MÓDULO II:
Desarrolla software con herramientas orientadas a la productividad.

SUBMÓDULO 2:
Aplica metodologías ágiles para el desarrollo de software

Compendio aula invertida

Docente: Mtra. Lizbeth Alcántara Blas

No.	TAREA	FECHA DE ENTREGA	ENTREGA	
			😀	😖
1	Leer la **Lectura 1. Introducción a metodologías ágiles**. Identificar las diferencias entre desarrollo tradicional y ágil, los 4 valores del Manifiesto Ágil, los principios y beneficios de las metodologías ágiles, así como la ética y buenas prácticas en el desarrollo de software. Generar imágenes alusivas al contenido de la lectura.			
2	Leer la **Lectura 2. SCRUM**. Identificar qué es *SCRUM* y cuáles son los roles que lo componen: *Product Owner*, *Scrum Master* y *Scrum Team*; comprender las responsabilidades de cada uno. Realizar recortes o dibujos que representen los conceptos clave de la lectura.			
3	Leer la ***Lectura 3. Recopilación de información***. Comprender la importancia de recopilar información para el desarrollo de un proyecto, así como las técnicas y herramientas para hacerlo. Elaborar un resumen ilustrado (con dibujos o recortes) que sintetice el contenido de la lectura.			
4	**Trabajo de campo (solo representantes):** Los representantes del equipo acuden a la comunidad de Las Moras, realizan la entrevista con los delegados o la instructora de otomí y documentan su experiencia (fotos, audio o video, con permiso). El resto del equipo apoya en la preparación y análisis de la información.			
5	Investigar la estructura básica de un informe de entrevista. Identificar los elementos que debe contener (presentación, desarrollo, preguntas, respuestas, conclusiones, anexos). Elaborar un esquema o guía con los apartados que debe contener el informe.			
6	Leer la **Lectura 4. Planeación del proyecto**. Identificar cómo se definen los objetivos de un proyecto, quiénes son los stakeholders y los usuarios del sistema; comprender su importancia. Elaborar un mapa de actores (stakeholders) del proyecto Ñätho Digital y redactar un objetivo general del proyecto.			
7	Leer la **Lectura 4. Planeación del proyecto.** Comprender qué es una hoja de ruta (Roadmap) y una visión del producto; identificar su utilidad en la planeación estratégica. Construir el Roadmap del proyecto (organizado por fases, de septiembre a diciembre 2026) y redactar la Visión del Producto de Ñätho Digital.			
8	Leer la **Lectura 5. *Sprint* y eventos de *SCRUM*.** Identificar qué es un Sprint, su duración (1 a 4 semanas) y su propósito; comprender cada uno de los eventos de *SCRUM*: *Sprint Planning* (qué y cómo se va a hacer), Daily *Scrum* (sincronización diaria de 15 min), *Sprint Review* (inspección y *feedback*), *Sprint* Retrospective (mejora continua). Elaborar un glosario con los términos clave. **Preparar materiales:** papel bond, tijeras, plumines y cinta adhesiva para la actividad en clase.			
9	Leer la **Lectura 6. Requerimientos funcionales y no funcionales**. Identificar las características de cada tipo de requerimiento y comprender sus diferencias. Elaborar un glosario con los términos clave.			
10	Leer la **Lectura 7. Técnica MoSCoW**. Comprender el significado de cada categoría: Must (debe tener), Should (debería tener), *Could* (podría tener), *Won't* (no tendrá); analizar ejemplos y practicar la clasificación de requerimientos. Elaborar un cuadro comparativo de la técnica *MoSCoW*. **Clasificar 5 requerimientos del catálogo de Ñätho Digital** (disponible en Google Drive) en *Must*, *Should*, *Could* o *Won't*, y **escribir una breve justificación** para cada clasificación. **Importante:** No realizar todavía el Tablero *MoSCoW* definitivo (se construirá en clase).			
11	Leer la **Lectura 8. Historias de Usuario.** Comprender la estructura de una historia: **"Como [tipo de usuario], quiero [acción/necesidad], para [beneficio]"**. Identificar las características de una buena historia, qué son los criterios de aceptación y cómo deben ser claros, observables y comprobables; reconocer la diferencia entre Historia de Usuario y Criterio de Aceptación. Analizar ejemplos y detectar errores en historias mal redactadas. **Revisar el Tablero *MoSCoW* Definitivo**, seleccionar 2 requerimientos clasificados como Must o Should y **redactar sus Historias de Usuario con sus respectivos Criterios de Aceptación**.			
12	Leer la ***Lectura 9. Product Backlog y MVP***. Comprender qué es el *Product Backlog*, su propósito y estructura; identificar el concepto de valor de negocio; analizar qué es un *MVP* (Producto Mínimo Viable) y su relación con el valor para el usuario y el alcance del proyecto. Revisar ejemplos y analizar qué funcionalidades deberían formar parte de un *MVP*. **Consultar las Historias de Usuario de Ñätho Digital y proponer un orden preliminar** de acuerdo con su valor de negocio (de mayor a menor).			
13	**Leer la Lectura 9. Product Backlog y MVP. Ordenar 5 Historias de Usuario** por valor de negocio (de mayor a menor). **Proponer de manera individual** cuáles de ellas conformarían el *MVP* y **justificar** por qué cada una es indispensable para la primera versión del producto.			
14	**Integrar TODOS los productos** elaborados durante el parcial (Hitos 1 al 4) en el **Expediente de Planeación Ágil** durante la clase, con la guía del docente. **Asegurarse de que todos los archivos estén organizados** en la carpeta digital del proyecto.			

Lectura 1. Introducción a las metodologías ágiles

1. ¿Qué son las metodologías ágiles?

Las metodologías ágiles son formas de organizar el desarrollo de software que buscan que los equipos puedan trabajar de manera colaborativa, entregar avances frecuentes, recibir retroalimentación y adaptarse a los cambios.

A diferencia de un proceso en el que se intenta definir todo desde el principio y mantenerlo sin modificaciones, el enfoque ágil permite revisar continuamente el trabajo y realizar ajustes cuando aparecen nuevas necesidades.

Idea clave: Ser ágil no significa trabajar sin plan. Significa planear, desarrollar, revisar y adaptar de manera continua.

