UNIVERSIDAD TECMILENIO®

Innovación con propósito de vida.

Guía para el Profesor

Diseño y Arquitecturas de Software

ÍNDICE

Certificados

Para entender la importancia del curso del cual usted será **Facilitador**, es necesario ofrecer un contexto mayor sobre el programa de **Certificados** de la Universidad Tecmilenio, pues son parte medular del nuevo modelo educativo basado en el **aprender haciendo** y en **brindar una experiencia educativa a la medida de los alumnos**.

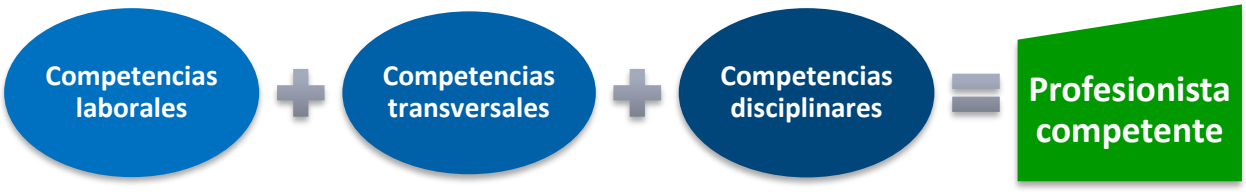
Un certificado es un **programa académico corto compuesto de varias materias,** embebido en la segunda mitad del plan de estudios de profesional, que busca desarrollar **competencias muy específicas** en el alumno y lo prepara para desempeñarse de la mejor manera en un empleo.

SABER + HACER + BIEN

Con este enfoque, buscamos en los egresados de profesional que además de **saber** (tener un conocimiento teórico), también sean **capaces de hacer** (tener la habilidad de realizar una tarea) y de **saber-hacer** (entender lo que se hace y tener la capacidad para hacerlo de la mejor forma), como se explica en este video (https://www.youtube.com/watch?v=g1maCpZXX8s):

Haz clic en la imagen

En Universidad Tecmilenio, **aprender haciendo** significa que el participante cursará **Certificados en los que desarrolla competencias disciplinares de especialidad que son valoradas por el mercado laboral**, convirtiéndose en un profesional altamente competente y elevando así su índice de empleabilidad.



La mayoría de nuestros Certificados se compone en promedio de cuatro materias, las cuales tienen un seguimiento lógico y terminan con un proyecto de gran calado y un alto nivel de complejidad (última materia). Una correcta realización del proyecto integrador demostrará el dominio de la competencia global declarada en cada certificado.

¿Certificado o certificación?

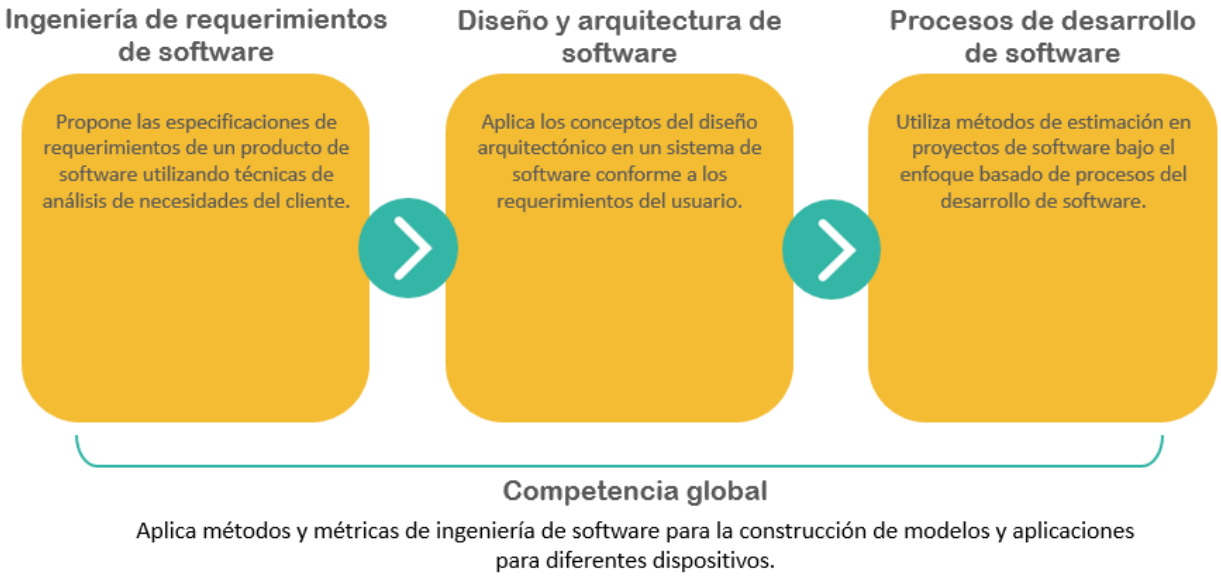
Es muy importante tener en claro que un certificado y una certificación son dos cosas distintas. Un **certificado** es un reconocimiento formal que **otorga internamente la Universidad Tecmilenio** a los estudiantes que demuestren haber aprobado las materias correspondientes, y adquirido la **competencia** global del certificado.

