**FORMATO PARA EL DESARROLLO DE COMPONENTE FORMATIVO**

| PROGRAMA DE FORMACIÓN | Estructura de arquitectura para un proyecto software |
| --- | --- |

| COMPETENCIA | 220501116 diseñar la arquitectura del software de acuerdo con arquetipos y requisitos técnicos | RESULTADOS DE APRENDIZAJE | 220201116-01- Definir el tipo de arquitectura de acuerdo especificaciones técnicas y buenas prácticas de la industria del software. |
| --- | --- | --- | --- |

| NÚMERO DEL COMPONENTE FORMATIVO | CF01 |
| --- | --- |
| NOMBRE DEL COMPONENTE FORMATIVO | Introducción a la arquitectura de *software.* |
|  |  |
| PALABRAS CLAVE | Arquitectura, arquetipos, artefactos, diseño, *stakeholders.* |

| ÁREA OCUPACIONAL | 2 - CIENCIAS NATURALES, APLICADAS Y RELACIONADAS |
| --- | --- |
| IDIOMA | Español |

1. **TABLA DE CONTENIDOS:**
2. **Conocimientos en metodologías de desarrollo de s*oftware***
3. **Introducción a la educción de requisitos**
4. **Establecer las necesidades del negocio**
5. **Identificar los *stakeholders***
6. **Definición de requerimientos**
7. **Definir el tipo de arquitectura**
8. **DESARROLLO DE CONTENIDOS:**

**a**

**INTRODUCCIÓN**

La teoría de la arquitectura de *software*, se refiere a los fundamentos y componentes básicos para ejecutar procesos de computación, estos sirven para desarrollar las habilidades de los programadores al momento de construir una aplicación, permitiendo tener conceptos sobre patrones de desarrollo, abordando las teorías más importantes y utilizadas en la industria del desarrollo de *software*.

En el siguiente video, se dan a conocer más detalles y generalidades relacionadas a la temática de arquitectura de *software*.

| Anexo: CF01\_Video\_Introduccion |
| --- |

**1. Conocimientos en metodologías de desarrollo de *software***

Las metodologías en el desarrollo de *software* son parámetros o modelo utilizados para estructurar, controlar y planificar el proceso de desarrollo de sistemas de información, estos sirven en cualquier ámbito y no importa el lenguaje de programación que se utilice, son de suma importancia, convirtiéndose en una hoja de ruta para llegar a la consolidación de un producto satisfactorio.

| Hombre girando la perilla para cambiar la metodología de gestión de proyectos de pm tradicional a ágil. imagen compuesta entre una fotografía de mano y un fondo 3d. Foto Premium | La metodología de desarrollo de *software* que más se conocen por su utilidad son las ágiles y tradicionales o clásicas |
| --- | --- |

**Metodologías tradicionales o clásicas**

También conocidas como modelos de proceso descriptivo, fueron creadas con el fin de mitigar el desorden del desarrollo de *software*, que existía cuando se empezó a producir de manera masiva en la década de los 60, según Maida y Pacienzia (2015), constituyen un mapa razonablemente eficaz para los desarrolladores, aquí se pueden encontrar algunas de estas metodologías:

* **Modelo Cascada**: también se lo conoce como ciclo de vida o secuencial, este enfoque ordena los procesos de desarrollo por etapas: análisis, diseño, codificación, pruebas y mantenimiento.



SENA (2022). Fases de la metodología Cascada o Ciclo de Vida*.*

* **Modelo incremental: a**quí el producto se diseña, implementa y prueba de forma incremental hasta la entrega del *software.*
* **Modelo evolutivo: a**quí se desarrolla un *software* beta el cual se va mejorando según los comentarios o requerimientos por parte de los *stakeholders.*
* **Modelo espiral:** como su nombre lo indica es un modelo en espiral que se integra por actividades compuestas por bucles o iteraciones.
* **Modelo de desarrollo basado en componentes:**  en este modelo se desarrollan componentes abiertos y distribuidos de forma aislada, estos pueden ser reutilizables y al final se los une para conformar un producto final.

Dentro de las ventajas y desventajas al utilizar metodologías tradicionales en el desarrollo de *software* están:

| Ventajas   * Seguimiento detallado en cada una de las fases. * Son de fácil uso ya que siguen pasos necesarios al momento de desarrollar la aplicación. * Son muy usadas en proyectos de innovación. * Evalúan cada fase del proyecto obteniendo cambios con objetividad. * Determinan etapas claramente definiendo los roles. | Desventajas   * Presenta complejidad en la evaluación de riesgo. * Altos costos para implantar cambios. * Incomodidad para el cliente, ya que debe ser capaz de describir muchos detalles para realizar el proyecto. * Es muy flexible para algunos proyectos. |
| --- | --- |

**Metodologías agiles**

Estas metodologías permiten adaptar la forma de trabajo a las condiciones de un proyecto, dando un enfoque para la toma de decisiones ágil y flexible, basándose en el desarrollo iterativo e incremental, lo cual se verá reflejado en el producto final.

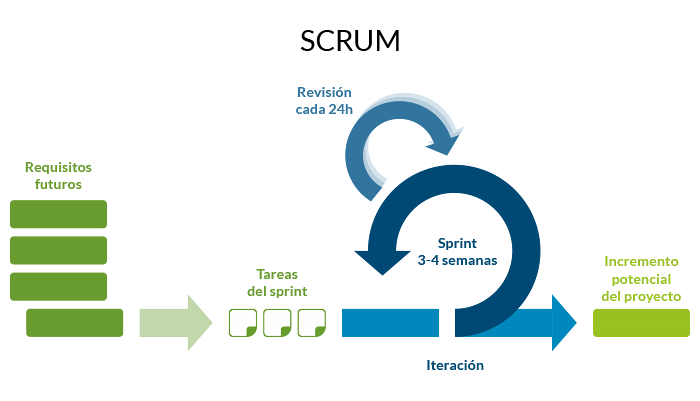
Ventajas y desventajas de Agile Project ManagementVentajas y desventajas de Agile Project Management

| Ágil agilidad ágil rápido rápido concepto volant Foto gratis | Este tipo de metodologías ágiles se centra más en el *software* que en la documentación exhaustiva. De acuerdo con Maida y Pacienzia, (2015), la alta competitividad actual hace que los sistemas de información se tengan que desarrollar de forma rápida para adaptarse a la organización. |
| --- | --- |

Estas metodologías se renuevan constantemente, aquí se muestran las más utilizadas:

* **Scrum**

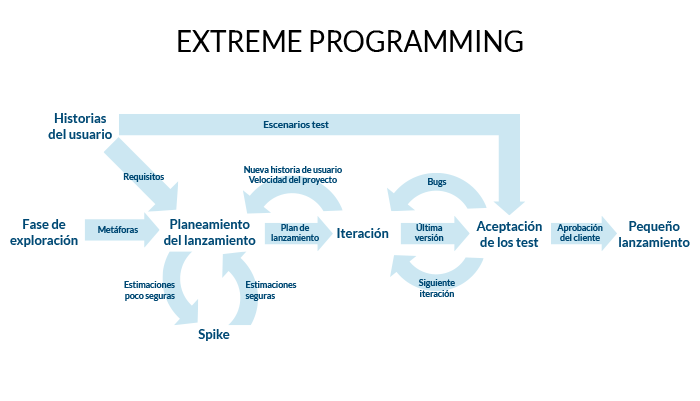
Es utilizada por toda clase de empresas debido a su capacidad para proporcionar colaboración de alto nivel y eficiencia para el proyecto de *software*.  Scrum consta de dos métodos diferentes y se pueden usar por separado; sin embargo, sus beneficios combinados hacen que esta metodología sea de las más populares. Scrum significa incremental, lo cual permite que los equipos desarrollen proyectos por separado; a esta división de los proyectos en partes considerables se les conoce como “*sprints*" o proyectos rápidos. Implementa el método científico   del empirismo y reemplaza un enfoque algorítmico programado por uno heurístico.