---

2. Desarrollo tradicional vs. desarrollo ágil

Existen diferentes formas de organizar un proyecto de software. Dos enfoques que permiten comprender la diferencia son el desarrollo tradicional y el desarrollo ágil.

Desarrollo tradicional	Desarrollo ágil
Busca definir gran parte del proyecto desde el inicio.	Planea y desarrolla de manera progresiva.
El cambio puede ser difícil de incorporar.	El cambio forma parte del proceso.
Las entregas suelen concentrarse hacia el final.	Se realizan entregas frecuentes de avances.
El trabajo puede organizarse por etapas consecutivas.	El trabajo se realiza mediante ciclos iterativos.
La retroalimentación puede llegar después de desarrollar gran parte del producto.	La retroalimentación se obtiene durante el desarrollo.
Se da mayor importancia al seguimiento del plan inicial.	Se da mayor importancia a responder a las necesidades y cambios.

Ejemplo: Imaginemos que un equipo debe desarrollar una plataforma web.

Enfoque tradicional	Enfoque ágil
Puede intentar definir desde el principio todas las funcionalidades, diseños y requisitos para después desarrollar el sistema.	El equipo puede comenzar con las funcionalidades más importantes, desarrollar una primera versión, mostrarla a los usuarios, recibir comentarios y utilizar esa información para mejorar las siguientes versiones.

En pocas palabras

- **Tradicional:** Planear → desarrollar → entregar.
- **Ágil:** Planear → desarrollar → revisar → recibir retroalimentación → mejorar → repetir.

3. El Manifiesto Ágil

El Manifiesto para el Desarrollo Ágil de Software establece cuatro valores fundamentales que orientan la manera de trabajar de los equipos ágiles.

1. Individuos e interacciones sobre procesos y herramientas.

Las personas y la comunicación entre ellas tienen mayor importancia que depender únicamente de procesos o herramientas.
Esto no significa que las herramientas no sean importantes. Significa que una buena herramienta no sustituye la colaboración y comunicación del equipo.

2. Software funcionando sobre documentación extensiva.

Es importante contar con documentación útil, pero se da mayor prioridad a lograr que el software funcione y aporte valor.
La documentación debe apoyar el desarrollo, no convertirse en un objetivo que impida entregar resultados.

3. Colaboración con el cliente sobre negociación contractual.

El cliente o usuario debe participar durante el desarrollo.
Su retroalimentación permite verificar si el producto realmente responde a sus necesidades.

4. Responder al cambio sobre seguir un plan.

Un proyecto puede cambiar porque aparecen nuevas necesidades, problemas o información.
El enfoque ágil considera que responder adecuadamente a esos cambios puede generar un mejor producto.
Los valores del Manifiesto Ágil no significan que los elementos de la derecha no sean importantes. Significan que se da mayor valor a los elementos de la izquierda.

4. Principios de las metodologías ágiles.

Además de los cuatro valores, el Manifiesto Ágil establece principios que orientan el trabajo de los equipos. Entre los más importantes se encuentran:

- Buscar la satisfacción del cliente mediante entregas tempranas y continuas.
- Aceptar cambios en los requisitos, incluso durante el desarrollo.
- Entregar software funcional con frecuencia.
- Mantener una comunicación constante entre las personas involucradas.
- Confiar en las personas que realizan el trabajo.
- Favorecer la comunicación directa.
- Considerar el software funcionando como una medida importante del progreso.
- Mantener un ritmo de trabajo sostenible.
- Buscar continuamente la excelencia técnica y un buen diseño.
- Promover equipos capaces de organizar su propio trabajo.
- Reflexionar periódicamente sobre cómo mejorar la forma de trabajar.

Idea central

Los principios ágiles buscan que el equipo: colabore + entregue valor + escuche al usuario + acepte cambios + mejore continuamente.

5. Beneficios de las metodologías ágiles

La aplicación de principios ágiles puede proporcionar diferentes beneficios al desarrollo de software.

- **Mayor adaptación:** El equipo puede responder a cambios en las necesidades del usuario.
- **Retroalimentación frecuente:** Los avances pueden revisarse antes de que termine todo el proyecto.
- **Entregas progresivas:** El producto se construye y entrega por partes.
- **Detección temprana de problemas:** Los errores pueden identificarse durante el desarrollo y no únicamente al final.
- **Mayor colaboración:** Los integrantes del equipo mantienen una comunicación constante.
- **Mayor participación del usuario:** El usuario puede proporcionar retroalimentación sobre los avances.
- **Mejora continua:** El equipo analiza lo realizado y busca mejorar tanto el producto como su forma de trabajo.

6. Ética en el desarrollo de software

Desarrollar software no consiste únicamente en programar correctamente. También implica actuar con responsabilidad, honestidad y respeto hacia las personas que utilizarán el producto.

La ética ayuda a tomar decisiones que protejan a los usuarios, al equipo y a la organización.

Algunas prácticas éticas importantes

- **Respetar la privacidad**

La información de los usuarios debe manejarse de manera responsable y únicamente para los fines establecidos.

- **Proteger la información**

El equipo debe procurar que los datos y sistemas cuenten con medidas adecuadas de seguridad.

- **Ser honestos**

No se deben ocultar deliberadamente errores, problemas, riesgos o limitaciones importantes del software.

- **Respetar el trabajo de otras personas**

No se debe presentar como propio el código, diseño, documentación o trabajo realizado por otra persona.

- **Evitar prácticas dañinas**

El software no debe diseñarse deliberadamente para engañar, perjudicar o poner en riesgo a los usuarios.

- **Cumplir acuerdos y responsabilidades**

Cada integrante debe cumplir los compromisos establecidos y comunicar oportunamente cuando exista un problema que pueda afectar el proyecto.

7. Buenas prácticas en el desarrollo de software

Las buenas prácticas son formas de trabajar que ayudan a obtener productos de mejor calidad y a mantener un ambiente profesional.

Entre ellas se encuentran:

- Escribir código claro y organizado.
- Utilizar nombres comprensibles para variables, funciones y archivos.
- Documentar lo necesario.
- Realizar pruebas.
- Revisar el código.
- Utilizar sistemas de control de versiones.
- Mantener respaldos del trabajo.
- Respetar las reglas del repositorio.
- Comunicar los problemas oportunamente.
- Proteger las contraseñas y datos sensibles.
- Respetar licencias y derechos de autor.
- Trabajar de manera colaborativa.
- Aceptar y proporcionar retroalimentación de manera respetuosa.

