CURSO DE PROGRAMACIÓN FULL STACK

PROYECTO FINAL

PROGRAMACIÓN FULLSTACK

PROYECTO FINAL

Nombre del Proyecto:

Integrantes:

Líder del Proyecto. (Elijan a un referente del equipo, la persona que va a organizar el proyecto y va a ser el nexo entre los tutores y el equipo)

Breve descripción del proyecto. Definir en no mas de 20 palabras de que se trata el sistema a realizar

División del equipo

Acá vamos a dividir nuestro equipo en dos partes, la parte que se va a encargar del backend de la aplicación y la parte que se va a encargar del frontend de la aplicación. La parte de back se encargará de todo lo que sea código de Java y funcionalidades de la aplicación, mientras que la parte de front se encargará del diseño de la pagina y como se van a representar dichas funcionalidades en la pagina

La cantidad de participantes para cada parte, la deciden entre los integrantes.
La idea es que las dos partes se mantegan comunicados, ya que no puede ser uno sin el otro, pero es útil que las partes de back se comuniquen entre ellos y las partes de front entre ellos.

CICLO DE VIDA DE UN PROYECTO DE SOFTWARE

El ciclo de vida de un proyecto hace referencia a las etapas por las cuales pasa el proyecto desde su concepción hasta su implementación y puesta en marcha final.
A continuación, se muestran las etapas por las cuales va a ir pasando nuestro proyecto.
El ciclo de vida básico de un software consta de los siguientes procedimientos:

1- **Definición de objetivos:** define la finalidad del proyecto.
2- **Análisis de los requisitos y su viabilidad:** recopila, examina y formula los requisitos del cliente y examina cualquier restricción que se pueda aplicar.
3- **Diseño general:** requisitos generales de la arquitectura de la aplicación
4- **Diseño en detalle:** definición precisa de cada subconjunto de la aplicación.
5- **Programación (programación e implementación):** implementación de un lenguaje para crear las funciones definidas durante la etapa de diseño
6- **Prueba de unidad:** prueba individual de cada subconjunto de la aplicación para garantizar que se implementaron de acuerdo con las especificaciones.
7- **Integración:** garantiza que los diferentes módulos se integren con la aplicación. Este es el propósito de la prueba de integración que está cuidadosamente documentada.
8- **Prueba beta (o validación):** garantiza que el software cumple con las especificaciones originales.
9- **Documentación:** sirve para documentar información necesaria para los usuarios del software y para desarrollos futuros.
10- **Implementación**
11- **Mantenimiento:** comprende todos los procedimientos correctivos (mantenimiento correctivo) y las actualizaciones secundarias del software (mantenimiento continuo).

El orden y la presencia de cada uno de estos procedimientos en el ciclo de vida de una aplicación dependen del tipo de modelo de ciclo de vida acordado entre el cliente y el equipo de desarrolladores.

Durante la ejecución del proyecto final vamos a atravesar por varias de estas etapas, para poder llevar a cabo la mejor solución para nuestro problema a resolver.

ETAPA 1: DEFINICIÓN DEL OBJETIVO GENERAL Y LOS OBJETIVOS ESPECÍFICOS DEL PROYECTO

Identificar los objetivos del proyecto

El primer paso para definir los objetivos del proyecto es identificar qué aspectos del proyecto son realmente importantes; y aquí debemos separar dos ámbitos:

- Los objetivos del proyecto a nivel del producto que este debe entregar a la organización, cliente, u otros involucrados.
- Aquellos aspectos internos del proyecto que son más importantes, y por tanto van a requerir mayor atención, para la consecución de los objetivos.

Para el primer punto deberemos empezar identificando y priorizando los objetivos a través de la información que nos den los involucrados (cliente, sponsor, usuarios, etc.), y de los documentos contractuales que puedan dar origen al proyecto (pedido, oferta, etc.). El tema de priorizar es muy importante, ya que, aunque nuestro objetivo será cumplir con todo, la realidad es que nos podemos encontrar con situaciones que nos impidan poder hacerlo, por lo que debemos ser capaces de saber cuáles objetivos pueden ser más flexibles.