Nota. Lorbada (2017).

* **Extreme**

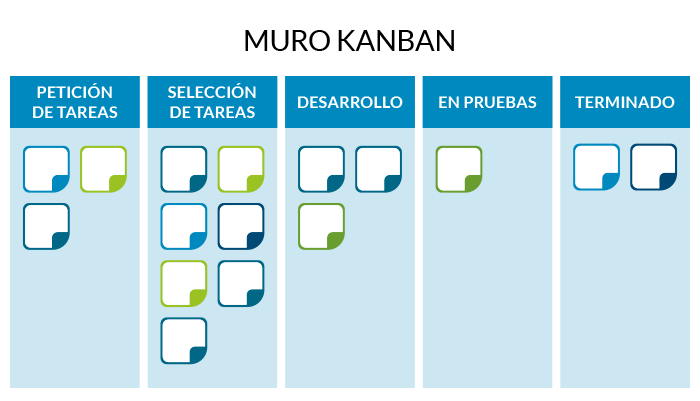
También conocido como eX*treme**Programming (XP*), su objetivo es producir *software* de mayor calidad y mejorar la dinámica para el equipo de desarrollo, *XP* es la más específica de las metodologías ágiles con respecto a las prácticas de ingeniería.



Nota. Lorbada (2017).

* **Kanban**

Esta metodología es muy popular al momento de implementarla en el desarrollo de *software* [ágil](https://www.atlassian.com/es/agile) y de [*DevOps*](https://www.atlassian.com/es/devops/what-is-devops). Requiere una comunicación en tiempo real, y una total transparencia de trabajo. Los elementos de trabajo se representan visualmente en un [tablero de Kanban](https://www.atlassian.com/es/software/jira/templates/kanban), lo que permite al equipo ver el estado de cada proceso en cualquier momento, logrando resultados más flexibles, rápidos, un enfoque más claro y transparencia a lo largo del ciclo de desarrollo.



Nota. Lorbada (2017).

* **Lean Startup**.

Este enfoque tiene sus raíces en el movimiento *Lean manufacturing* de la década de 1980. Sin embargo, ahora se considera una parte integral de la metodología de [desarrollo de *software* *agile*](https://searchsoftwarequality.techtarget.com/definition/agile-software-development), que está basada en siete principios:

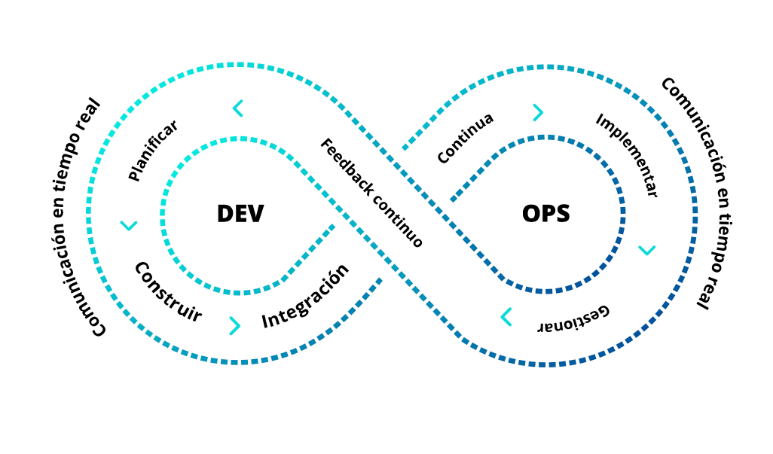
* + Evitar pérdida de tiempo en el desarrollo.
  + Construcción de calidad.
  + Amplifica el aprendizaje.
  + Retrasar compromisos innecesarios.
  + Entregas rápidas.
  + Respeto a personas.
  + Optimizar el proyecto.



Nota. Master marketing. (2020).

* **DevOPs**

Es un enfoque de desarrollo de *software* por medio del cual se puede desarrollar aplicaciones de alta calidad, de manera rápida y con mayor confiabilidad. Consta de varias etapas, desarrollo continuo, integración continua, pruebas continuas, implementación continua y monitoreo continuo.



Nota. Tenea. (2021). DevOps: Qué es y cómo beneficia a tu empresa . <https://blog.tenea.com/devops-que-es-y-como-beneficia-a-tu-empresa/>

* ***Design Thinking***

Ayuda a los desarrolladores de *software* a diseñar productos que tienen más probabilidades de acoplarse con el usuario, para crear productos que satisfagan sus necesidades y requisitos más fácilmente. Los desarrolladores de *software* deben pensar con empatía y comprender mejor al usuario final para hacer lograr un producto óptimo. Este enfoque *Design Thinking*, se desarrolla en cinco pasos: empatía, definición, ideación, prototipos y prueba.



Nota. ItMadris Digital School. (2020).

Algunas funciones claves que caracterizan a estas metodologías son (ver figura 1):

**Figura 1**

Tipos de metodologías Ágiles

| Anexo: CF01\_1\_Grafico interactivo\_metodologias \_agiles |
| --- |

Nota. SENA (2022). Metodologías ágiles más utilizadas.

La aplicación de las metodologías ágiles también tiene unas ventajas y desventajas:

| Ventajas   * Soluciones inmediatas durante el proceso. * Simplificación en la sobrecarga de procesos. * Calidad optimizada. * Mejora la previsibilidad a través de una mejor gestión de riesgos. * Acompañamiento del cliente y mejor interacción. * La entrega del producto y retroalimentación es más rápida. * Se optimizan recursos y resultados. | Desventajas   * Dificultad en determinar con precisión la cantidad de recursos debido a sus constantes cambios. * El equipo necesita tener habilidades sólidas. * Hay que estar muy pendiente de la documentación que se genere. |
| --- | --- |

Para ampliar los conocimientos en este tema consultar en el material complementario el artículo sobre **“Metodologías actuales de desarrollo de *software*”.**

**2. Introducción a la educción de requisitos**

Maida y Pacienzia (2015) plantean que la educción de requisitos también conocida como elicitation, consiste en hallar e identificar los requisitos que deben satisfacer una aplicación o *software*, en palabras sencillas es el modo en que se deben obtener estos requisitos. Es una actividad inherente a la ingeniería de *software* y forma parte de las metodologías ágiles, como una antesala al análisis de requisitos. Algunos procesos para la identificación de requisitos son:

* Definir la pregunta de investigación.
* Alcance de la revisión.
* Criterios de exclusión e inclusión.
* Conducta de búsqueda.
* Selección de trabajos primarios.
* Definición de criterios de análisis.
* Esquema de caracterización
* Análisis de resultados.