---

8. Ética y trabajo en equipo

En un proyecto ágil, la responsabilidad no corresponde únicamente a una persona.

Cada integrante debe contribuir al buen funcionamiento del equipo.

Por ejemplo:

Un buen integrante del equipo:

- Comparte sus conocimientos.
- Ayuda a sus compañeros.
- Reconoce los errores.
- Respeta las opiniones de los demás.
- Cumple los acuerdos.
- Comunica los problemas.
- Da crédito al trabajo de otras personas.
- Busca soluciones en lugar de culpar.

La calidad de un software también depende de la calidad ética y profesional de las personas que lo desarrollan.

9. Ejemplo aplicado a Ñätho Digital

Supongamos que el equipo está desarrollando una funcionalidad para que estudiantes puedan aprender palabras en otomí.

Aplicación del enfoque ágil

1. El equipo desarrolla una primera versión, la prueba y recibe comentarios de los usuarios.
2. Después puede realizar ajustes y agregar nuevas funcionalidades.
3. Aplicación de la ética

   El equipo debe:

   - Respetar la cultura y conocimientos de la comunidad.
   - Solicitar autorización para utilizar materiales que correspondan a otras personas.
   - Dar reconocimiento a quienes proporcionen información o materiales.
   - Utilizar responsablemente fotografías, audios y otros recursos.
   - Evitar publicar información que no deba hacerse pública.
   - Presentar de manera honesta las capacidades y limitaciones del sistema.

De esta manera, la metodología ágil y la ética trabajan juntas para desarrollar un software útil, responsable y respetuoso con sus usuarios y comunidad.

10. Lo que debes recordar

- **DESARROLLO TRADICIONAL:** Busca seguir un plan definido y avanzar principalmente mediante etapas establecidas.
- **DESARROLLO ÁGIL:** Trabaja de manera iterativa, colaborativa y adaptable.
- **MANIFIESTO ÁGIL:** Establece 4 valores fundamentales para orientar el desarrollo ágil.
- **PRINCIPIOS ÁGILES:** Colaboración, entregas frecuentes, adaptación y mejora continua.
- **BENEFICIOS:** Permiten responder mejor a los cambios, obtener retroalimentación y entregar valor progresivamente.
- **ÉTICA:** Implica desarrollar software de manera responsable, honesta y respetuosa.
- **BUENAS PRÁCTICAS:** Ayudan a mejorar la calidad del software y la forma de trabajar del equipo.

IDEA CENTRAL:

Las metodologías ágiles permiten desarrollar software de manera colaborativa, entregando valor progresivamente, aceptando cambios y mejorando continuamente, siempre con responsabilidad ética y profesional.

Act001. Generación de imágenes sobre el contenido de la Lectura 1.

LECTURA 2. SCRUM

1. Definición de Scrum

Scrum es un marco de trabajo ágil que permite organizar y desarrollar proyectos mediante ciclos cortos de trabajo llamados **Sprints**.

Su propósito es que un equipo pueda **trabajar de manera colaborativa, entregar avances funcionales, recibir retroalimentación y adaptarse a los cambios**.

Scrum se basa en tres ideas importantes:

- **Trabajo colaborativo:** los integrantes trabajan juntos para alcanzar un objetivo común.
- **Desarrollo iterativo:** el producto se construye y mejora por partes.
- **Adaptación:** el equipo puede ajustar el trabajo de acuerdo con los resultados y las necesidades del usuario.

Ejemplo

En un proyecto de software, en lugar de desarrollar toda la aplicación de una sola vez, el equipo puede trabajar primero en el registro de usuarios, después en el contenido principal y posteriormente en otras funcionalidades.

Cada avance puede revisarse y mejorarse antes de continuar.

Scrum permite avanzar paso a paso, revisar lo realizado y mejorar continuamente.

2. Roles de Scrum

Scrum establece **tres roles principales**:

1. **Product Owner**
2. **Scrum Master**
3. **Scrum Team**

Los tres trabajan de manera colaborativa y **ninguno está por encima de los demás**. Cada rol tiene responsabilidades diferentes.

2.1 Product Owner

El **Product Owner** representa la **voz del cliente o usuario**.

Es responsable de identificar las necesidades del producto, establecer prioridades y buscar que el desarrollo genere el mayor valor posible.

Sus principales responsabilidades son:

- Conocer las necesidades de los usuarios.
- Comunicar los requisitos al equipo.
- Priorizar lo que debe desarrollarse.
- Mantener actualizado el Product Backlog.
- Revisar si el producto responde a las necesidades planteadas.
- Ayudar a decidir qué funcionalidades aportan mayor valor.

Pregunta clave:

¿Qué debemos desarrollar y qué es más importante para el usuario?

2.2 Scrum Master

El **Scrum Master** es el **facilitador del equipo**.

Su función es ayudar a que Scrum se aplique correctamente y que el equipo pueda trabajar sin obstáculos innecesarios.

Sus principales responsabilidades son:

- Facilitar la aplicación de Scrum.
- Orientar al equipo sobre las prácticas de Scrum.
- Ayudar a identificar y eliminar impedimentos.
- Favorecer la colaboración.
- Facilitar las reuniones de Scrum.
- Promover la autoorganización del equipo.
- Evitar interferencias que dificulten el trabajo.

Importante

El Scrum Master **no es el jefe del equipo**.

No debe asignar tareas ni imponer cómo debe trabajar cada integrante. Su función es **facilitar y apoyar**.

Pregunta clave: ¿Qué necesita el equipo para poder avanzar?

2.3 Scrum Team

El **Scrum Team** es el grupo responsable de **desarrollar el producto**.

Está formado por personas con diferentes habilidades que colaboran para transformar los requisitos en un producto funcional.

Puede incluir, por ejemplo:

- Programadores.
- Diseñadores.
- Personas encargadas de pruebas.
- Especialistas en otras áreas necesarias para el proyecto.