Por otro lado, una vez hayamos planificado el proyecto, veremos que no todas las tareas o recursos involucrados van a tener el mismo efecto a la hora de conseguir los objetivos. Por lo que nuestra atención se deberá centrar en aquellos que sean más relevantes. Esto se entiende bien cuando planificamos con Camino Crítico, ya que las tareas dentro de este camino son las que marcan la duración total del proyecto, por lo que pasan a ser más relevantes que el resto.

Definir correctamente los objetivos del proyecto

"No se puede controlar lo que no se puede medir", por tanto, para controlar si estamos consiguiendo o no los objetivos del proyecto, debemos definir estos de forma que sean medibles.

Objetivo general:

Los objetivos generales son aquellos que se formulan para completar una meta o señalar el fin de un proyecto o investigación.

El objetivo general en el caso de un proyecto siempre va relacionado con el título del mismo, es decir, es el mismo título con la diferencia que se encuentra redactado en infinitivo.

Es por ello que el objetivo general es el paso más sencillo de formular al momento de realizar una investigación a diferencia del planteamiento del problema y los marcos legales, metodológicos y conceptuales.

Incluso es más sencillo de redactar que los objetivos específicos.

Al formular el objetivo general es conveniente tomar en cuenta que debe ser:

- Cualitativo: Debe indicar calidad.
- Integral: Debe ser capaz de integrar los objetivos específicos y para ello debe estar encabezado por un verbo de mayor nivel.
- Terminal: Debe definir un plazo.

Objetivos específicos:

Se trata de objetivos que provienen del objetivo general, son objetivos redactados con un verbo de menor nivel al verbo en el que fue redactado el objetivo general del que provienen.
Los objetivos específicos son una serie de fines que llevan a cumplir el objetivo general.

Los objetivos específicos deben tener:

- Claridad: El lenguaje en el que se encuentran redactados debe ser claro y preciso.
- Factibilidad: Los objetivos deben poder ser cumplidos a través de la metodología seleccionada.
- Pertinencia: Deben relacionarse lógicamente con el problema a solucionar.
  Diferencia entre objetivo general y específico

- Un objetivo general precisa la finalidad de un estudio, investigación, organización o empresa.

- Los objetivos generales son actividades a cumplir para llevar a cabo el objetivo general.

En nuestro caso vamos a pensar que nuestro sistema va a quedar completamente operativo e instalado, entonces por ejemplo para el tinder de mascotas un objetivo sería:

Tener 1500 usuarios de la aplicación al finalizar diciembre del 2020.

Y un objetivo específico sería:

Invertir 100 pesos en campañas de Marketing para fidelizar la aplicación.

ETAPA 2: ANÁLISIS DE REQUISITOS Y DISEÑO

En esta etapa vamos a "entrevistar" al ideador del proyecto para poder determinar qué es lo que debe hacer el Sistema, una vez relevados los datos debemos validar con el ideador las funcionalidades detectadas. Deberemos preguntar y anotar todas las funcionalidades del proyecto, recordemos que mientras más especificas sean las funcionalidades mejor.

La toma de requerimientos es de las etapas mas importantes, ya que todo nuestro sistema se fundamentará en el relevamiento que hagamos en esta etapa.

Una vez definida las funcionalidades, estas se documentan mediante Casos de Uso. ¿Qué son los casos de uso?

Casos de Uso:

En ingeniería del software, un caso de uso es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico. Normalmente, en los casos de usos se evita el empleo de jergas técnicas, prefiriendo en su lugar un lenguaje más cercano al usuario final. En ocasiones, se utiliza a usuarios sin experiencia junto a los analistas para el desarrollo de casos de uso.

En otras palabras, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.
Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas.

