**ENGINYERIA DE PROGRAMARI**



Alumne: David Peñalver Navarro

Professor: Joan Puigcerver Ibáñez

Assignatura: Entorns de desenvolupament

Año académico: 2023/2024

Índice

[Exercici 1: implantació vs implementació 3](#_Toc147225501)

[Exercici 2: Scrum 4](#_Toc147225502)

[Exercici 3: TDD 5](#_Toc147225503)

[Exercici 4: CI/CD 6](#_Toc147225504)

[Referencias 7](#_Toc147225505)

# Exercici 1: implantació vs implementació

**En les fases del cicle de vida del software existeixen dos fases amb un nom similar: implantació i implementació. Identifica de quines fases es tracta, i indica on es troben en el cicle de vida del programari.**

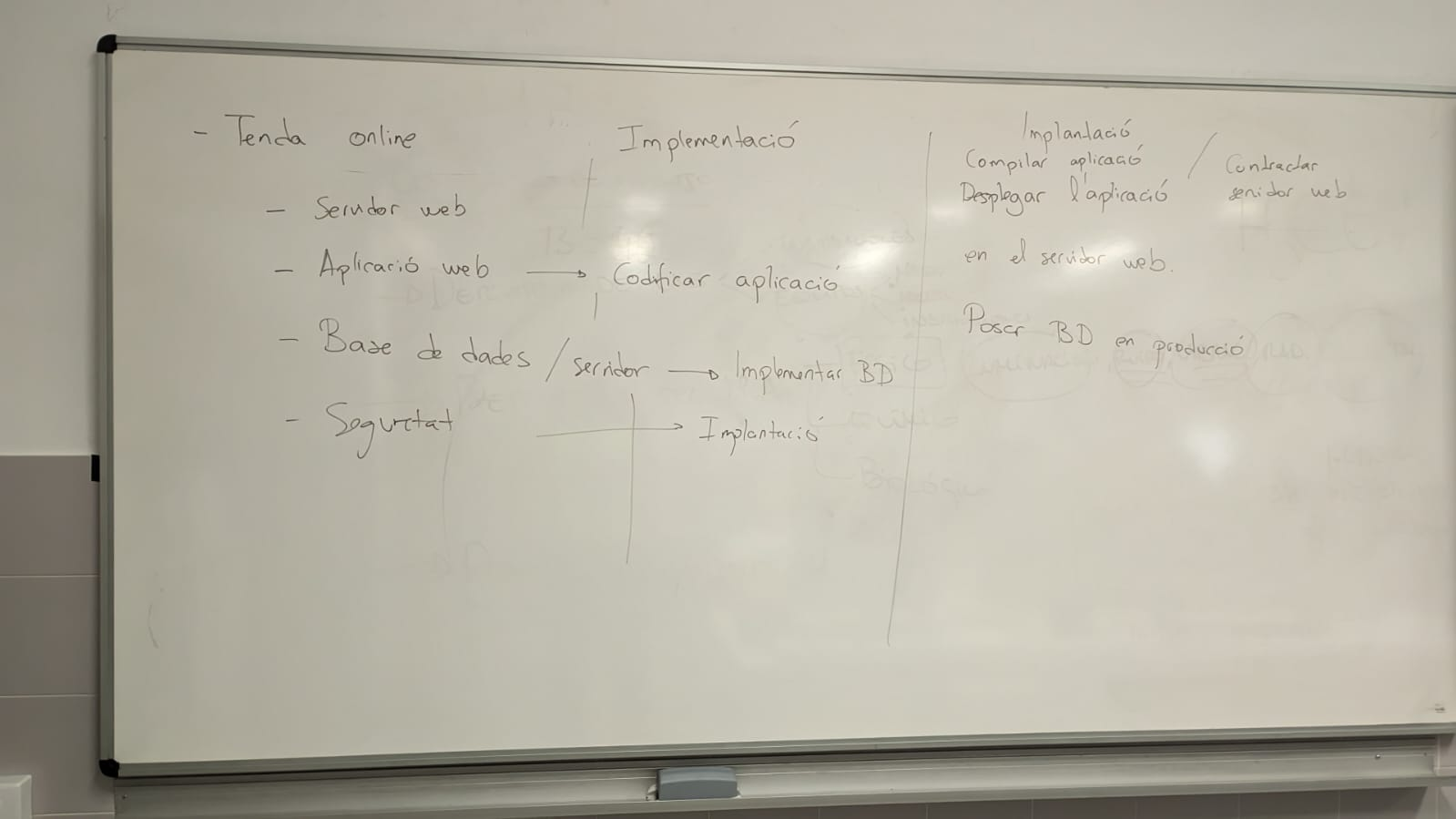
**Pensa un exemple de desenvolupament de programari i identifica en què consistirien aquestes fases.**