Sus principales responsabilidades son:

- Comprender los requisitos.
- Planificar el trabajo.
- Desarrollar las funcionalidades.
- Realizar pruebas.
- Resolver problemas técnicos.
- Colaborar entre sus integrantes.
- Entregar incrementos funcionales del producto.

El Scrum Team debe trabajar de manera **colaborativa y autoorganizada**.

Pregunta clave: ¿Cómo vamos a construir y entregar el producto?

3. Los tres roles en una sola idea

Rol	Función principal	Pregunta que responde
Product Owner	Representa al usuario y prioriza las necesidades.	¿Qué debemos hacer primero?
Scrum Master	Facilita Scrum y ayuda a eliminar obstáculos.	¿Qué necesita el equipo para avanzar?
Scrum Team	Desarrolla y entrega el producto.	¿Cómo vamos a construirlo?

Para recordar:
Product Owner → prioriza
Scrum Master → facilita
Scrum Team → desarrolla

Los tres roles trabajan juntos para alcanzar el objetivo del proyecto.

Act002. Realizar recortes o dibujos que representen los conceptos clave de la lectura.

LECTURA 3 . RECOPILACIÓN DE INFORMACIÓN

Entrevistas

Es una conversación dirigida con un propósito específico, en la cual se usa un formato de preguntas y respuestas. En la entrevista hay que obtener:

- opiniones del entrevistado y
- sentimientos sobre el estado actual del sistema,
- objetivos de la organización y los personales,
- procedimientos informales para interactuar con las tecnologías de la información (HCI <Interacción humano computadora> aspectos ergonómicos, la capacidad de uso del sistema, qué tan placentero y divertido es el sistema, y qué tan útil es para apoyar las tareas individuales).

En la entrevista debemos establecer una relación con un extraño. Hay que generar confianza y comprensión rápidamente y, al mismo tiempo, mantener el control de la entrevista. También necesitamos vender el sistema, para lo cual hay que proveer información necesaria al entrevistado. Planear previamente la entrevista le facilitará dirigirla. Los cinco pasos para la preparación de una entrevista, que debe seguir el analista de sistemas:

Pasos para planear la entrevista

1. Leer el material sobre los antecedentes.
2. Establecer los objetivos de la entrevista.
3. Decidir a quién entrevistar.
4. Preparar al entrevistado.
5. Decidir sobre los tipos de preguntas y su estructura.

1. ANTECEDENTES: Lea y comprenda todo lo que pueda sobre los antecedentes de los entrevistados y la organización. Investigar la organización es aprovechar al máximo el tiempo invertido en las entrevistas, al no tener que hacer preguntas generales sobre antecedentes.

2. OBJETIVOS DE LA ENTREVISTA: Defina los objetivos de la entrevista a partir de los antecedentes investigados y de su propia experiencia, en las siguientes áreas:

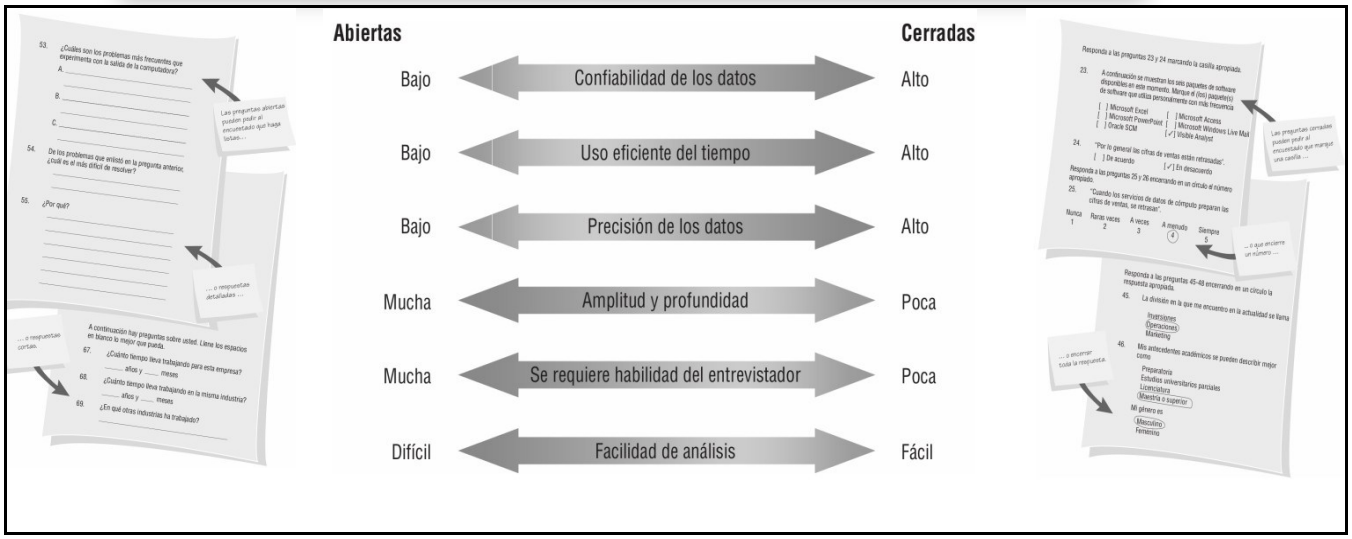
- *HCI* (utilidad y capacidad de uso del sistema, cómo se ajusta a los aspectos físicos, a las capacidades cognitivas de un usuario, si es interesante o estéticamente atractivo, y si al utilizar el sistema se obtienen o no las consecuencias deseadas)
- Procesamiento de información (fuentes de información y formatos de información)
- Comportamiento de los encargados de tomar las decisiones (frecuencia de la toma de decisiones, calidades de la información y el estilo de la toma de decisiones).

3. DECIDA A QUIÉN ENTREVISTAR: Incluya personas clave de todos los niveles que se vean afectados por el sistema en cierta forma. Trate de que la muestra sea representativa para indagar sobre la mayor cantidad posible de necesidades de usuario. Su contacto de la organización también le puede dar ideas sobre las personas que debe entrevistar.