Los métodos más utilizados para obtener requisitos son:

* **Entrevistas**: el objetivo de realizar una entrevista es comprender las expectativas del cliente con respecto a la solución de *software*. Es imposible entrevistar a todas los *stakeholders*, por lo que los representantes de los grupos se seleccionan en función de su experiencia y credibilidad; las entrevistas pueden ser abiertas o estructuradas.
* **Sesiones de lluvia de ideas:** es una técnica grupal, su objetivo es generar muchas ideas nuevas y compartir puntos de vista diferentes.
* **Técnica de especificación de aplicación facilitada (FAST):** su fin es cerrar la brecha de expectativas, la diferencia entre lo que los desarrolladores creen que deben construir y lo que los clientes creen que obtendrán.
* **Despliegue de la función de calidad (QFD):** esta técnica mide la satisfacción del cliente, lo cual es la principal preocupación, por lo tanto, enfatiza los requisitos que son valiosos para el cliente.   
  Se identifican tres tipos de requisitos que son: normales, esperados y emocionales.
* **Enfoque de caso de uso:** en esta técnica se combina texto e imágenes para proporcionar una mejor comprensión de los requisitos, los casos de uso describen el “qué” de un sistema y no el “cómo”; por lo tanto, solo dan una visión funcional del sistema. Los componentes del diseño de casos de uso incluyen tres cosas principales: actor, casos de uso y diagrama de casos de uso.

El éxito de la elicitación depende de la madurez del analista, los desarrolladores, los usuarios y el cliente involucrado. En la siguiente figura 2 se puede observar un proceso de educción de requisitos:

**Figura 2**

Fases de la educción de requisitos 

Nota. SENA.(2022) Proceso desarrollado en la educción de requisitos.

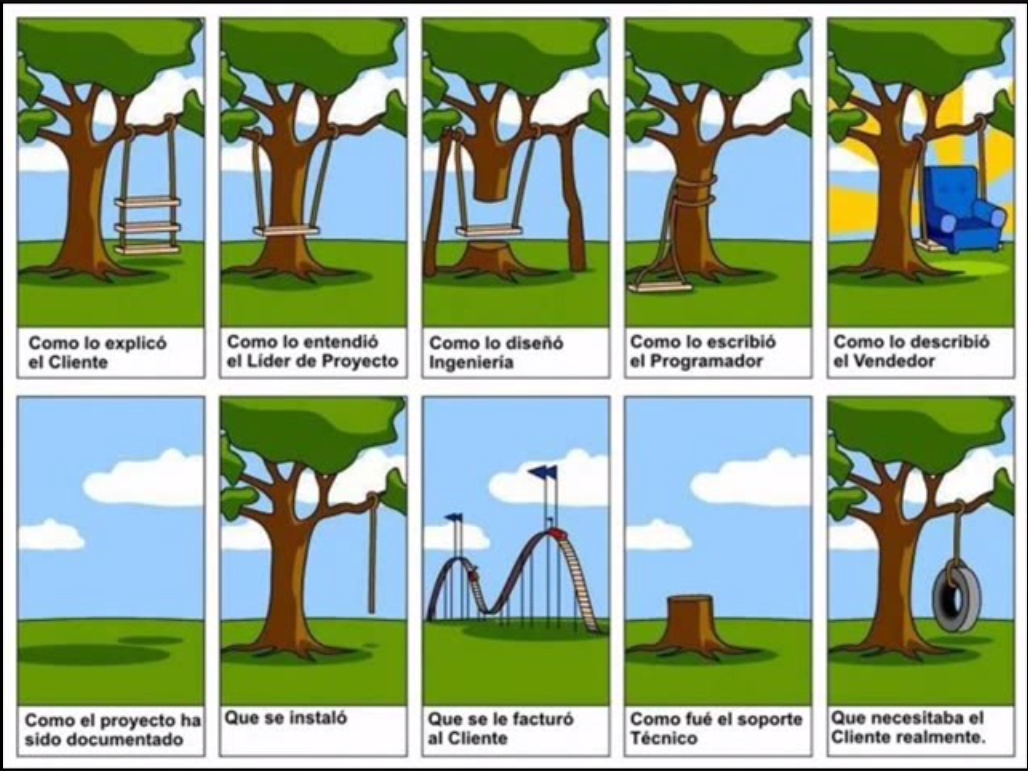
**3. Establecer las necesidades del negocio**

Esta parte del proceso (*customer requirements)*, tiene como objetivo principal el conocimiento del negocio del cliente, los procesos de este y el dominio de su problemática, enfocando el esfuerzo, en obtener todas las nuevas necesidades y características requeridas que el cliente considere necesarias para que el producto o servicio sea mínimamente viable frente al usuario final.

Además, como resultado de esta etapa del proceso, es imperativo que se genere  para cada requerimiento identificado su respectiva documentación en un lenguaje entendible para los involucrados en el proceso de construcción, a fin de lograr un entendimiento universal entre clientes y desarrolladores; siendo clave, encontrar un espacio compartido donde se almacenen las versiones finales aprobadas y pactadas por ambas partes; buscando que, no existan inconformidades entre lo solicitado por el cliente y lo construido en el *software*, permitiendo además, que todos dentro del equipo de desarrollo puedan acceder a ellas sin dificultad o restricción alguna (ver figura 3).

**Figura 3**

Versiones del producto final



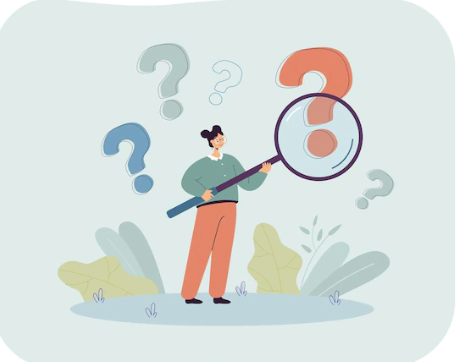
Nota. Martínez (2013).

Aunque no es obligatorio cumplir rigurosamente con ciertas reglas o ítems, se han identificado algunas actividades que permiten de una manera incremental y efectiva capturar las necesidades del negocio, que se listan a continuación:

* [**Analizar el dominio del problema**](http://www.juntadeandalucia.es/servicios/madeja/contenido/procedimiento/19#Estudiar_el_dominio_del_problema)

Comprender su negocio, tener clara las necesidades y poder proponer una solución efectiva. La primera etapa del proceso, busca obtener el conocimiento suficiente del universo del discurso con el cual, el equipo comprenderá el negocio y sus necesidades, con la intención de lograr una comunicación bidireccional eficaz con los usuarios e interesados finales que no tienen conocimiento técnico y así plantear una solución adecuada y a la medida de este.

