se pasa a producción: la app se cae, los pedidos no llegan, y los usuarios se quejan.

1. Explicá cómo se podrían aplicar los principios de DevOps para resolver este problema.
2. Indicá qué herramientas automatizadas usarías en cada etapa: desarrollo, testeo, integración, despliegue.
3. Dibujá (o describí) cómo sería tu “puente DevOps” en este proyecto: ¿quiénes lo cruzan?, ¿cómo?, ¿qué se automatiza?

RESPUESTA

Integrar los equipos Dev – Ops => Comunicación – Responsabilidades => Automatización – Feedback claro para sector

Xp1 – se encarga de gps

Xp2 – se encarga de registros

Xp3 – pagos

Xp4 – deploy

Xp5 – matenimiento

Xp6 – At. Cliente => repote para Xp1(Tello o github project) –

A xp1 recibe lo subsana -> Xp4 ->producción

**✍️ Actividad**

* El aula se divide en "estaciones" por modelo.
* Cada grupo rota por las estaciones y:
  + Lee una descripción.
  + Analiza un caso (párrafo breve).
  + Responde: ¿Este modelo sería apropiado? Justificán.
* Cierre: en ronda, cada grupo explica un modelo desde su experiencia.

**⚖️ 3. Actividad integradora**

**Consigna:**

* Formar equipos.
* Elegir una app conocida (Spotify, WhatsApp, TikTok, MercadoLibre).
* Identificar el modelo de desarrollo que mejor se ajusta y justificarlo.
* Elaborar una presentación breve (5 diapositivas o pizarra).

**Recursos:**

* Miro / Google Slides / Canva.
* Ejemplos prearmados con plantillas.
* Apoyo del docente para guiar en la elección del modelo.

**💬 4. Puesta en común y discusión** Cada grupo expone su caso.

* Debate cruzado: ¿Coinciden con la elección de sus compañeros?
* Pregunta final: ¿Creen que los modelos se pueden combinar?

**🧠 5. Cierre y tarea**

* Repaso de lo visto: modelos secuenciales vs iterativos vs ágiles.
* Importancia del contexto y del equipo.
* Tarea: simular una "daily meeting" con sus compañeros en grupo de WhatsApp o Discord y registrar la conversación.

**📚 Recursos recomendados**

* **Libro**: Sommerville, I. "Ingeniería del Software" (cap. 4 y 17).
* **Libro**: Pressman, R. S. "Ingeniería del Software" (cap. 3 y 5).
* **Web**: scrum.org, atlassian.com/agile/kanban, devops.com

**🎥 Videos:**

* Scrum en 7 minutos: https://www.youtube.com/watch?v=9TycLR0TqFA
* Kanban explicado: https://www.youtube.com/watch?v=iVaFVa7HYj4
* Waterfall vs Agile: https://www.youtube.com/watch?v=3fKjKQkI1pU