Tema 2: Prácticas Facilitadoras del Desarrollo Ágil de Software.

- **Programación en parejas:** es la practica por la que todo el código desarrollado es escrito por dos desarrolladores sentados frente a una única maquina de trabajo. Un miembro es el "conductor" que es el que controla el ratón y teclado, y el otro es el "observador" que controla los defectos y va pensado en alternativa y pruebas. Ambos roles se van intercambiando, y deben tener niveles de experiencia similares. Y debe haber respeto y no hacer constantes correcciones.
  - Se finaliza más rápido, se corrigen los errores mas rápidos.
  - Más personas conocen el funcionamiento del código y han realizado pruebas.
  - Los diseños son de mayor calidad.
  - Posibles pasos:
    - Preparación: Definir como se va ha llevar a cabo.
    - Trabajo individual.
    - Cierre del trabajo en pareja: Poner el código funcional y explicar como se lo hemos hecho, y ejecutar las pruebas correspondientes, para comprobar las funcionalidades. Ver si sigue la normativa.
- **Propiedad colectiva de Código:** Un mismo código que puede ser modificado por varios programadores simultáneamente.
  - Todos deben seguir el mismo Estándar de Código.
  - El código que hay integrado ha sido probado y se almacenan las pruebas.
  - Cuenta con un adecuado control de versiones.
  - Funcionamiento:
    - Se descarga el código a modificar del sistema de control.
    - Se descargan las pruebas necesarias para ese código.
    - Desarrollamos la nueva funcionalidad y realizamos las pruebas necesarias.
    - Si el código pasa las pruebas y cumple el estándar de codificación, lo subimos al sistema de control.
  - Facilita la difusión del conocimiento.
  - El código sigue estándares y tiene mayor calidad, y este mecanismo permite detectar errores y corregirlos.
  - El objetivo no es corregir el código de otros sin propósito o cuestionarlo.
  - No se debe modificar el mismo código simultáneamente varias personas.
- **Normativas de Código:** Conjunto de reglas y recomendaciones sobre como se debe escribir, estructurar y definir el código, para que en un futuro se pueda escribir sobre el y saber que y como lo hace. Sin gastar tiempo en un futuro en reescribir el códigos.
  - Ayuda a construir programas correctos, entendibles y fáciles de mantener.
  - Las normas deben cumplirse obligatoriamente y las recomendaciones se aplican siempre en general, pero puede haber excepciones.
  - **Aspectos básicos** a tener en cuenta en una normativa:
    - Nombre y Codificación de Ficheros:
    - Organización de Ficheros: La aparición de la leyenda de derechos de autor,...
    - Comentarios.
    - Secuencias.
    - Reglas para asignar nombre: Determinado regla de mayúsculas o minúsculas para determinado tipo de variable, archivo o parte del codigo. Camel, Pascal, solo mayúsculas o minúsculas... Nombres en ingles, cortos pero representativos.
  - **Aspectos avanzados:**
    - Gestión de errores y excepciones: Si hay un error sacar una excepción, dar información auxiliar sobre el error y que aparezcan en la aplicación principal.

        - Seguridad.
        - Patrones de diseño.
        - Rendimiento.
        - Globalización.
        - Fiabilidad de mantenimiento.
        - Otras buenas practicas.

- **Integración continua y automatizada:** Cada vez que se genera una nueva función o porción de código se integra con el código que ya se había generado y probado anteriormente. Se construye incrementalmente la funcionalidad, en vez de hacer todo por separado y juntarlo mas tarde. Y cada vez que se integre una parte se debe probar la totalidad.
    - Antes de introducir nuevas funcionalidades se debe comprobar que funcione correctamente el código previo, por lo que hay que dar a conocer el estado de la integración.
    - No se harán nuevas integraciones en poco tiempo, excepto si las pruebas se pueden hacer rápido o se codifica en parejas.
    - **Características**:
        - Localización centralizada del código fuente.
        - Un único comando para compilar y enlazar los ejecutables.
        - Soporte para automatizar las pruebas.
        - Todos pueden acceder a un ejecutable confiable del sistema.
    - **Beneficios**:
        - Se reducen los riesgos técnicos.
        - Se reduce el pesado proceso de integrar todo en un solo momento, de esta manera se empieza a integrar desde el inicio.
        - Los errores se resuelven en el primer momento que se producen.
    - **Desventajas**:
        - El coste de poner apunto todo el sistema, configurar , mantener e integrar la plataforma.
        - Mantenimiento de los scripts de configuración.
        - Dificultad de incluir código preexistente.
    - Hay herramientas que facilitan esta tarea como Jenkins, Bamboo, Cascade...
    - **Cuando se esta llevando a cabo una integración correcta?**
        - La versión mas actual esta en el repositorio, nadie tiene una visión mas actualizada.
        - La versión del repositorio supera todas las pruebas sin errores, y estas están registradas para que cualquiera las pueda ejecutar.
        - Todo el mundo conoce el estado del código, para sabe si pueden trabajar con el o no.
        - Ejecutable en repositorio de código.
        - El proceso está automatizado y no necesitará intervención humana.