Omitir esta actividad podría ocasionar que la solución final no sea lo que espera el cliente, afectando el esfuerzo y el tiempo invertido en la propuesta. Además, al momento de interactuar con los usuarios en reuniones sin el conocimiento del problema, disminuirá la confianza en la capacidad del equipo de desarrollo y por ende la imagen de la empresa en general.



* [**Identificar aspectos positivos y negativos de la situación actual**](http://www.juntadeandalucia.es/servicios/madeja/contenido/procedimiento/19#Identificar_aspectos_positivos_y_negativos_de_la_situacion_actual)

El objetivo de esta actividad es identificar qué aspectos actualmente funcionan adecuadamente y cuáles no del proceso o sistema (si se desea cambiar de sistema o se desea sistematizar algo que no existe aún), antes de iniciar a desarrollar el nuevo componente, esto buscando que se puedan conservar los aspectos positivos y enfocarse en dar solución a los negativos.

Omitir esta actividad podría conducir al equipo de desarrollo a repetir errores que se presenten actualmente y a dejar de lado prácticas o procesos que funcionan muy bien y solo deben replicarse, generando rechazo y resistencia por parte del usuario final.



* [**Analizar el modelo de negocio del cliente**](http://www.juntadeandalucia.es/servicios/madeja/contenido/procedimiento/19#Estudiar_el_modelo_de_negocio_del_cliente)

En este caso la prioridad es comprender el funcionamiento y los procesos del negocio actualmente, sean estos manuales o automatizados, permitiendo comprender de manera holística el contexto en el cual se desenvuelve la solución a desarrollar.

Omitir este punto puede generar que la solución propuesta no se integre de manera correcta con lo que el negocio del cliente solicita, llevando a que esta sea poco usada por su complejidad en la solución o por no ser lo esperado.



* [**Estudiar el entorno tecnológico del cliente**](http://www.juntadeandalucia.es/servicios/madeja/contenido/procedimiento/19#Estudiar_el_entorno_tecnologico_del_cliente)

Para este punto lo que se busca es identificar el ecosistema tecnológico del cliente en cuanto a *hardware*, *software* y licenciamiento, capacidad adquisitiva, escalabilidad horizontal y vertical, redes, madurez de los procesos internos tecnológicos y el despliegue de aplicaciones.

Omitir esta actividad provocaría que se proponga una arquitectura de *software* y despliegue incompatible con el entorno actual del cliente, provocando un dimensionamiento incorrecto y por ende un rendimiento inadecuado de la solución final.



* [**Obtener y documentar las necesidades de clientes y usuarios**](http://www.juntadeandalucia.es/servicios/madeja/contenido/procedimiento/19#Obtener_y_documentar_las_necesidades_de_clientes_y_usuarios)

El último punto y el más importante, consiste en obtener de manera clara, concisa y sintetizada las necesidades del cliente como objetivos del negocio, incluyendo los modelos de procesos; para luego ser trasladados a un documento base que explícitamente indique lo deseado por el cliente final.

Omitir esta actividad tan crítica, podría generar un producto final que no sea lo esperado por el cliente, puesto que, él inicialmente pidió una solución a un problema presente que no fue plasmado en un documento; el cual con el paso del tiempo, para el equipo de desarrollo se convirtió en otra solución tergiversada con respecto a la inicial, creando al final una solución incongruente con la idea primaria. Sumado a ello, esta idea transmutada presenta integración pobre o baja con el antiguo proceso o sistema por ser inconsistente con lo planteado inicialmente; lo cual, desencadena en una baja usabilidad del cliente por no ser lo esperado y una pérdida de tiempo y esfuerzo, dado que no es posible aplicar mejoras a dicho sistema que inició con errores.



 Estas actividades generan un concepto claro en cuanto a:

* Precio.
* Calidad.
* Funcionalidad.
* Fiabilidad.
* Rendimiento.
* Sustentabilidad.
* Eficiencia.
* Seguridad.
* Compatibilidad.
* Diseño.
* Accesibilidad.

Para determinar los requisitos del cliente, las empresas pueden investigar su mercado objetivo comprendiendo sus deseos y necesidades, también pueden adoptar un enfoque directo solicitando a sus clientes comentarios a través de encuestas, sondeos o redes sociales.

| Anexo: CF01\_3\_Infografia\_necesidades\_negocio |
| --- |

**4. Identificar los s*takeholders*.**

Los *stakeholders* son las partes interesadas en el proyecto, hacen referencia a todas las personas que se relacionan con las actividades y decisiones de la empresa para la cual se hará la solución informática; estos generan un papel fundamental en la elicitacion, especificación y validación de requisitos del proyecto.

Los *stakeholders* pueden ser internos y externos, y se debe evaluar qué función o relación tienen con el negocio. Sin su aporte, los desarrolladores están trabajando a partir de una lista incompleta de requisitos.

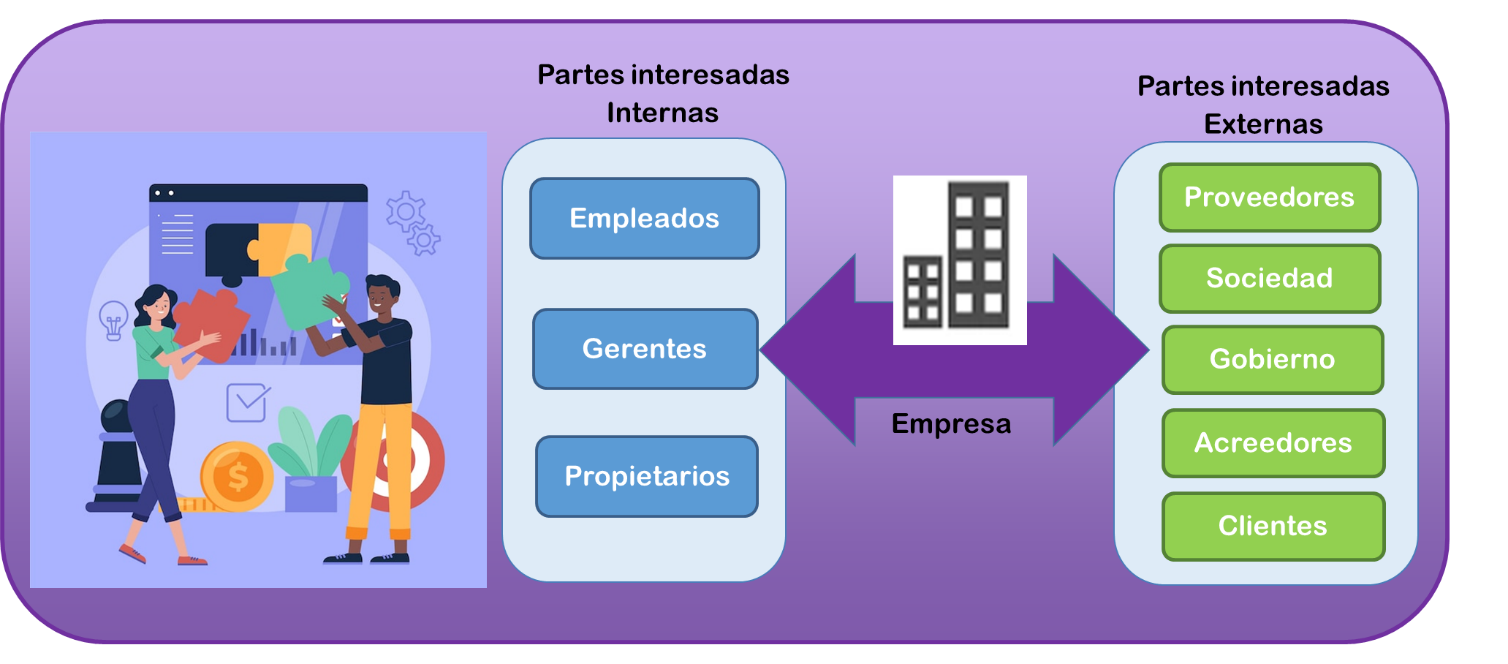
Es probable que surjan necesidades sorpresa durante el desarrollo; estas adiciones repentinas provocan un aumento del alcance, donde el proyecto aumenta más allá de sus límites establecidos y los requisitos iniciales de tiempo y presupuesto se ven obligados a estirarse para cubrir los nuevos requisitos. Eso no siempre es posible, es mucho más probable que haya que eliminar algunas características para cumplir con los plazos.