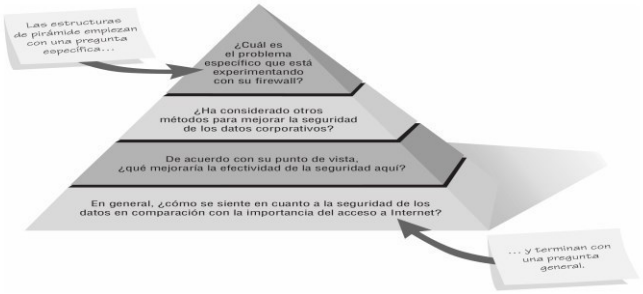
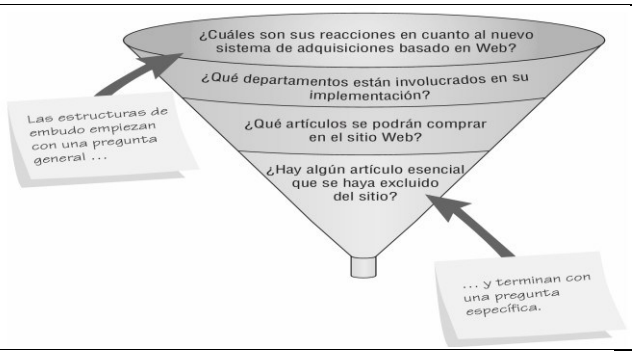
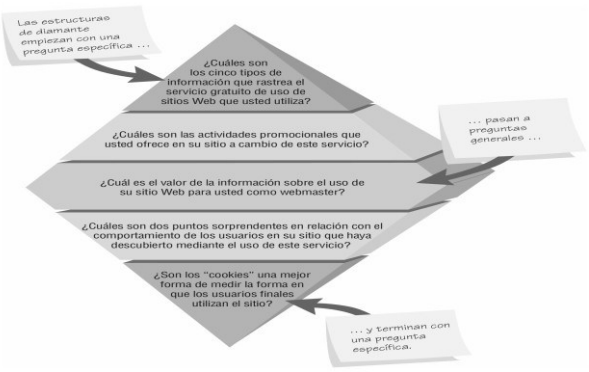
4. PREPARE AL ENTREVISTADO: Llame por teléfono o envíe un mensaje con anticipación, de manera que el entrevistado esté preparado; si la entrevista es muy detallada, envíe previamente el cuestionario para que pueda pensar en sus respuestas. Las entrevistas se deben realizar en persona. Deben durar de 45 minutos a 1 hora como máximo.

5. DECIDA SOBRE LOS TIPOS DE PREGUNTAS Y SU ESTRUCTURA: Redacte preguntas para cubrir las áreas clave de la HCI y el proceso de toma de decisiones que haya descubierto al momento de determinar los objetivos de la entrevista. Las técnicas de interrogación apropiadas son la base de la entrevista. Los dos tipos básicos de preguntas son:

TIPOS DE PREGUNTAS			
TIPOS	**VENTAJAS**	**DESVENTAJAS**	**EJEMPLOS**
ABIERTAS Describe a detalle las respuestas, que puede constar de dos palabras o de dos párrafos.	1. El **entrevistado**: • baja la guardia y encuentra el proceso más interesante. 2. El **entrevistador**: • percibe el vocabulario del entrevistado, lo cual refleja su educación, valores, posturas y creencias. 3. Se proveen muchos detalles, puede expresar mejor las preguntas y puede recurrir a ellas en caso de que tenga que improvisar.	1. Las preguntas pueden generar muchos detalles irrelevantes. 2. Se puede llegar a perder el control de la entrevista. 3. Pueden requerir demasiado tiempo debido a la cantidad obtenida de información útil. 4. Puede darse la impresión de que el entrevistador "no está preparado", sin objetivos bien definidos.	• ¿**Qué piensa** en cuanto a poner a todos los gerentes en una intranet? • **Por favor explique cómo** toma una decisión sobre la programación de tiempos y fechas. • **¿Qué opina sobre** el estado actual del comercio electrónico de negocio a negocio en su empresa? • **¿Cuáles son los objetivos críticos** de su departamento? • **Describa el proceso** de monitoreo disponible en línea … **¿cómo se procesan?** • **¿Cuáles son algunos de los errores comunes** al introducir datos que se cometen en este departamento? • **¿Cuáles son las mayores frustraciones** que ha experimentado durante la transición al comercio electrónico?
CERRADAS Limita la respuesta. Se proporciona pregunta y sus **cinco opciones de respuestas**.	1. Ahorro de tiempo. 2. Comparar las entrevistas con facilidad. 3. Van directo al grano. 4. Se mantiene el control sobre la entrevista. 5. Se cubre mucho terreno con rapidez. 6. Se obtienen datos relevantes.	1. Son aburridas para el entrevistado. 2. No proporcionan detalles adicionales. 3. No se puede generar una buena comunicación entre el entrevistador y el entrevistado.	• ¿**Cuántas veces** por semana se actualiza el repositorio del proyecto? • ¿**Está de acuerdo o en desacuerdo** en cuanto a que el sistema carece de seguridad? • ¿**Desea recibir un documento** impreso de su estado de cuenta cada mes? • ¿Está completo este formulario? • **En promedio, ¿cuántas** llamadas recibe el centro de atención al mes? • **Enliste sus dos principales prioridades** para mejorar la infraestructura de tecnología. • **¿Quién recibe esta entrada?** • **¿Cuál de las siguientes fuentes de información es más valiosa para usted**? a) Formularios completos de quejas de los clientes. b) Quejas por correo electrónico de los consumidores que visitan el sitio Web. c) Interacción cara a cara con los clientes. d) Mercancía devuelta.
• **Pregunta bipolar**, limita a un sí o no, verdadero o falso, de acuerdo o en desacuerdo. • **Preguntas sondeo**, ¿Por qué?, ¿Me puede dar un ejemplo? y ¿Podría explicarme eso?			



Ordenar las preguntas de la entrevista en una secuencia lógica

Estructura	Definición	Cuando utilizar	Ejemplo
Pirámide	Empieza con cerradas y termina con preguntas abiertas.	Su entrevistado necesita entrar en calor en cuanto al tema.	
Embudo	Inicia con preguntas abiertas y termina con preguntas cerradas.	El entrevistado está relacionado sentimentalmente con el tema y necesita libertad para expresar esos sentimientos.	
Diamante	Inicia con preguntas cerradas, después preguntas abiertas y concluye con cerradas	La estructura de diamante combina las ventajas de los otros dos métodos, pero tiene la desventaja de que toma más tiempo que las otras dos estructuras.	