O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. Una relación es una conexión entre los elementos del modelo, por ejemplo, la especialización y la generalización son relaciones. Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar cómo reacciona a eventos que se producen en su ámbito o en él mismo

Características

Los casos de uso evitan típicamente la jerga técnica, prefiriendo la lengua del usuario final o del experto del campo del saber al que se va a aplicar. Los casos del uso son a menudo elaborados en colaboración por los analistas de requerimientos y los clientes.

Cada caso de uso se centra en describir cómo alcanzar una única meta o tarea de negocio. Desde una perspectiva tradicional de la ingeniería de software, un caso de uso describe una característica del sistema. Para la mayoría de proyectos de software, esto significa que quizás a veces es necesario especificar diez o centenares de casos de uso para definir completamente el nuevo sistema. El grado de la formalidad de un proyecto particular del software y de la etapa del proyecto influenciará el nivel del detalle requerido en cada caso de uso.

Los casos de uso pretenden ser herramientas simples para describir el comportamiento del software o de los sistemas. Un caso de uso contiene una descripción textual de todas las maneras que los actores previstos podrían trabajar con el software o el sistema. Los casos de uso no describen ninguna funcionalidad interna (oculta al exterior) del sistema, ni explican cómo se implementará. Simplemente muestran los pasos que el actor sigue para realizar una tarea.

Un caso de uso debe:

- describir una tarea del negocio que sirva a una meta de negocio
- tener un nivel apropiado del detalle
- ser bastante sencillo como que un desarrollador lo elabore en un único lanzamiento

Situaciones que pueden darse:

- Un actor se comunica con un caso de uso (si se trata de un actor primario la comunicación la iniciará el actor, en cambio si es secundario, el sistema será el que inicie la comunicación).
- Un caso de uso extiende otro caso de uso.
- Un caso de uso utiliza otro caso de uso.

Ventajas

La técnica de caso de uso tiene éxito en sistemas interactivos, ya que expresa la intención que tiene el actor (su usuario) al hacer uso del sistema.
Como técnica de extracción de requerimiento permite que el analista se centre en las necesidades del usuario, qué espera éste lograr al utilizar el sistema, evitando que la gente especializada en informática dirija la funcionalidad del nuevo sistema basándose solamente en criterios tecnológicos.

A su vez, durante la extracción (*elicitation* en inglés), el analista se concentra en las tareas centrales del usuario describiendo por lo tanto los casos de uso que mayor valor aportan al negocio. Esto facilita luego la priorización del requerimiento.

Limitaciones

Los casos de uso pueden ser útiles para establecer requisitos de comportamiento, pero no establecen completamente los requisitos funcionales ni permiten determinar los requisitos no funcionales. Los casos de uso deben complementarse con información adicional como reglas de negocio, requisitos no funcionales, diccionario de datos que complementen los requerimientos del sistema. Sin embargo, la ingeniería del funcionamiento especifica que cada caso crítico del uso debe tener un requisito no funcional centrado en el funcionamiento asociado.

Buenas prácticas y recomendaciones de uso

Los casos de uso, como el resto de los requisitos, deben tener una redacción cuidada para evitar problemas de interpretación. En general, algunas recomendaciones a tener en cuenta son:

- El caso de uso debe describir qué debe hacer el sistema a desarrollar en su interacción con los actores y no cómo debe hacerlo. Es decir, debe describir sólo comportamiento observable externamente, sin entrar en la funcionalidad interna del sistema.
- El nombre del caso de uso debe ilustrar el objetivo que pretende alcanzar el actor al realizarlo.
- El caso de uso debe describir interacciones con los actores sin hacer referencias explícitas a elementos concretos de la interfaz de usuario del sistema a desarrollar.
- La invocación de unos casos de uso desde otros casos de uso (lo que se conoce como inclusión, o extensión si es condicional, en UML), sólo debe usarse como un mecanismo para evitar repetir una determinada secuencia de pasos que se repite en varios casos de uso. Nunca debe usarse para expresar posibles menús de la interfaz de usuario.
- Se debe ser cuidadoso al usar estructuras condicionales en la descripción del caso de uso, ya que los clientes y usuarios no suelen estar familiarizados con este tipo de estructuras, especialmente si son complejas.
- Se debe intentar que todos los casos de uso de una misma ERS estén descritos al mismo nivel de detalle.