Para definir las partes interesadas (ver figura 4) se debe tener en cuenta los siguientes conceptos:

| .Concepto de personas de oficina Vector Premium | * Quién sabe lo suficiente sobre ellos para asesorar al equipo de desarrollo. * Quién tiene autoridad para realizar cambios en el plan de desarrollo una vez finalizado. * Hay alguna persona cuyo apoyo sea absolutamente vital para el éxito del proyecto. * De quién es necesaria la participación. * Quién usará o se verá afectado por el producto final. * Quién usa la herramienta actual o el software que reemplazará el nuevo software. * Qué departamentos utilizan los productos del software actual y propuesto. * Cómo cambiarán los flujos de trabajo. * Se modificarán o crearán puestos. * Qué restricciones legales y normativas se aplican. |
| --- | --- |

**Figura 4**

Identificación de los stakeholders



Nota. SENA (2022). Identificación de las partes interesadas de un proyecto.

**5. Definición de requerimientos**

Según Sommeville (2005), requerimiento son las necesidades que provienen del negocio de parte de los usuarios y *stakeholders.* Un requisito son las especificaciones puntuales sobre el sistema o *software*, que dan una solución específica a las necesidades o carencias del negocio. Los dos tipos de requerimientos son:

* **Requerimientos funcionales (RF**)

Describen la actividad que la aplicación debe realizar, como el comportamiento o funciones particulares. Son las características que tendrá la solución de *software* que pueden dividirse en dos partes:

| Concepto de desarrollo de app con diseño plano Vector Premium  *Requerimientos funcionales del sistema* | Grupo de personajes jóvenes vector gratuito  *Requerimientos funcionales del usuario* |
| --- | --- |
| Establece los detalles del servicio y restricciones de la aplicación. | *S*on los servicios o funciones que la aplicación debe hacer y que interactúan directamente con el usuario; como son las interfaces, diseños menú, imágenes entre otras. |

Algunos ejemplos de Requisitos Funcionales (RF) son:

* Describe los datos a ser ingresados en el sistema.
* Describe las operaciones que se realizarán por cada pantalla.
* Describen los flujos de trabajo realizados por el sistema.
* Descripción de los reportes del sistema y otras salidas.
* Define quién puede ingresar datos en el sistema.
* **Requerimientos no funcionales (RNF)**

Son los encargados de que la solución de *software* cumpla con su fin o dé cumplimiento al objetivo de la aplicación o solución de informática. También se les conoce como restricciones o requisitos de calidad; estos requerimientos no pueden ser vistos ni tocados por el usuario, están inmersos en las funciones de la programación y se dividen en tres partes:

| *Requisitos de producto* | *Requisitos de organización* | *Requisitos externos* |
| --- | --- | --- |
| Especifican el comportamiento de la aplicación. | Se producen a partir de las políticas y procedimientos de la organización. | Se derivan de los factores externos del sistema. |

Algunos ejemplos de Requisitos no funcionales (RNF) son:

* Cualidades en los tiempos de ejecución, usabilidad y la seguridad.
* El sistema debe ser capaz de procesar N transacciones por segundo.
* El sistema será capaz de trabajar hasta con un millón de usuarios.
* Cualidades relacionadas con la evolución del sistema, como, comprobación, extensibilidad, escalabilidad y mantenibilidad, las cuales están inmersas en la estructura del sistema de software.

**6. Definir el tipo de arquitectura**

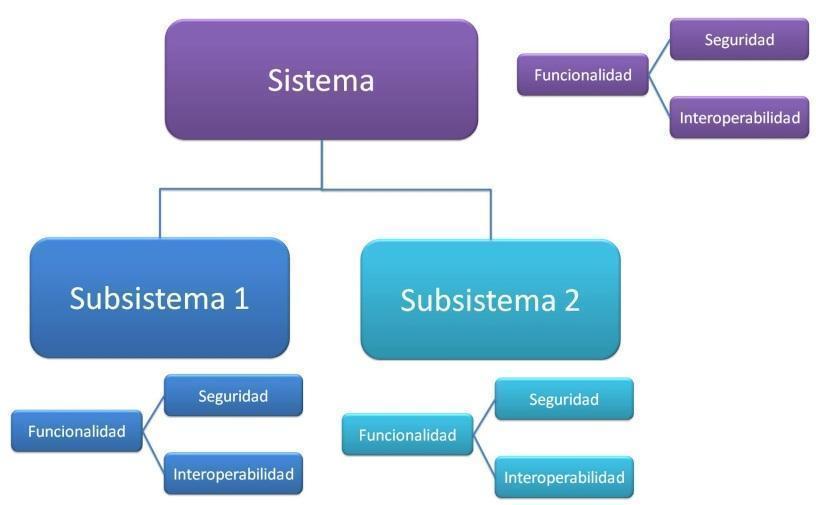
La arquitectura de *software* es también denominada arquitectura lógica. De acuerdo a Garlan y Shaw (1994), la arquitectura de *software* se refiere a la estructuración de la aplicación que generalmente se aplica al inicio del desarrollo del *software*. Por lo general se adopta un patrón de arquitectura o “arquetipo” existente, sin necesidad de inventar una nueva arquitectura de software, esto en función de sus ventajas e inconvenientes para cada proyecto. Las arquitecturas más utilizadas son:

* ***Arquitectura de*** [***descomposición modular***](https://es.wikipedia.org/wiki/Dise%C3%B1o_estructurado)