Por su parte, la **certificación** es también un reconocimiento, pero ésta se obtiene a través de la acreditación de un curso específico del programa académico de la Universidad y aprobando un examen de suficiencia aplicado por una **entidad acreditadora externa** (mapas mentales, idiomas, uso de software, etc.).

Su trabajo como docente facilitador de este curso es muy importante para nosotros. Gracias por aportar su conocimiento y experiencia en la impartición de este certificado. A continuación podrá revisar información detallada del curso que impartirá.

Certificado en Ingeniería de Software

El certificado de Ingeniería de Software se compone de 3 cursos más una materia de proyecto integrador, de acuerdo a la siguiente distribución:



Como se puede apreciar, este curso de **Diseño y arquitectura de software** es el segundo curso del certificado de Ingeniería de software. Por lo mismo, es importante que como **Facilitador verifique** que sus estudiantes hayan aprobado los cursos anteriores, pues de no haberlo hecho se podrá ver afectado el aprovechamiento académico de este curso.

Competencia del certificado

Al finalizar el **certificado de Ingeniería de software**, el participante deberá haber desarrollado y adquirido la siguiente competencia global, en toda su extensión:

> Aplica métodos y métricas de ingeniería de software para la construcción de modelos y aplicaciones para diferentes dispositivos.

Competencia del curso

La competencia específica que el participante habrá de obtener al aprobar satisfactoriamente el **curso de Diseño y arquitectura de software** es la siguiente, en toda su extensión:

> Aplica los conceptos del diseño arquitectónico en un sistema de software conforme a los requerimientos del usuario.

Metodología del curso

En este curso de **Diseño y arquitectura de software** se revisarán 15 temas divididos en 3 módulos.

En cada tema, el participante encontrará:

- Una breve explicación del tema que ayudará al estudiante a ampliar su conocimiento.
- Una serie de lecturas y videos obligatorios para una mejor comprensión de los temas.
- Una lista de lecturas y videos recomendados para complementar el estudio del tema.
- Una práctica no evaluable que servirá para repasar los conceptos abordados en el tema.
- Una tarea o actividad de aprendizaje (evaluable) cuyo propósito es aplicar y experimentar con los conceptos estudiados.

A lo largo del curso, el participante debe trabajar en lo siguiente:

- 15actividades
- 1 avance de evidencia
- 1 entrega final de evidencia

Actividades

Las actividades deben enviarse a través de la plataforma Blackboard en la fecha indicada.
Si las actividades se realizaron en forma física (“a mano”), deberán ser digitalizadas para enviarlas a través de dicha plataforma.