- En los diagramas de casos de uso, debe evitarse que se crucen las líneas que unen los actores a los casos de uso.

A continuación, les dejo unos ejemplos de un caso de uso del tinder de mascotas.

Recuerden identificar TODAS las funcionalidades que quieran incluir en el Sistema y diagramarlo como caso de uso. (Para el tinder de mascotas por ejemplo: Login de Usuario, Registración Usuario, Crear mascota, Buscar mascota cercana, Like mascota, etc)La idea es hacer un caso de uso de manera sencilla.

CU 1. Login del usuario

CU 1. Login del usuario	
Descripción	El usuario ingresa al sistema mediante usuario y clave
Precondición	(Acá va algo sólo si debe existir algo antes que esta acción) Por ejemplo: El usuario debe estar dado de alta.
Acciones (Siempre se describe el camino feliz, osea la situación exitosa)	1- El usuario ingresa nombre y clave 2- El usuario presiona Aceptar 3- El sistema valida que exista un usuario con esas credenciales. 4- El sistema muestra la pantalla principal de la aplicación. (Debemos colocar paso a paso la interacción entre el/los actoresy el sistema)
Postcondición	El usuario se encuentra logueado y listo para usar el sistema ¿Qué pasa cuando termina el caso de uso?
Nota	Acá va alguna aclaración
Actores	Usuario Personas que interactúan con el Sistema
Alternativas	4.1- El sistema muestra pantalla de error con el mensaje "Sus credenciales son incorrectas, intente nuevamente." Acá van las acciones en el caso de que el camino no sea feliz, por ejemplo, ¿que pasa si el usuario o la clave con incorrectas?

CU 2. Registrar usuario

CU 2. Registrar usuario	
Descripción	El usuario nuevo desea registrarse para poder usar el sistema
Precondición	-
Acciones	1- El usuario ingresa a la pantalla de registración. 2- El sistema le solicita los datos personales. 3- El usuario ingresa los datos solicitados y presiona "Registrar". El sistema envía un mail confirmando la registración exitosa.
Postcondición	El usuario se registrado y listo para loguearse.
Nota	-

Actores	Usuario
Alternativas	-

Diagrama de Clases (UML)

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) puede ayudarte a modelar sistemas de diversas formas. Uno de los tipos más populares en el UML es el diagrama de clases. Popular entre los ingenieros de software para documentar arquitectura de software, los diagramas de clases son un tipo de diagrama de estructura porque describen lo que debe estar presente en el sistema que se está modelando.

Beneficios de los diagramas de clases

Los diagramas de clases ofrecen una serie de beneficios para toda organización. Usa los diagramas de clases UML para:

- Ilustrar modelos de datos para sistemas de información, sin importar qué tan simples o complejos sean.
- Comprender mejor la visión general de los esquemas de una aplicación.
- Expresar visualmente cualesquier necesidades específicas de un sistema y divulgar esa información en toda la empresa.
- Crear diagramas detallados que resalten cualquier código específico que será necesario programar e implementar en la estructura descrita.
- Ofrecer una descripción independiente de la implementación sobre los tipos empleados en un sistema que son posteriormente transferidos entre sus componentes.

Una vez que tengamos los casos de uso y las funcionalidades del proyecto pensadas. Vamos a proceder a hacer un diagrama de clases de las entidades de nuestro proyecto. Para pensar en las entidades que vamos a necesitar, vamos a pensar en las funcionalidades que debemos cumplir para que nuestro proyecto funcione.

A la hora de hacer nuestro diagrama de clases es importante pensar que mientras más clases tengamos mejor, obviamente todo tiene un limite, pero siempre que podamos dividir los problemas en problemas más pequeños mejor. Lo mismo se aplica a las clases, en vez de tener una clase con 10 atributos, para no hacer una clase más, ver la manera de dividir esa clase en 2 y que nos queden dos clases mucho más manejables.