También conocida como modularización, consiste en la descomposición de un sistema en módulos más simples, con el fin de resolverlos de manera fácil y sencilla como indica la imagen. Cada sub modulo es una tarea completa la cual se programa de manera independiente, a esto se lo conoce como refinamiento por pasos:

**Figura 5**

Modelo descomposición modular



Nota. Ittgweb (2016).

| Ventajas | Pasos que se deben realizar para la descomposición modular | Cualidades de la descomposición modular |
| --- | --- | --- |
| * Claridad. * Reducción de costos. * Reutilización. | * Identificar los módulos. * Describir los módulos. * Describir la relación entre los módulos. | * Independencia funcional. * Acoplamiento. * Cohesión. * Comprensibilidad. |

* ***Arquitectura*** [***cliente-servidor***](https://es.wikipedia.org/wiki/Cliente-servidor)

De acuerdo a Garlan y Shaw (1994), este tipo de arquitectura consiste en un diseño, en el cual las tareas se reparten entre los servidores “proveedores de recursos o servicios”, y los [clientes](https://es.wikipedia.org/wiki/Cliente_(inform%C3%A1tica)) “cada uno de los usuarios”. Un cliente ejecuta peticiones a otro programa, el [servidor](https://es.wikipedia.org/wiki/Servidor) y es el que da la respuesta, en este arquetipo la capacidad de proceso está repartida entre los clientes y los servidores. Algunos ejemplos que utilizan este modelo son: [*World Wide Web*](https://es.wikipedia.org/wiki/World_Wide_Web), correo electrónico, servidor de impresión (ver figura 6).

**Figura 6**

Arquitectura cliente servidor 

Nota. SENA. (2022). Representación gráfica de la arquitectura cliente servidor.

| Ventajas | Desventajas |
| --- | --- |
| * Facilita la integración entre distintos sistemas y comparte información. * Favorece las interfases gráficas con lo cual mejora la comunicación con los usuarios. * Estructura modular lo cual facilitando la escalabilidad y permiten la integración de la información. | * Mantenimiento complejo de los sistemas ya que implica la interacción de diferentes partes de hardware y de software, distribuidas por distintos proveedores, entorpeciendo el diagnóstico de fallas. * Pocas herramientas para la administración y ajuste del desempeño de los sistemas. * El desempeño o (performance), se presentan problemas por congestión en la red, dificultad de tráfico de datos. |

* [***Arquitectura de tres niveles***](https://es.wikipedia.org/wiki/Arquitectura_de_tres_niveles)

También se la conoce como multinivel, es una arquitectura de *software* de aplicación correctamente establecida, que aísla las aplicaciones en tres niveles como se observa en el siguiente recurso (ver figura 7):

**Figura 7**

Modelo Tres Capas

| Anexo: CF01\_6\_Infografia animada\_Tres\_niveles |
| --- |

Nota. Multiniveles de la arquitectura tres capas.

| Ventajas | Desventajas |
| --- | --- |
| * Desarrollo más ágil. * Escalabilidad mejorada. * Mejor confiabilidad. * Seguridad mejorada. | * Aumenta el tráfico en la red. * Gasto de espacio de la aplicación dependencia. * A veces realiza trabajos innecesarios. |

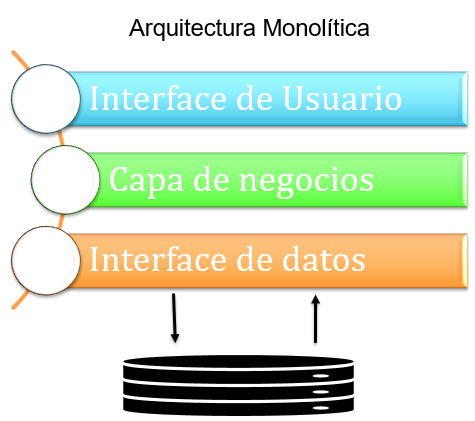
* ***Arquitecturas monolíticas***

Este tipo de arquitecturas según Garlan y Shaw (1994), se enfocan en desarrollar aplicaciones que sean autosuficiente y desarrollen toda la funcionalidad necesaria para realizar la tarea para la cual fue diseñada, sin depender de otros módulos o dependencias externas que complementen su funcionalidad.

Al inicio de la computación todos los programas eran desarrollados con este estilo arquitectónico; es solo recordar los sistemas antiguos, donde todo funcionaba en una sola computadora. Con la aparición de internet se hace posible consumir servicios externos, por lo que se crearon las arquitecturas modulares que permitieron separar el código en unidades de software cohesivas, fáciles de administrar y más manejables (ver figura 8).

**Figura 8**

*Arquitectura Monolítica*



Nota. SENA (2022). Unidades de *software* en la arquitectura monolítica.

| Características | Ventajas | Desventajas |
| --- | --- | --- |
| * Las aplicaciones autosuficientes no requieren paquetes externos para funcionar. * Realizan todas las operaciones para finiquitar una tarea. * Todo el sistema corre bajo una plataforma. * Son sitios de datos privados, cada instalación administra su propia base de datos. | * Fácil de desarrollar. * Fácil de escalar. * Pocos puntos de fallo. * Autónomo. * Performance. * Fácil de probar: | * Anclado a un Stack tecnológico. * Escalado monolítico. * Si falla, falla todo. * Es fácil perder el rumbo. * Se debe conocer todo el código para encontrar errores. |

* ***Arquitectura*** [***orientada a servicios***](https://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios_(inform%C3%A1tica)) ***(SOA)***(Service Oriented Architecture)

Garlan y Shaw (2011), plantean que SOA está enfocada a los servicios disponibles, es la conexión que une las metas de negocio con la solución de software. Un servicio es una representación lógica de una actividad de negocio, está por lo general tiene un resultado de negocio específico, como por ejemplo: comprobar estados de cuenta, obtener datos del tránsito o clima en la ciudad y comprar *tkts*. El SOA, funciona como un marco de trabajo el cual permite que las empresas unan los objetivos de negocio con la infraestructura tecnológica, permitiendo integrar los datos de sistemas que se encuentran separados, ya sea por plataforma, lenguaje de programación o base de datos (ver figura 9).

**Figura 9**

Características de la Arquitectura SOA



Nota. SENA (2022). Funcionamiento de la arquitectura orientada a servicios (SOA).

| Características | Ventajas | Desventajas |
| --- | --- | --- |
| * Agiliza los procesos para que hagan los negocios de manera más eficiente. * Facilita la adaptación al cambio, habilitando la posibilidad de implementar nuevas estrategias, conformes con la actividad de mercado. | * Aumenta la eficiencia en los procesos. * Amortiza la inversión realizada en sistemas. * Reduce costes de mantenimiento. * Fomenta la innovación orientada al desarrollo de servicios. * Simplifica el diseño, optimizando la capacidad de organización. * Optimizan el trabajo con datos y su coordinación. * Versatilidad, que posibilita que los servicios se consuman por los clientes en aplicaciones o procesos distintos de negocio. * Flexibilidad, lo cual permite su reutilización. | * SOA depende de la implementación de estándares. * No es óptima para aplicaciones con alto nivel de transferencia de datos. * Implica conocer los procesos del negocio, clasificarlos y extraer las funciones que son comunes para formar con ellas capas de servicios que serán requeridas por cualquier proceso de negocio. |

* ***Arquitectura MVC modelo vista controlador***

Es un tipo de arquitectura de *software* que separa los datos de una solución de *software*, la interfaz de usuario, y la lógica de control en tres componentes diferentes. Este modelo muy consolidado, ha demostrado su validez a lo largo del tiempo en todo tipo de aplicaciones y sobre muchos lenguajes y plataformas de desarrollo. En el gráfico a continuación se pueden observar los tres componentes claves de esta arquitectura (ver figura 6).

