

Actividad práctica de Lean Development

Introducción

Sois los integrantes de un equipo de trabajo que tiene que crear una aplicación para un supermercado siguiendo el modelo de desarrollo de software conocido como Lean Development. Como ya sabréis, esta metodología se basa en aplicar 7 principios fundamentales:

- 1.- Eliminar los desperdicios
- 2.- Amplificar el aprendizaje
- 3.- Decidir lo más tarde posible
- 4.- Entregar lo más rápido posible
- 5.- Capacitar al equipo
- 6.- Construir integridad intrínseca
- 7.- Ver todo el conjunto

Se os plantean una serie de decisiones que deberéis tomar para desarrollar la aplicación de la empresa. En vuestro grupo de trabajo tenéis un experto en cada uno de los 7 principios del desarrollo de software Lean y **cada una de las decisiones que toméis deberá estar justificada en base uno o más de los 7 principios**. Si dos principios entran en conflicto, explicadlo y proponed una solución alternativa. Por cada respuesta correcta, el equipo sumará 5 puntos. Si la respuesta correcta viene justificada por medio del principio adecuado se añadirán otros 10 puntos más al equipo. Al final del recuento, el equipo que tenga más puntuación recibirá 30 puntos extra. Vuestro objetivo como equipo es demostrar que sois los que más sabéis de Lean Development obteniendo el máximo número de puntos posibles para ganar aquello por lo que todos estamos aquí, el título del Doble Grado de Ingeniería Informática y Matemáticas, y los 6 créditos de descuento.

Elaboración del proyecto

El supermercado quiere una aplicación para dispositivos móviles que permita realizar una compra online en uno de los supermercados de la franquicia. También se pretende que la aplicación ofrezca un servicio de compra a domicilio desde la propia app. La empresa divaga sobre la posibilidad de implementar otras funcionalidades como poder reservar plaza en el parking para días de mucha clientela y está indecisa sobre ampliar la aplicación apta a otros soportes.

Vuestra primera responsabilidad es elaborar una **lista de requisitos** para la aplicación (no olvidéis la **justificación de vuestras decisiones**):

(Requisitos: aplicación para móviles, funcionalidad de comprar a domicilio.)

No requisitos: reservar plaza en el parking (desperdicio, requisito poco claro), otros soportes (incertidumbre, decidir lo más tarde posible)

Una vez establecidas los requisitos tenéis una reunión con vuestros jefes y os plantean distintos **sistemas de trabajo** de entre los cuales debéis **elegir uno**:

- a) Establecer una jerarquización en la que ellos organizarán las tareas y marcarán el trabajo a seguir por vuestro equipo. (No sigue el principio de capacitar al equipo, decir a los trabajadores cómo hacer su trabajo)
- b) Libertad absoluta de vuestro equipo para tomar cualquier tipo de decisión sobre el proyecto sin necesidad de ser consultada. (No es la filosofía, los administradores también tienen un papel, aunque es más secundario)
- c) Establecer un sistema de comunicación entre miembros del equipo que tenga que tener como intermediario el inmediato superior para garantizar la coordinación entre los miembros. (Desperdicios, comunicación interna lenta)
- d) Regirse de manera autónoma mientras los administradores se limitarán explicar el qué se debe hacer y no el cómo. (Capacitar al equipo)

En la reunión también debéis acordar como se van a dividir las tareas. Hay dos posturas enfrentadas. Por un lado, hay quien pretende que los trabajadores se especialicen únicamente en uno de los requisitos de la aplicación y que ignoren el resto. Otros, sin embargo, apuestan por un sistema en el que la interacción entre los distintos equipos dentro del proyecto sea fundamental. Escoged la opción más acorde con la filosofía Lean.