La fase d’implementació o de codificació és la fase on es du a terme la realització del programa atenent als seus components (bases de dades, servidors i comunicacions).

Per contra, la fase d’implementació o desplegament consistix en publicar la solució final o lliurar el codi.

En quant al cicle de vida del programari, la fase d’implementació es la tercera. La fase d’implementació es troba en la sexta de les set fases.

Ficant un exemple de desenvolupament, podríem imaginar que estem en l’empresa X de Elon Musk. Ací, els càrrecs, cap de projecte i similars analitzarien què implementar, com per exemple l’opció de poder manar missatges de veu. Posteriorment, en la fase de disseny es faria tota l’estructura i comportament previst. Ja a continuació, entraria la fase d’implementació, on es produiria el codi que faria la funció de poner manar missatges de veu. A partir d’ací es farien les proves, la documentació i, a continuació, arribariem a la d’implantació, on publicaríem la nova funcionalitat de X (compilem i despleguem, BD en producció). Finalmente, es mantindria el correcte funcionament d’esta.



# Exercici 2: Scrum

**Busca i explica en què consiteix la metodologia SCRUM.**

**Intenta buscar opinions a favor i en contra d'utilitzar aquesta metodologia.**

Scrum és un procés o metodologia àgil on s'apliquen de manera regular un conjunt de bones pràctiques per a treballar col·laborativament, en equip, i obtindre el millor resultat possible d'un projecte. Estes pràctiques es recolzen les unes a les altres i la seua selecció té origen en un estudi de la manera de treballar d'equips altament productius. Estes pràctiques s’organitzen en esprints d’1-3 setmanes en els quals el treball realitzat es entregable. Dins d’això, es planifiquen les tasques de l’esprint i es fa una revisió diària també.

En quant a punts a favor, trobem que millora el foc; també la comunicació I transparència; el treball en equip, delimita les responsabilitats de cada membre; també compromet als equips treballant com una unitat. Un altre funt a favor, proporciona una major implicació del client I el seu feedback. Pel que respecta a les tarees, les prioritza y les traza per a comprovar quines estan en cada fase.

Pel que respecta a punts en contra, n’hi ha que conèixer previement com es treballa amb esta metodologia. Així mateix, les tasques i les fases sempre tenen que estar definits. Tampoc es poden deixar activitats sense acabar. Per últim, en quant a la dinàmica de l’empresa, es té que aplicar en equips reduïts, la qual cosa pot complicar el treball en grans empreses i, a més a més, es complicat gestionar els canvis dins de l’empresa.

Situacions

* Soluciones rápidas.
* Entornos complejos
* Situaciones flexibles.

Beneficis

* Flexibilidad y adaptación.
* Resultados anticipados.
* Productividad y calidad.
* Equipos motivados.