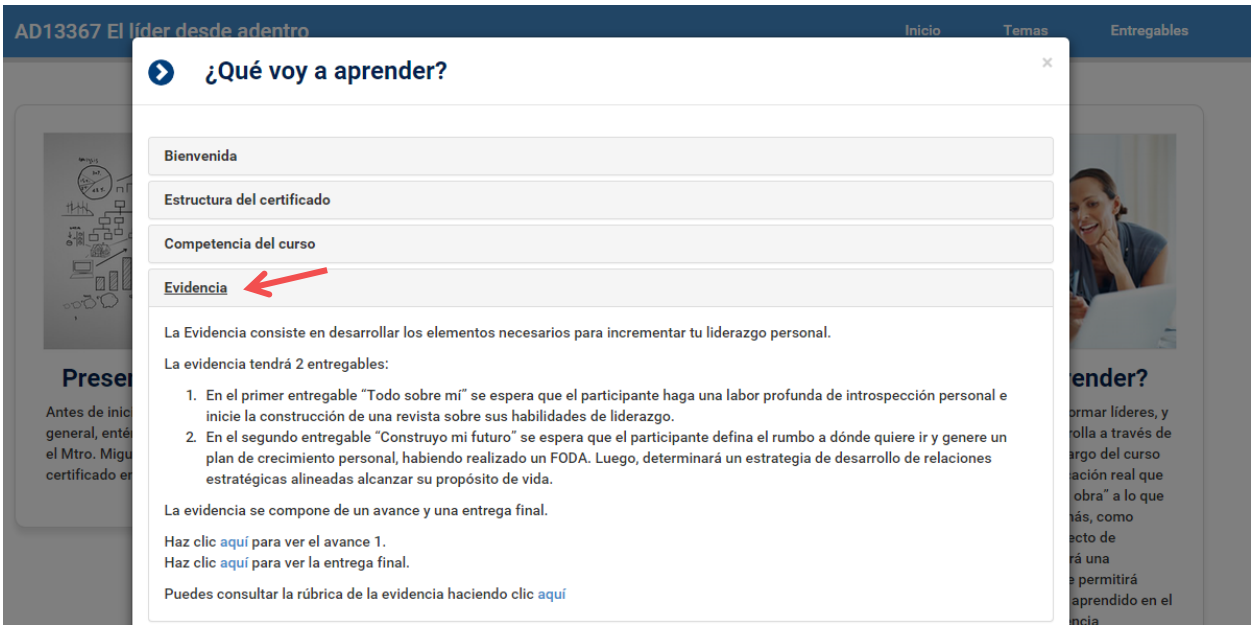
Evidencia

El proyecto final (evidencia) de este curso consiste en diseñar una aplicación de banca por internet, basado en componentes que permitan la reutilización de código, creando interfaces de usuario de

acuerdo a los principios del diseño, manteniendo en equilibrio la funcionalidad, el desempeño y estética del sistema. A través de ella el participante demostrará la capacidad de aplicar los conocimientos y habilidades que obtendrá a lo largo de los temas revisados en el curso. Es importante revisar la agenda del curso, pues la mayoría de las **evidencias requieren entregas de avances** que los alumnos tienen que realizar conforme avanza el periodo académico.

Los detalles de la evidencia pueden ser consultados en la última sección de este documento. Asimismo, tanto usted como los participantes podrán encontrar esta información dentro del curso, siguiendo alguna de estas 2 rutas:

Mi curso > Inicio > ¿Qué voy a aprender? > Evidencia, como se muestra enseguida:



O bien: **Mi curso > Inicio > Evidencia**, como se muestra enseguida:

NOTA: Es de suma importancia que **enfatice en los participantes** guardar todos los trabajos y productos que generen durante el curso (actividades, tareas, evidencias). Esto les servirá para conformar un portafolio personal de proyectos, así como para la elaboración de su proyecto integrador (último curso del certificado). Para ello, se le solicita colocar un aviso en Blackboard (sección *Announcements*), tomando como referencia el siguiente texto:

Estimado participante, recuerda guardar siempre una copia digital de todos los trabajos, actividades y evidencias que realices en tus cursos. Contar con estos documentos te será de utilidad especialmente para dos fines:

1. Conformar un portafolio personal de proyectos, que te servirá como un medio importante para enriquecer tu proyección profesional.
2. Poder elaborar el proyecto integrador de tu certificado (última materia).