Al concluir la entrevista, haga un resumen y provea retroalimentación sobre sus impresiones en general. Informe al entrevistado sobre los pasos a seguir, y lo que harán usted y los demás miembros del equipo a continuación. Pregunte al entrevistado con quién debería usted hablar en adelante. Establezca citas para dar seguimiento a las entrevistas, agradezca al entrevistado por su tiempo y despídase de mano.

Informe de la entrevista.

Deberá elaborar un reporto lo más pronto posible para no dejare de lado ningún detalle de la entrevista, en el cual considere la forma y el contenido:

- Forma (Portada, introducción, desarrollo, conclusiones y anexos (documentos de soporte).
- Contenido (Objetivos de la entrevista, contexto del proyecto, tipo de entrevista, formato, duración, perfil de los entrevistados, preguntas, respuestas análisis de la información para el diseño del sistema: funcionalidad, usabilidad, seguridad, problemas actuales y oportunidades de mejora).

Act003. Elaborar un resumen ilustrado (con dibujos o recortes) que sintetice el contenido de la lectura.

Act004. Trabajo de campo (solo representantes): Los representantes del equipo acuden a la comunidad de Las Moras, realizan la entrevista con los delegados o la instructora de otomí y documentan su experiencia (fotos, audio o video, con permiso). El resto del equipo apoya en la preparación y análisis de la información.

Describe la organización:

Act005. Investigar la estructura básica de un informe de entrevista.

Identificar los elementos que debe contener (presentación, desarrollo, preguntas, respuestas, conclusiones, anexos).

Act005. Elaborar un esquema o guía con los apartados que debe contener el informe.

LECTURA 4. PLANEACIÓN DEL PROYECTO

1. ¿Qué es la planeación de un proyecto?

La **planeación de un proyecto** consiste en definir qué se quiere lograr, para quién se desarrollará el producto y quiénes participarán o tendrán interés en sus resultados.
Antes de comenzar a desarrollar software es importante responder preguntas como:

- ¿Qué queremos lograr?
- ¿Por qué es necesario el proyecto?
- ¿Quién utilizará el sistema?
- ¿Quiénes tienen interés en el proyecto?
- ¿Qué necesidades debemos atender?

Una buena planeación ayuda al equipo a trabajar con un propósito claro y evita desarrollar funcionalidades que no respondan a una necesidad real.
Idea clave: Antes de programar, debemos entender **qué problema queremos resolver, para quién y con qué propósito**.

2. Objetivos del proyecto

Un **objetivo** expresa lo que se pretende lograr con el proyecto.
Debe indicar claramente **qué se quiere conseguir** y orientar las decisiones del equipo durante el desarrollo.
Un buen objetivo debe ser:

- **Claro:** debe entenderse fácilmente.
- **Concreto:** debe expresar un resultado específico.
- **Realista:** debe poder alcanzarse con los recursos disponibles.
- **Medible:** debe permitir comprobar si se logró.
- **Relacionado con una necesidad:** debe contribuir a resolver el problema identificado.

Ejemplo

Un objetivo poco claro sería: "Hacer una aplicación sobre la cultura."
Un objetivo más claro sería: "Desarrollar una plataforma digital que permita difundir información cultural, histórica y turística de la comunidad y facilitar el acercamiento de niñas, niños y jóvenes a su patrimonio."
El segundo objetivo permite comprender mejor **qué se quiere desarrollar y para qué**.

3. Objetivo general y objetivos específicos

Objetivo general	Objetivos específicos
Expresa el **resultado principal** que se quiere alcanzar con el proyecto.	Son resultados más concretos que ayudan a alcanzar el objetivo general.
Responde principalmente:	Responden:
¿Qué queremos lograr con todo el proyecto?	**¿Qué debemos lograr para cumplir el objetivo genera**

Ejemplo

Objetivo general:

Desarrollar una plataforma digital para difundir y valorar el patrimonio cultural de la comunidad.

Objetivos específicos:

- Recopilar información cultural e histórica relevante.
- Diseñar la estructura de la plataforma.
- Desarrollar las funcionalidades principales.
- Incorporar contenidos multimedia.
- Realizar pruebas con usuarios.
- Mejorar la plataforma a partir de la retroalimentación obtenida.

4. Objetivos SMART

Una forma de revisar si un objetivo está bien planteado es utilizar el modelo **SMART**.
SMART reúne cinco características que ayudan a formular objetivos claros y comprobables.

Letra	Significado	¿Qué pregunta responde?
S	**Specific – Específico**	¿Qué queremos lograr exactamente?
M	**Measurable – Medible**	¿Cómo sabremos que lo logramos?
A	**Achievable – Alcanzable**	¿Es posible lograrlo?
R	**Relevant – Relevante**	¿Por qué es importante?
T	**Time-bound – Temporal**	¿Cuándo debe lograrse?

En palabras sencillas Un objetivo SMART debe indicar:**qué se hará + cómo se comprobará + que sea posible + por qué importa + cuándo se logrará.**

Ejemplo

Un objetivo poco preciso sería:
"Mejorar la plataforma."
No permite saber exactamente qué se mejorará ni cómo se comprobará que el objetivo fue alcanzado.
Un objetivo SMART podría ser:
"Desarrollar durante el Sprint 1 la sección de contenidos culturales de la plataforma, incorporando al menos cinco contenidos revisados y validados por los responsables del proyecto."

¿Por qué es SMART?

- **S – Específico:** indica qué sección se desarrollará.
- **M – Medible:** establece al menos cinco contenidos.
- **A – Alcanzable:** se plantea para un periodo concreto y de acuerdo con los recursos disponibles.
- **R – Relevante:** contribuye al propósito del proyecto.
- **T – Temporal:** establece como límite el Sprint 1.

Recuerda: Un objetivo SMART permite responder:
¿qué?, ¿cuánto?, ¿es posible?, ¿para qué? y ¿cuándo?

5. ¿Quiénes son los stakeholders?