ETAPA 3: PROGRAMACIÓN

Llegamos a la etapa de programación, esta va a ser la etapa en la que vamos a poner todas las funcionalidades, los casos de uso y el diagrama de clases en acción. Pero antes de ponernos a programar vamos a utilizar una serie de herramientas para poder organizarnos como equipo y saber que tenemos que funcionalidades tenemos que programar el día de hoy y que tareas nos van a quedar por hacer. Para esto vamos a utilizar Kanban.

Kanban

Actualmente, el término **Kanban** ha pasado a formar parte de las llamadas **metodologías ágiles**, cuyo objetivo es gestionar de manera general cómo se van completando las tareas. Kanban es una palabra japonesa que significa "tarjetas visuales", donde Kan es "visual", y Ban corresponde a "tarjeta".
Las principales ventajas de esta metodología es que es muy fácil de utilizar, actualizar y asumir por parte del equipo. Además, destaca por ser una técnica de gestión de las tareas muy visual, que permite ver a golpe de vista el estado de los proyectos, así como también pautar el desarrollo del trabajo de manera efectiva.

Los principios de la metodología Kanban

La metodología Kanban se basa en una serie de principios que la diferencian del resto de metodologías conocidas como ágiles:

- **Reducción del desperdicio:** Kanban se basa en hacer solamente lo justo y necesario, pero hacerlo bien. Esto supone la reducción de todo aquello que es superficial o secundario.
- **Mejora continua:** Kanban no es simplemente un método de gestión, sino también un sistema de mejora en el desarrollo de proyectos, según los objetivos a alcanzar.
- **Flexibilidad:** Lo siguiente a realizar se decide del backlog (o tareas pendientes acumuladas), pudiéndose priorizar aquellas tareas entrantes según las necesidades del momento (capacidad de dar respuesta a tareas imprevistas).

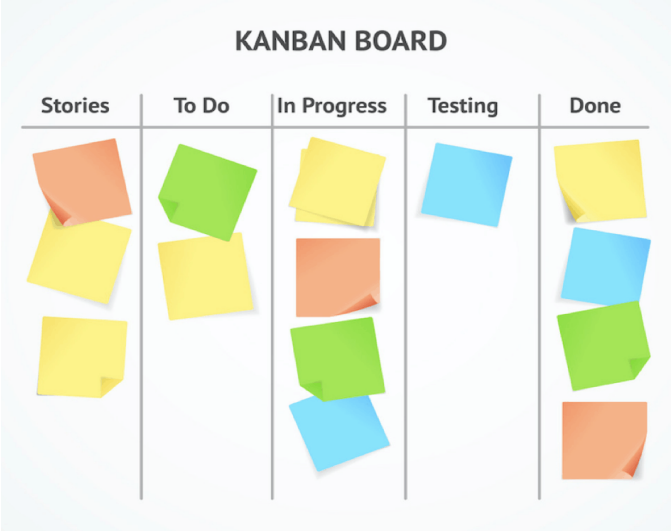
Pasos para configurar tu estrategia de Kanban

La aplicación del método Kanban implica la generación de un tablero de tareas que permitirá mejorar el flujo de trabajo y alcanzar un ritmo sostenible. Para implantar esta metodología, deberemos tener claro los siguientes aspectos:

1) DEFINIR EL FLUJO DE TRABAJO DE LOS PROYECTOS

Para ello, simplemente deberemos crear nuestro propio tablero, que deberá ser visible y accesible por parte de todos los miembros del equipo. Cada una de las columnas corresponderá a un estado concreto del flujo de tareas, que nos servirá para saber en qué situación se encuentra cada proyecto. El tablero debe tener tantas columnas como estados por los que pasa una tarea, desde que se inicia hasta que finaliza (p.e: diagnóstico, definición, programación, ejecución, testing, etc.).