Roles

* Producto owner (representa al cliente): determina la visión del producto y las prioridades
* Scrum master (facilitador del equipo): potencia su productividad y supervisa
* Team members (equipo de desarrollo): entregar el proyecto, equipo auto organizado de 5-9 personas

Esprint

* 2-4 semanas
* Contenido fijo
* Reunión de planificación
  + Cliente presenta requisitos ordenados por prioridad (producto backlog)
  + Sprint backlog: el equipo selecciona los de mayor prioridad y elabora una lista de tareas
* Reunión diaria
  + Reunión de sincronización de 15’
  + Cada miembro responde a:
    - ¿qué he hecho desde la última reunión?
    - ¿qué voy a hacer a partir de este momento?
    - ¿qué impedimentos tengo o voy a tener?
  + Scrum master supervisa que se cumpla todo y no baje la productividad
* Reunión de revisión
  + Último día del esprint
  + Se entregan los requisitos completados al cliente
  + El cliente realiza las adaptaciones necesarias replanificando el proyecto
* Reunión retrospectiva
  + El equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que se podrían encontrar mejorando de manera continua la productividad

A screen shot of a computer

Description automatically generated

# Exercici 3: TDD

**Busca i explica en què consiteix el TDD o Test-Driven Development.**

EL TDD és una pràctica de programació que consistix en escriure primer les proves, després el codi font i, per últim, refactoritzar el codi escrit.

Els grans avantatges d’esta metodologia són que amb esta pràctica s'aconsegueix, entre altres coses, un codi més robust, més segur, més mantenible i més rapidesa en el desenvolupament.

El procés de disseny de programari, combinant TDD amb metodologies àgils, seria el següent:

1. El client escriu la història d'usuari.
2. S'escriuen juntament amb el client els criteris d'acceptació d’esta història, desglossant-los molt per simplificar-los tant com siga possible.
3. S’elegix el criteri més simple d'acceptació i es traduix en una prova unitària.
4. Es comprova que esta prova falla.
5. S'escriu el codi que fa passar la prova.
6. S'executen totes les proves automatitzades.
7. Es refactoritza i es neteja el codi.
8. Es tornen a passar totes les proves automatitzades per a comprovar que tot seguix funcionant.
9. Tornem al punt 3 amb els criteris d'acceptació que falten i repetim el cicle una vegada i una altra fins completar la nostra aplicació.

finales de los 90 de principios de los 2000 pero en españa pues ha llegado un poco más tarde

no es no es un lenguaje de programación, sino que es es más bien una técnica. es una práctica de ingeniería de software que involucra pues otras dos prácticas cuales han cogido dos prácticas que son muy útiles y se han se han juntado para construir una técnica las dos prácticas son la de escribir pruebas primero antes que el código escribimos las pruebas y la segunda técnica es la de refactorizar el propósito de este esta técnica es y centrarse en tres objetivos básicos

el primer objetivo es minimizar el número de bugs porque por qué al final cuando haces una estimación del coste de tu aplicación pues llegas hasta el desarrollo y alomejor porsche incluso al estudiante le das un tiempo de garantía al final cuantos más books salgan pues menos rentable te ha salido te ha salido el proyecto corregir los books es tiempo que te quitas de hacer otras tareas entonces si no tiene los books vas a conseguir mayor rentabilidad de turno de tu código de tu aplicación después sin implementar las soluciones justas que el cliente necesita más es muy común que cuando nos dan los requisitos de una aplicación las especificaciones en las fases de análisis y diseño pues diseñamos una aplicación y pues el diseñador por su experiencia pues completa con funcionalidades pensando que van a ser útiles para el pri aunque van a ser útiles para la aplicación van a ser útiles para otros componentes porque por su experiencia no se sabe pues que puede ser útil y que era muy común usar o no ese es un trabajo extra que que luego en el 90 al 95 99 de los casos esas funcionalidades extra no las usa ninguna parte de la aplicación entonces es tiempo extra hemos invertido en desarrollando la aplicación que no que no haya llegado a nada entonces el objetivo de que debes eliminar es ese código innecesario esas funcionalidades que no ha pedido el cliente con lo cual redonda en en eficiencia en el trabajo y en rentabilidad definitivamente