Los **stakeholders**, también llamados **partes interesadas**, son las personas, grupos u organizaciones que tienen algún interés en el proyecto o que pueden verse afectados por sus resultados.
No necesariamente utilizan directamente el sistema.
Pueden participar proporcionando:

- Información.
- Opiniones.
- Recursos.
- Autorizaciones.
- Retroalimentación.
- Requisitos.
- Apoyo al proyecto.

Algunos ejemplos de stakeholders

- Cliente o institución solicitante.
- Comunidad.
- Directivos.
- Docentes.
- Padres o tutores.
- Organizaciones.
- Proveedores.
- Personas especialistas en el tema.
- Equipo de desarrollo.

Importante:

Un stakeholder puede tener interés en el proyecto aunque **no utilice directamente el software**.

6. ¿Quiénes son los usuarios del sistema?

Los **usuarios** son las personas que utilizarán directamente el sistema o interactuarán con sus funcionalidades.
Por ejemplo, pueden:

- Consultar información.
- Registrar datos.
- Realizar actividades.
- Buscar contenidos.
- Descargar información.
- Utilizar juegos o herramientas.
- Administrar información.

Ejemplo: Si desarrollamos una plataforma educativa:

Usuarios:

- Estudiantes.
- Docentes.

Otros stakeholders:

- Directivos.
- Padres o tutores.
- Institución educativa.
- Personas que proporcionan contenidos.

7. Diferencia entre stakeholder y usuario

Aunque pueden coincidir, **no significan exactamente lo mismo**.

Stakeholder	Usuario
Tiene interés en el proyecto o puede verse afectado por él.	Utiliza directamente el sistema.
Puede o no utilizar el software.	Interactúa con el software.
Puede proporcionar requisitos, información o recursos.	Utiliza las funcionalidades.
Puede participar en decisiones o validaciones.	Proporciona retroalimentación desde su experiencia de uso.

Ejemplo sencillo

En una plataforma escolar:

- **Director:** puede ser stakeholder, aunque no utilice diariamente la plataforma.
- **Estudiante:** es usuario porque interactúa directamente con ella.
- **Docente:** puede ser usuario y también stakeholder.

Una misma persona puede ser al mismo tiempo stakeholder y usuario.

8. ¿Por qué es importante identificar a los stakeholders y usuarios?

Identificarlos desde el inicio permite conocer **quiénes tienen necesidades relacionadas con el proyecto** y evitar tomar decisiones basadas únicamente en suposiciones.

Esto ayuda a:

- Conocer mejor el problema.
- Identificar necesidades reales.
- Definir requisitos.
- Establecer prioridades.
- Evitar funcionalidades innecesarias.
- Obtener diferentes puntos de vista.
- Detectar posibles riesgos.
- Recibir retroalimentación.
- Validar si el producto realmente es útil.

---

9. ¿Cómo identificar a los stakeholders y usuarios?

El equipo puede realizar preguntas como:

Sobre los stakeholders

- ¿Quién solicitó el proyecto?
- ¿Quién tiene interés en sus resultados?
- ¿Quién proporciona información?
- ¿Quién puede verse afectado por el proyecto?
- ¿Quién puede autorizar o tomar decisiones?
- ¿Quién puede apoyar o limitar el proyecto?

Sobre los usuarios

- ¿Quién utilizará el sistema?
- ¿Con qué propósito lo utilizará?
- ¿Qué necesidades tiene?
- ¿Qué conocimientos tecnológicos posee?
- ¿Qué dificultades podría encontrar?
- ¿Qué espera obtener del sistema?

---

10. Ejemplo aplicado a Ñätho Digital

Para el proyecto **Ñätho Digital**, la identificación de las personas involucradas permite comprender mejor las necesidades que debe atender la plataforma.

Posibles usuarios

- Niñas, niños y jóvenes que utilizarán los contenidos y actividades.
- Personas de la comunidad que consulten información cultural y turística.
- Administradores o responsables de gestionar los contenidos.

Posibles stakeholders

- Comunidad de Las Moras.
- Personas que aportan información cultural.
- Autoridades o representantes de la comunidad.
- Instituciones educativas relacionadas con el proyecto.
- Docentes.
- Equipo de desarrollo.

Algunos participantes pueden pertenecer a **ambas categorías**.

Por ejemplo, un docente puede ser stakeholder por su interés en el proyecto y usuario si utiliza directamente la plataforma.

11. Relación entre objetivos, stakeholders y usuarios

Estos elementos están relacionados:

Necesidad o problema

↓

Objetivos del proyecto

↓

Identificación de stakeholders y usuarios

↓

Identificación de necesidades

↓

Definición de requisitos

↓

Desarrollo del sistema

Por eso, antes de decidir qué funcionalidades tendrá un sistema, el equipo debe comprender **qué problema quiere resolver y para quién lo está resolviendo**.

---

12. Lo que debes recordar

- **OBJETIVO:** Indica **qué se quiere lograr** con el proyecto.
- **OBJETIVO SMART:** Ayuda a formular objetivos **específicos, medibles, alcanzables, relevantes y temporales**.
- **STAKEHOLDER:** Es una persona, grupo u organización que **tiene interés en el proyecto o puede verse afectada por él**.
- **USUARIO:** Es la persona que **utiliza directamente el sistema**.

IMPORTANCIA

Identificar correctamente estos elementos permite desarrollar un software que responda a **necesidades reales** y no solamente a lo que el equipo supone que los usuarios necesitan.

IDEA CENTRAL

Un buen proyecto comienza por definir claramente qué quiere lograr, establecer objetivos que puedan comprobarse, identificar quiénes están involucrados y comprender quiénes utilizarán el sistema.

Act006. Elaborar un mapa de actores (stakeholders) del proyecto Ñätho Digital.

Act006. Redactar un objetivo general del proyecto.

LECTURA 4. PLANEACIÓN DEL PROYECTO

13. ¿Qué es la visión del producto?

La visión del producto es una descripción clara de lo que se quiere lograr con el producto y el valor que proporcionará a sus usuarios.
Permite que todas las personas involucradas en el proyecto tengan una idea común sobre:

¿Qué queremos crear?
¿Para quién lo estamos creando?
¿Qué necesidad queremos atender?
¿Qué valor queremos proporcionar?
¿Qué queremos lograr a futuro?