Por lo tanto, asegúrate de respaldar todos tus documentos localmente en un disco duro (computadora + USB flash drive), y de preferencia también almacenarlos en la nube (servicios como Dropbox y Google Drive).

Temario

Los temas que se abordarán en este curso de certificado son los siguientes:

Tema 1 Conceptos del diseño
Tema 2 Modelo general del diseño
Tema 3 Arquitectura del software
Tema 4 Estructura general de la arquitectura del software
Tema 5 Diseño arquitectónico
Tema 6 Fundamentos de los patrones
Tema 7 Arquitecturas genéricas de software
Tema 8 Arquitectura de sistemas distribuidos
Tema 9 Arquitecturas de aplicación
Tema 10 Otros patrones
Tema 11 Fundamentos del diseño de componentes
Tema 12 Proceso del diseño por componentes
Tema 13 Interfaz de usuario
Tema 14 Análisis y diseño de interfaz de usuario
Tema 15 Diseño de WepApps

Recursos especiales

Para la impartición de este curso, se requerirá de hacer uso de las salas de PC con acceso a internet y disponer del software Microsoft Office, Gliffy (www.gliffy.com) y el software Arduino https://www.arduino.cc/en/Main/Software. Adicional, la tarjeta Arduino UNO, cable USB y foco LED.

Asimismo, el libro de texto que deberán adquirir los participantes es el siguiente:

Pressman, R. (2010). *Ingeniería de Software. Un enfoque práctico* (7ª ed.). México: McGraw Hill.
ISBN: 9786071503145
ISBN [e-Book]: 9781615022946

Las explicaciones de cada tema en Blackboard no sustituyen de ninguna forma la necesidad de comprar el libro de texto que ha sido designado para este curso. Es importante hacer hincapié en esto frente a los participantes.

Evaluación

La evaluación del curso se estructura de la siguiente manera:

Unidades	Instrumento Evaluador	Puntos
15	Actividades	65
1	Entrega primer avance evidencia	15
1	Evidencia final	20
	Total	**100 puntos**

Dichos productos se entregarán de acuerdo a la siguiente agenda, definida una vez que se hayan **validado fechas y valores con la información disponible en Servicios en Línea**:

Actividad	Temas correspondientes	Ponderación
Actividad 1	**Tema 1**	4
Actividad 2	**Tema 2**	4
Actividad 3	**Tema 3**	4