**Figura 6**

Arquitectura MVC

| Anexo: CF01\_6\_Grafico interactivo\_Arquitectura\_MVC |
| --- |

Nota. SENA (2022). Estructura de la arquitectura. Modelo Vista Controlador (MVC).

| Funciones | Modelo | Vista | Controlador |
| --- | --- | --- | --- |
| * Accede a la capa de almacenamiento de los datos. * Define las reglas de negocio “cómo funciona el sistema”. * Registro el número de las vistas y controladores del sistema. * Notifica a las vistas los cambios que en los datos se producen por un agente externo, ya que es un modelo activo. | * Recibe los datos del modelo y los muestra al usuario. * Lleva el registro de su controlador asociado. * Pueden dar el servicio de "Actualización ()", para que sea invocado por el controlador o por el modelo. | * Recibir los eventos de entrada   contiene reglas de gestión de eventos. |

Para ampliar la información sobre arquitectura de *software*, consultar en el material complementario el artículo “**An Introduction to Software”.**

* **Otras arquitecturas**

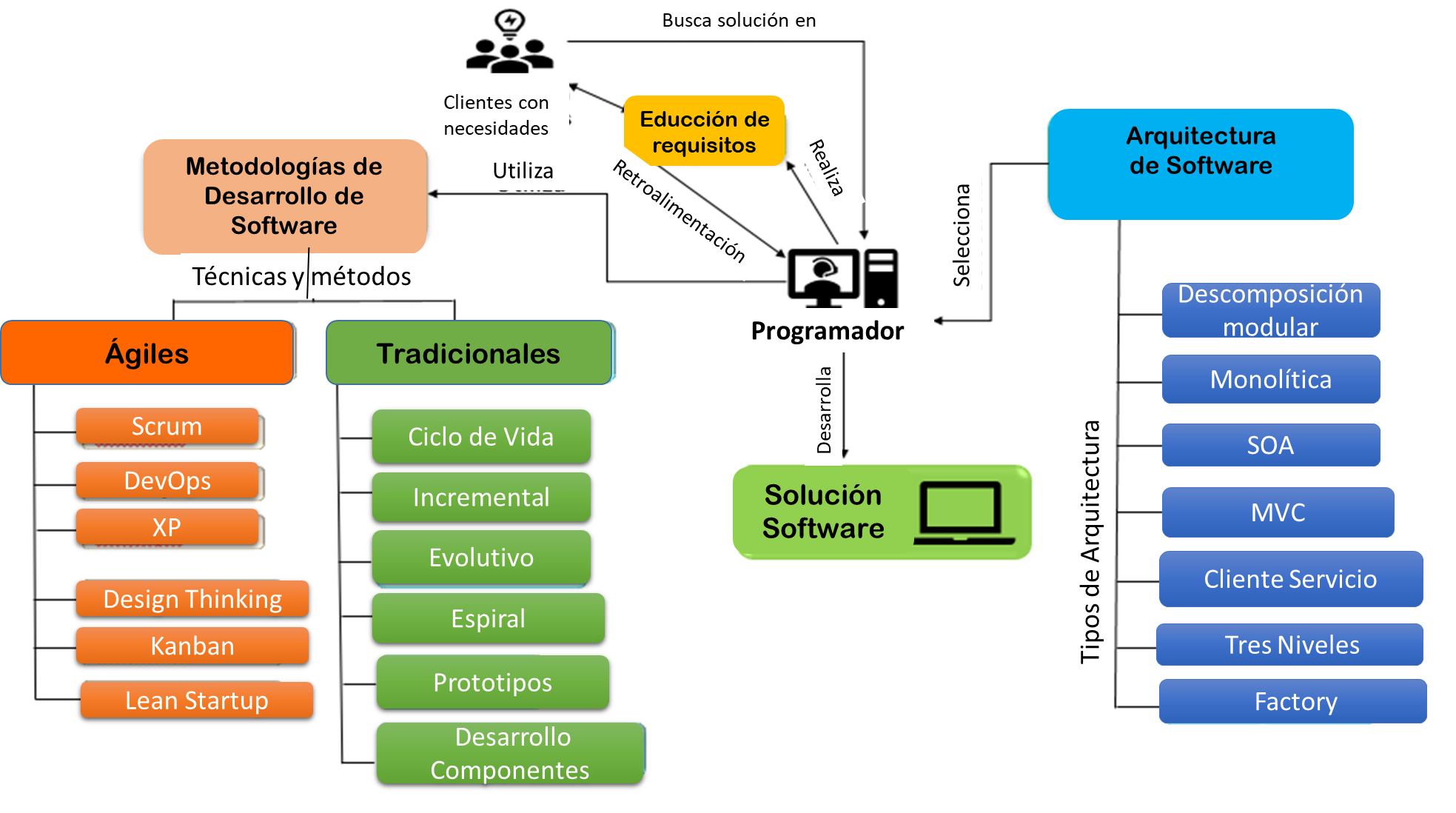
Existen otros tipos de arquitecturas aún vigentes, las cuales son menos utilizadas y poco conocidas como son:

* Arquitectura [Máquinas virtuales](https://es.wikipedia.org/wiki/Arquitectura_de_m%C3%A1quinas_virtuales_(inform%C3%A1tica)).
* Factory.
* Singleton.
* [En pipeline](https://es.wikipedia.org/wiki/Arquitectura_en_pipeline_(inform%C3%A1tica)).
* [Entre pares](https://es.wikipedia.org/wiki/Arquitectura_peer-to-peer_(inform%C3%A1tica)).
* [En pizarra](https://es.wikipedia.org/wiki/Arquitectura_en_pizarra_(inform%C3%A1tica)).
* [Orientada a servicios](https://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios_(inform%C3%A1tica)) (SOA del inglés Service-Oriented Architecture).
* [Arquitectura de microservicios](https://es.wikipedia.org/wiki/Arquitectura_de_microservicios) (MSA del inglés MicroServices Architecture). Algunos consideran que es una especialización de una forma de implementar SOA.
* [Dirigida por eventos](https://es.wikipedia.org/wiki/Arquitectura_dirigida_por_eventos).
* [Máquinas virtuales](https://es.wikipedia.org/wiki/Arquitectura_de_m%C3%A1quinas_virtuales_(inform%C3%A1tica)).