La visión no describe todas las funcionalidades del sistema. Explica principalmente hacia dónde queremos llevar el producto y por qué es importante.

Idea clave: La visión indica a dónde queremos llegar.

14. ¿Por qué es importante la visión del producto?

Una visión clara ayuda al equipo a tomar decisiones durante el desarrollo.
Permite:

- Mantener un propósito común.
- Orientar las decisiones del equipo.
- Identificar qué funcionalidades aportan valor.
- Evitar desarrollar elementos que no contribuyen al propósito.
- Comunicar el propósito del producto a los stakeholders.
- Mantener el enfoque cuando surgen cambios.

Ejemplo
Para Ñätho Digital, una posible visión del producto sería:
Crear una plataforma digital atractiva y accesible que acerque a niñas, niños y jóvenes a la lengua, cultura, historia y patrimonio de la comunidad, mediante contenidos interactivos y experiencias digitales.
Esta visión no indica todavía cómo se programará la plataforma. Indica qué se quiere conseguir y para quién.

15. ¿Qué es una hoja de ruta o Roadmap?

Una hoja de ruta (Roadmap) es una representación general de cómo se espera que evolucione un producto a lo largo del tiempo.
Permite organizar y comunicar:

- Los principales objetivos.
- Las funcionalidades o resultados que se planean desarrollar.
- Las prioridades.
- Las etapas o periodos de desarrollo.
- La evolución esperada del producto.

El Roadmap proporciona una visión general del camino del proyecto, pero no necesita contener todas las tareas que realizará el equipo.

Idea clave: El *Roadmap* muestra el camino para llegar hacia la visión del producto.

16. ¿Para qué sirve un *Roadmap*?

Una **hoja de ruta** ayuda a:

- Orientar al equipo
- Permite conocer qué se busca conseguir en cada etapa.
- Organizar prioridades
- Ayuda a determinar qué resultados son más importantes para el producto.
- Comunicar el proyecto
- Facilita explicar a los stakeholders cómo se espera que evolucione el producto.
- Visualizar el futuro
- Permite observar de manera general qué se pretende desarrollar después.
- Tomar decisiones

Cuando aparece una nueva propuesta, el equipo puede preguntarse:

¿Esta propuesta contribuye a la visión y a los objetivos del producto?

17. Diferencia entre visión y *Roadmap*

Aunque están relacionados, no son lo mismo.

Visión del producto = Destino	Roadmap = Camino
Indica hacia dónde queremos llegar.	Muestra cómo esperamos avanzar hacia ese objetivo.
Explica el propósito y valor del producto.	Organiza objetivos y resultados por etapas o periodos.
Tiene una perspectiva general.	Presenta una evolución planificada.
Responde: ¿Para qué y hacia dónde?	Responde: ¿Qué desarrollaremos y cuándo aproximadamente?

18. ¿El *Roadmap* es un plan rígido? No.

En un proyecto ágil, el *Roadmap* funciona como una guía estratégica, **no** como una lista inamovible.

A medida que el equipo obtiene nueva información, recibe retroalimentación o cambian las necesidades de los usuarios, el *Roadmap* puede actualizarse.

Por ejemplo, una funcionalidad que inicialmente estaba prevista para una etapa posterior puede convertirse en una prioridad si los usuarios consideran que es más importante.

En un proyecto ágil, planear no significa impedir los cambios. Significa tener una dirección y adaptarla cuando sea necesario.

19. Ejemplo de Roadmap para Ñätho Digital. Un *Roadmap* sencillo podría organizarse de la siguiente manera:

Este ejemplo muestra la evolución general del producto, pero no contiene todas las tareas que deberán realizar los estudiantes.

Etapa	Enfoque principal	Resultado esperado
Etapa 1	Investigación y definición	Necesidades, usuarios y contenidos identificados.
Etapa 2	Diseño y estructura	Prototipos y estructura de la plataforma.
Etapa 3	Desarrollo inicial	Funcionalidades principales disponibles.
Etapa 4	Pruebas y mejora	Retroalimentación incorporada y errores corregidos.
Etapa 5	Publicación y entrega	Plataforma funcional y documentación final.

20. Relación entre visión, objetivos y Roadmap

Estos elementos se complementan:
Problema o necesidad
↓
Visión del producto
↓
Objetivos
↓
Roadmap
↓
Prioridades y requisitos
↓
Sprints
↓
Desarrollo y entregas

Ejemplo. Problema: Existe una necesidad de fortalecer la difusión de la cultura y patrimonio de la comunidad.

- **Visión**:

Crear una plataforma digital que acerque a los jóvenes a la cultura y patrimonio de la comunidad.

- **Objetivo:**

Desarrollar una plataforma con contenidos culturales e interactivos.

- **Roadmap:**

Organizar las principales etapas para investigar, diseñar, desarrollar, probar y publicar la plataforma.

- **Sprints**:

Dividir el trabajo en ciclos cortos para construir y mejorar el producto.

21. Planeación estratégica

La planeación estratégica permite establecer una dirección general para alcanzar los resultados que se desean.
En un proyecto de software, la visión y el Roadmap ayudan a mantener esa dirección.
La **visión** responde: ¿A dónde queremos llegar?
Los **objetivos** responden: ¿Qué queremos lograr?
El **Roadmap** responde: ¿Cómo esperamos avanzar y en qué etapas?
El **Sprint** responde: ¿Qué vamos a trabajar ahora?
Esta relación permite pasar de una idea general a acciones concretas.

---

22. Lo que debes recordar

VISIÓN DEL PRODUCTO
Describe qué queremos crear, para quién y qué valor queremos proporcionar.
ROADMAP
Representa de manera general cómo se espera que evolucione el producto a través de diferentes etapas o periodos.
PLANEACIÓN ESTRATÉGICA
Ayuda a establecer una dirección general y a tomar decisiones coherentes con los objetivos del proyecto.

IDEA CENTRAL

La visión define el destino, el Roadmap muestra el camino y los Sprints permiten avanzar paso a paso hacia ese destino.

Act007. Redactar la Visión del Producto de Ñätho Digital.

Act007. Construir el Roadmap del proyecto (organizado por fases, de septiembre a diciembre 2026).