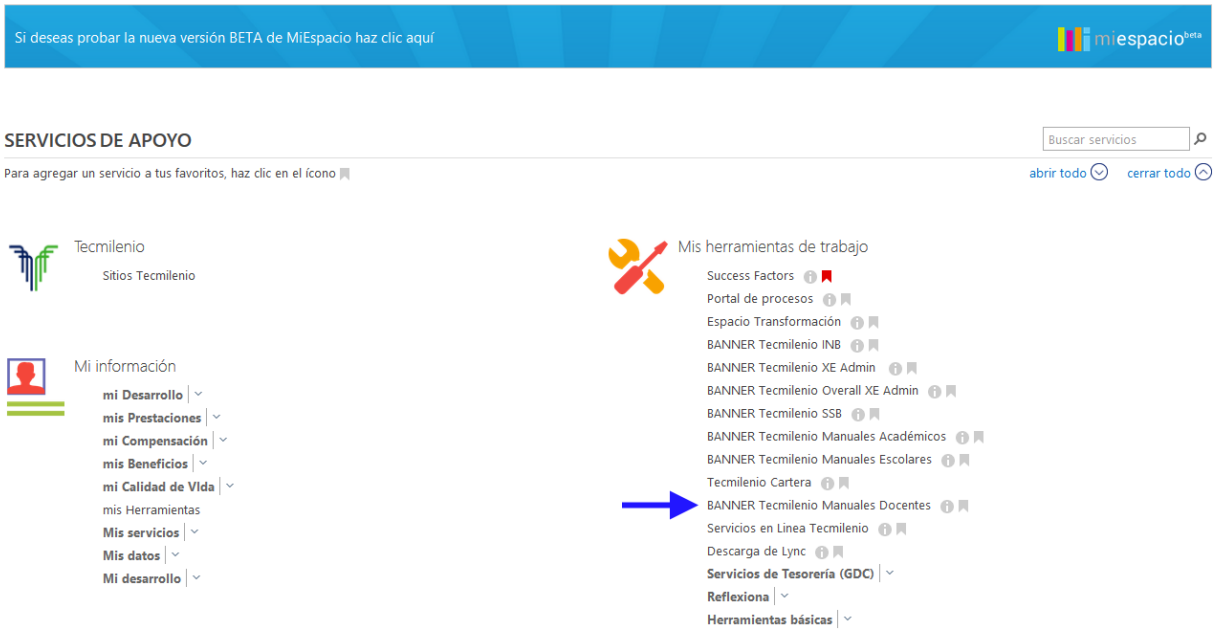
Actividad 4	**Tema 4**	4
Actividad 5	**Tema 5**	4
Actividad 6	**Tema 6**	4
Actividad 7	**Tema 7**	4
Actividad 8	**Tema 8**	4
	Avance 1 evidencia	15
Actividad 9	**Tema 9**	4
Actividad 10	**Tema 10**	4
Actividad 11	**Tema 11**	5
Actividad 12	**Tema 12**	5
Actividad 13	**Tema 13**	5
Actividad 14	**Tema 14**	5
Actividad 15	**Tema 15**	5
	Evidencia final	20
	Total	**100**

IMPORTANTE:

Estimado profesor, no olvides capturar las calificaciones de tu grupo en las fechas indicadas

Puedes ver un manual para capturar calificaciones siguiendo esta ruta en Mi espacio:
Mi espacio → Servicios → De Apoyo → BANNER Tecmilenio Manuales Docentes

Puedes ver un manual para capturar inasistencias siguiendo esta ruta en Mi espacio:
Mi espacio → Servicios → De Apoyo → BANNER Tecmilenio Manuales Docentes



Notas de enseñanza por tema

Antes de impartir el curso, por favor revise de manera general los datos y conceptos proporcionados en el mismo, con el fin de detectar y, en su caso, poder actualizar y/o enriquecer previamente la información específica al tiempo en que se está impartiendo el curso.

Un aspecto de gran importancia en el desarrollo de los temas es el involucramiento del Facilitador para propiciar que la competencia del curso se cumpla, pero también ir preparando a los participantes para que vayan desarrollando propuestas de soluciones innovadoras a problemas actuales del diseño y arquitectura de software.

Las notas de enseñanza aquí mostradas son referencia para la versión presencial y en línea, a menos que se indique lo contrario en cada tema. Puede revisarlas a continuación.

Generalidades

Para la impartición de este curso, se sugiere:

1. Revisar con tiempo la lista de entregables y la agenda en Servicios en Línea para saber en qué temas y semanas se deben realizar las actividades.
2. Revisar el manual de Blackboard para conocer las mejores formas de mantener una comunicación constante y efectiva con los estudiantes, despejar dudas y motivarlos. Puede ver un tutorial de la plataforma en esta liga: https://drive.google.com/file/d/0Bw75UcLH85hkOHVLaGo3WC1qUDA/view?usp=sharing
3. Revisar periódicamente el foro de dudas en Blackboard para resolver las preguntas e inquietudes de los alumnos acerca de las actividades y la evidencia.
4. Motivar al alumno a participar y realizar sus actividades a tiempo.
5. Proveer retroalimentación constante de las actividades que realizan los participantes.
6. Realizar un calendario y subirlo a la plataforma para que los participantes puedan visualizar de manera esquemática los temas y actividades que deberán estar revisando cada semana.
7. Recordar a los participantes que es de suma importancia que guarden tanto las actividades como la evidencia del curso en su archivo personal, pues requerirán dichos documentos para elaborar su proyecto integrador (último curso del certificado).
8. Enriquecer el curso con videos o lecturas adicionales.