| Programadores trabajando en un proyecto de desarrollo de software Foto Premium | En el entorno de los programadores existe una confusión con los temas de arquitectura de software, patrones de *software* y estilos arquitectónicos, como se ha mencionado antes, la arquitectura de software hace referencia a la estructura con la cual se construye una aplicación al más alto nivel, organizando todos sus componentes e inter relacionándolos entre sí; los patrones arquitectónicos son soluciones recurrentes a un problema de diseño que se presentan entre componentes y su interacción, y los estilos arquitectónicos son marcos de referencia o estilos de trabajo que se componen de uno o más patrones arquitectónicos. |
| --- | --- |

1. **SÍNTESIS**

En el siguiente diagrama se resumen los aspectos principales desarrollados con este material de formación



1. **ACTIVIDADES DIDÁCTICAS (Se debe incorporar mínimo 1, máximo 2)**

| DESCRIPCIÓN DE ACTIVIDAD DIDÁCTICA | |
| --- | --- |
| Nombre de la Actividad | Conceptos básicos de arquitectura de *software* |
| Objetivo de la actividad | Afianzar algunos conceptos abordados en el componente formativo “Introducción a la Arquitectura de Software” |
| Tipo de actividad sugerida | Enlazar la definición con el término que corresponde. |
| Archivo de la actividad  (Anexo donde se describe la actividad propuesta) | Carpeta de Anexos : CF01\_Actividad didáctica 1 |

1. **MATERIAL COMPLEMENTARIO:**

Relacionar el material de apoyo o complementario de los temas abordados en este recurso. Se debe incluir al menos un par de elementos que complementen el tema del componente formativo.

| Tema | Referencia APA del Material | Tipo de material  (Video, capítulo de libro, artículo, otro) | Enlace del Recurso o  Archivo del documento o material |
| --- | --- | --- | --- |
| 1. Conocimientos en Metodologías de desarrollo de *Software* | Rivas, C. I., Corona, V. P., Gutiérrez, J. F., & Hernández, L. (2015). Metodologías actuales de desarrollo de software. *Revista de Tecnología e Innovación*, 2(5), 980-986. <https://www.ecorfan.org/bolivia/researchjournals/Tecnologia_e_innovacion/vol2num5/Tecnologia_e_Innovacion_Vol2_Num5_6.pdf> | Articulo web | <https://www.ecorfan.org/bolivia/researchjournals/Tecnologia_e_innovacion/vol2num5/Tecnologia_e_Innovacion_Vol2_Num5_6.pdf> |
| 1. Definir el tipo de arquitectura. | Garlan, D., & Shaw, M. (1994). An Introduction to Software Architecture. School of Computer Science, Carnegie Mellon University. <http://sunnyday.mit.edu/16.355/intro_softarch.pdf> | Articulo web | <http://sunnyday.mit.edu/16.355/intro_softarch.pdf> |

1. **GLOSARIO:**

| TÉRMINO | SIGNIFICADO |
| --- | --- |
| Arquetipos | Son patrones arquitectónicos en *software.* |
| Artefactos | Un artefacto es la especificación de un componente físico de información que es usado o producido por un proceso de desarrollo de *software*. |
| *Stakeholders* | Traducido al español como "grupos de interés" o "partes interesadas", son las personas, grupos u organizaciones que se ven afectadas por las acciones y decisiones de una empresa. Para identificarlos, es fundamental la realización de un mapeo. |
| Requerimientos | Es una descripción completa del comportamiento del sistema que se va a desarrollar. |

1. **REFERENCIAS BIBLIOGRÁFICAS:**

ItMadris Digital School. (2020). Qué es y para qué sirve Design Thinking. <https://www.itmadrid.com/que-es-y-para-que-sirve-design-thinking/>

Ittgweb (2016). Descomposición modular. https://ittgweb.wordpress.com/2016/05/29/descomposicion-modular/

Garlan, D., & Shaw, M. (1994). An Introduction to Software Architecture. *School of Computer Science, Carnegie Mellon University.*  <http://sunnyday.mit.edu/16.355/intro_softarch.pdf>

Lorbada, G. (2017), Diferentes metodologías ágiles. <https://lorbada.com/blog/2017/02/10/diferentes-metodologias-agiles/>

Maida, EG, Pacienzia, J. (2015). Metodologías de desarrollo de software. Tesis de Licenciatura en Sistemas y Computación. Facultad de Química e Ingeniería “Fray Rogelio Bacon”. Universidad Católica Argentina. <https://repositorio.uca.edu.ar/handle/123456789/522>

Martínez, S. (2013). La relación entre cliente, analista y programador: ¿condenados a no entenderse? https://www.mundoerp.com/blog/relacion-cliente-analista-y-programador/

Master marketing. (2020). ¿Qué es el método Lean Startup? <https://www.mastermarketing-valencia.com/marketing-digital/wp-content/uploads/sites/1/2020/06/Infograf%C3%ADa-fases-metodolog%C3%ADa-Lean-Startup.jpg>

Rivas, C. I., Corona, V. P., Gutiérrez, J. F., & Hernández, L. (2015). Metodologías actuales de desarrollo de software. *Revista de Tecnología e Innovación*, *2*(5), 980-986. <https://www.ecorfan.org/bolivia/researchjournals/Tecnologia_e_innovacion/vol2num5/Tecnologia_e_Innovacion_Vol2_Num5_6.pdf>

Sommerville, I. (2005). *Ingeniería del software*. Pearson educación.

Tenea. (2021). DevOps: Qué es y cómo beneficia a tu empresa. <https://blog.tenea.com/devops-que-es-y-como-beneficia-a-tu-empresa/>

1. **CONTROL DEL DOCUMENTO**

|  | Nombre | Cargo | Dependencia | Fecha |
| --- | --- | --- | --- | --- |
| Autor (es) | Mario Fernando Meneses Calvache | Experto temático | Regional Cauca -  [Centro Teleinformática y Producción Industrial](https://ctpisenacauca.blogspot.com/) | Marzo 2022 |
| Luz Aída Quintero Velásquez | Diseñador Instruccional | Regional Distrito Capital - Centro de Gestión Industrial | Marzo 2022 |
| Silvia Milena Sequeda Cárdenas | Asesora metodológica | Regional Distrito Capital - Centro de diseño y metrología | Marzo 2022 |
| Rafael Neftalí Lizcano Reyes | Responsable Equipo de Desarrollo Curricular | Regional Santander - Centro Industrial del Diseño y la Manufactura | Marzo 2022 |
| Jhon Jairo Rodríguez Pérez | Corrector de estilo | Regional Distrito Capital - Centro de Diseño y Metrología | Mayo 2022 |

1. **CONTROL DE CAMBIOS**

**(Diligenciar únicamente si realiza ajustes a la Unidad Temática)**

|  | Nombre | Cargo | Dependencia | Fecha | Razón del Cambio |
| --- | --- | --- | --- | --- | --- |
| Autor (es) |  |  |  |  |  |