(Ver todo el conjunto, pensar en grande y actuar en pequeño)

Antes de comenzar a trabajar, la empresa se pone en contacto con vuestro equipo para entregaros un dossier con el trabajo de otro equipo que fracasó a la hora de satisfacer sus necesidades. Entre la montaña de documentación que os entregan, encontráis un calendario de reuniones con la empresa de supermercados, que muestra que estas se realizaban con poca frecuencia, y un organigrama de distribución de tareas, que refleja que toda la comunicación entre el cliente y los desarrolladores se realizaba a través del manager del equipo. También habéis podido encontrar un libro de Tom y Mary Poppendieck y una lista con funcionalidades añadidas que la empresa de supermercados no había planteado.

Vuestra misión es **aprender de los errores** del equipo que lo había intentado anteriormente y también de sus **aciertos**. ¿Cómo interpretáis la información que habéis obtenido?




(Problemas: funcionalidades añadidas (desperdicio, no añaden valor al cliente), fechas de entrega poco frecuentes (no se cumple Entregar tan rápido como sea posible ni amplificar el aprendizaje), comunicación lenta (desperdicio y no capacitar al equipo).
Cosas de donde aprender: Libro de Tom y Mary Poppendieck, escribieron *Lean software development: an agile toolkit* y fueron los precursores de la metodología lean en el software.)

Una vez leída toda la documentación y analizada, se os plantea la duda de si vosotros debéis **incluir documentación** en vuestro proyecto.





(No, el flujo de información entre lo que debe construir y la construcción debe ser al mismo tiempo, no secuencialmente. No es válida la documentación, mejor la comunicación cara a cara. Relación con Construir integridad intrínseca)

Ya habéis comenzado a trabajar y debéis decidir la estructura del menú para vuestra aplicación. Habéis llegado a 2 diseños:

a)

Hacer pedido 	Envío de compra a domicilio 
Ayuda 	“Insertar funcionalidad”

b)

• Hacer pedido	
• Pago en app	
• Ofertas y descuentos	€
• Datos de contacto	
• Mapa de supermercados	

¿Cuál decidís escoger y por qué?

(a) ya que se aplica el principio de decidir lo más tarde posible y b) tiene desperdicios como funcionalidades no demandadas por el cliente. Nos ahorraremos código y funcionalidades innecesarias eliminar desperdicios)

Tras escoger el diseño, programáis el menú. Si habéis escogido la opción b), ¿la presentaríais al cliente? Si habéis escogido la opción a), ¿organizaríais una reunión con la empresa de supermercados o esperaríais a que os sugiriera una nueva funcionalidad?

(En cualquiera de los 2 casos lo que más se ajusta con el lean development es entregar el trabajo cuanto antes sea posible, con ello se obtiene feedback del cliente y en la próxima iteración es posible que ya se haya tomado una decisión de si incluir una nueva funcionalidad. La entrega lo más rápido posible y decidir lo más tarde posible)

Una vez más estáis ante una disyuntiva. El proceso lean software development es un modelo ágil e iterativo, como scrum, pero estas **iteraciones**, ¿han de ser **cortas o largas**?

(Claramente cortas: feedback, aprendizaje, presentar dudas al cliente o proponerle sugerencias)

Continuáis avanzando en el proyecto manteniendo reuniones sucesivas con el cliente, pero vuestro experto en la eliminación de desperdicios cree que hay muchas cosas en el proyecto que no añaden valor al cliente. Propone realizar un “**mapa de flujo de valor**”, ¿en qué consiste y qué beneficios puede aportar?

(Se utiliza una técnica llamada *value stream mapping* (o mapa de flujo de valor) para distinguir y reconocer los desperdicios. Primero se dibuja el mapa de flujo de valor tal como está el proceso, mostrando cada una de las etapas, las esperas y la información que se requieren para entregar el producto o servicio. El segundo paso consiste en señalar las fuentes de los desperdicios y eliminarlos. Estos suelen concentrarse en tiempos de espera, exceso de procesado y defectos del programa. Finalmente se dibuja un mapa del estado futuro, una vez eliminados los desperdicios. Lo mismo debe hacerse iterativamente hasta que incluso los procesos y procedimientos que parecían esenciales sean eliminados.)

Habéis conseguido entregar el proyecto y la empresa de supermercados está muy contenta, ¿pero habréis seguido al pie de la letra el desarrollo de software lean?