y la tercera es producir software modular que sea altamente reutilizable que esté preparado para el cambio esto realmente lo que sigue con cualquier técnica por qué con buenos hábitos de programación siempre vas a hacer que tu software sea modular y reutilizable de llevarte a los otros proyectos los componentes son un deseo y preparado para el cambio por su característica que no se consigue siempre hay que contener e s con conté de bercy que se consigue muchas veces cuando tienes que cambiar alguna funcionalidad de la aplicación pues para actualizarte porque toca el código que ya está hecho código que puede ser muy complicado o que hayan hecho otras personas e incluso que ya no estén en la empresa entonces con td tienes la confianza de que cuando haces cambios no vas a estropear las funcionalidades que ya tienes y esto no se consigue porque no te debe la forma funcional desde el primero se construye

el primero se construyen las pruebas y luego el código hace que todo el código surgido este testeado desde antes de existir incluso con lo cual cualquier cambio que vayas a introducir está cubierto por cortes y si metes la pata algún test cuando ejecute es la batería de la cántara entonces ahora la confianza y prepara tu software para cualquier cualquier cambio porque la modificación y bueno pues esto se consigue se debe pues cambiando un poco la mentalidad tradicional y primero hay que analizarlo los requisitos luego hacer un diseño completo y profesional y luego empezar a modificar y lo último es testear o contener

hay que cambiar la mentalidad hay que dejar de pensar en tareas tengo que hacer que la aplicación alan tal cosa tal tarea pues que de alta acusación son tareas al final de cabo y contiene de lo que pensamos son ejemplos concretos de ejemplos concretos con datos concretos puedes si me tienen tal caja el resultado va a salir en 500 referidos muy concretos ya no es la tarea de lo que haces sino se basa en lo que son los tres parámetros de entrada parámetro de salida y vio si la respuesta es la que yo esperaba si consigues tener concede consigues tener en sus especificaciones en tus requisitos una lista de ejemplos muy completa que elimine cualquier tipo de ambigüedad y que sean concretos que se puedan transformar en tés pues al final vas a tener una batería de test que te cubren todas las funcionalidades y un código resultante final que está 100% cubierto de por es por estos textos

# Exercici 4: CI/CD

**Busca i explica en què consiteix el CI/CD o Continuous Integration/Countinuous Deployment.**

Continuous Integration (CI) i Continuous Deployment (CD) són pràctiques de desenvolupament de programari que busquen millorar l'eficiència, qualitat i velocitat de lliurament del programari automatitzant i integrant continuament els canvis en el codi.

Continuous Integration (CI):

* Consistix en:
  + Integrar i provar automàticament els canvis al codi font tan prompte com són realitzats pels desenvolupadors.
  + Executar automàticament una sèrie de proves (unitàries, dintegració, etc.) per assegurar-se que els canvis no introduixen errors.
  + Proporcionar retroalimentació immediata als desenvolupadors sobre la qualitat del codi.
* Objectius:
  + Detectar i corregir problemes d'integració prompte en el cicle de desenvolupament.
  + Millorar la qualitat del codi.
  + Accelerar el cicle de desenvolupament en reduir el temps entre l'escriptura del codi i la detecció d’errors.

Continuous Deployment (CD):

* Consistix en:
  + Automatitzar el procés d’implementació i lliurament de programari a entorns de prova i producció.
  + Desplegar automàticament les versions del programari que han passat amb èxit les proves a entorns de producció.
* Objectius:
  + Accelerar el lliurament de programari en eliminar barreres manuals en el procés d’implementació.
  + Garantir un procés d’implementació consistent i fiable.
  + Permetre el lliurament freqüent i continu de noves funcionalitats i millores.

# Referencias

https://joapuiib.github.io/itb/DAM-ED/UD1/exercicis/03\_enginyeria.html

https://proyectosagiles.org/que-es-scrum/

https://www.iebschool.com/blog/ventajas-sprints-agile-scrum/

https://es.indeed.com/orientacion-laboral/desarrollo-profesional/ventajas-desventajas-metodologia-scrum

https://www.paradigmadigital.com/dev/tdd-como-metodologia-de-diseno-de-software/