Dicho tablero puede ser específico para un proyecto en concreto o bien genérico. No hay unas fases del ciclo de producción establecidas, sino que se definirán según el caso en cuestión, o se establecerá un modelo aplicable genéricamente para cualquier proyecto de la organización.



2) VISUALIZAR LAS FASES DEL CICLO DE PRODUCCIÓN

Al igual que Scrum, Kanban se basa en el principio de desarrollo incremental, dividiendo el trabajo en distintas partes. Esto significa que no hablamos de la tarea en sí, sino que lo dividimos en distintos pasos para agilizar el proceso de producción.

Normalmente cada una de esas partes se escribe en un post-it y se pega en el tablero, en la fase que corresponda. Dichos post-its contienen la información básica para que el equipo sepa rápidamente la carga total de trabajo que supone: normalmente descripción de la tarea con la estimación de horas. Además, se pueden emplear fotos para asignar responsables, así como también usar tarjetas con distintas formas para poner observaciones o indicar bloqueos (cuando una tarea no puede hacerse porqué depende de otra).

Al final, el objetivo de la visualización es clarificar al máximo el trabajo a realizar, las tareas asignadas a cada equipo de trabajo (o departamento), así como también las prioridades y la meta asignada.

3) STOP STARTING, START FINISHING

Este es el lema principal de la metodología Kanban. De esta manera, se prioriza el trabajo que está en curso en vez de empezar nuevas tareas. Precisamente, una de las principales aportaciones del Kanban es que el trabajo en curso debe estar limitado y, por tanto, existe un número máximo de tareas a realizar en cada fase.

En realidad, se trata de definir el máximo número de tareas que podemos tener en cada una de las fases (p.e: 3 tareas en la fase de planificación; 2, en la fase de desarrollo; una, en la fase de pruebas, etc.) y, por tanto, restringir el trabajo en curso.

De esta manera, se pretende dar respuesta al problema habitual de muchas empresas de tener muchas tareas abiertas, pero con una ratio de finalización muy bajo. Aquí lo importante es que las tareas que se abran se cierren antes de empezar con la siguiente.

Trello

Para vuestro tablero en el curso de programación hemos definido que usen la herramienta de Trello. Trello nos deja crear tableros para poner todas nuestras tareas y compartirlos con gente. Pueden hacer un tablero general tanto con tareas para el back y el front o pueden hacer un tablero para el back y un tablero para el front.

Recomendamos que creen un tablero con 3 columnas, tareas a realizar, tareas en proceso y tareas finalizadas, de esa manera va a quedar claro como van con las tareas y cuales ya se han terminado.

Es importante que se pongan tareas alcanzables por **días o semanas**, no pongan tareas que pueden llegar a llevar un mes. También es importante no poner tareas muy generales, sino ser lo más especifico posible con la tarea a realizar en sí. Por ejemplo: en vez de poner hacer login usuario, dividir eso en varias tareas, crear entidad usuario, crear el método del login, etc. Y así poder tener tareas alcanzables en un par de días o un par de semanas.

Github

La ultima herramienta que vamos a utilizar para organizarnos es GitHub, es importante que al que hayan designado como líder del proyecto sea el encargado de crear el repositorio y de cuando llegue el momento revisar el merge final a la rama Master, de esa manera evitamos que todo el equipo esté enviando cosas a la rama Master.

Recomendamos que tanto el equipo de back como el equipo de front tengan su propio repositorio. Ya que el equipo de back va a trabajar en un proyecto de Netbeans, mientras que el de front puede llevar a trabajar en Visual Studio Code.

Otra recomendación para los dos equipos es, discutir el nombre de las variables que se van a usar y dejarlas guardadas en un pdf para evitar problemas y conflictos en el repositorio.

¡¡Con todo esto dicho y hecho no queda más que empezar a trabajar!!
Recordemos que lo importante de este proyecto es disfrutar de la experiencia de trabajar en equipo y hacer un proyecto en un ambiente sin presiones y con la ayuda de los profes y los coachs. Asi que siempre recuerden que lo importante es disfrutar!!