Si usted imparte el **curso en modalidad online**, se recomienda también lo siguiente:

9. Realizar al menos 2 sesiones sincrónicas durante el curso con los participantes para repasar los temas revisados y resolver las diferentes dudas que puedan surgir. El Facilitador seleccionará la herramienta o plataforma que mejor le convenga: Collaborate (dentro de Blackboard), WebEx, Skype, Google Hangouts, Join.me, Zoom, etc.
   Puedes ver una **guía para organizar las sesiones sincrónicas** haciendo clic en este enlace: https://drive.google.com/file/d/0Bw75UcLH85hkdjA5bzNCNmllWW8/view?usp=sharing
10. Recordar con anuncios a los participantes acerca de las entregas de sus actividades por medio de la sección de Entrega de tareas o por correo electrónico.

Tema 1

Objetivo:
Reconocer la importancia del diseño del software, el proceso que se sigue para generar el diseño y algunos de sus conceptos generales del diseño.

Este tema contiene información introductoria del diseño de software, así como los conceptos generales que son importantes comprender para generar un diseño de calidad.

Algunos conceptos claves son:
Software, diseño, conceptos, aspecto, abstracción, modularidad, refinamiento, ocultamiento de información, revisión técnica, patrones de diseño, rediseño, refinamiento.

Notas para la enseñanza del tema:

Notas de enseñanza para el maestro (modalidad presencial)

- Los conceptos incluidos en este tema están basados en los propuestos por Roger Pressman, autor del libro de texto de este curso.
- En otras fuentes encontrará otros conceptos del diseño de software como concurrencia, polimorfismo, herencia. En este tema fueron omitidos ya que están relacionados con programación orientada a objetos que serán retomados más adelante en el curso.

- Notas de enseñanza para el tutor (modalidad en línea)
- Los conceptos incluidos en este tema están basados en los propuestos por Roger Pressman, autor del libro de texto de este curso.
- En otras fuentes encontrará otros conceptos del diseño de software como concurrencia, polimorfismo, herencia. En este tema fueron omitidos ya que están relacionados con programación orientada a objetos que serán retomados más adelante en el curso.

Notas para la actividad:

Presencial:

En el ambiente laboral tratar de explicar conceptos de desarrollo de software a personas que tienen poca experiencia en el tema suele ser común. Por lo que es importante que el participante trate de expresar términos de diseño de software en lenguaje llano en el que se pueda comunicar con diferente tipo de audiencia.

Se espera que las presentaciones sean de 10-15 min. Es importante que se genere un ambiente de crítica constructiva en donde el grupo se ponga en el papel del cliente y tratar de hacer preguntas al equipo expositor.

En Línea

En el ambiente laboral tratar de explicar conceptos de desarrollo de software a personas que tienen poca experiencia en el tema suele ser común. Por lo que es importante que el participante trate de expresar términos de diseño de software en lenguaje llano en el que se pueda comunicar con diferente tipo de audiencia.

Para esta actividad será necesario habilitar con algunos días de anticipación la función de Wikis en Blackboard. Puede ubicar esta función desde el Control Panel > Course Tools > Wikis
Para mayores detalles puede revisar el siguiente manual: https://help.blackboard.com/en-us/Learn/9.1_SP_10_and_SP_11/Instructor/050_Course_Tools/Wikis
Se espera que la propuesta y desarrollo de los términos que serán parte del glosario sea autodirigido por los mismos participantes.

Tema 2

Objetivo:

Reconocer los niveles de abstracción del modelo general del diseño para ofrecer al equipo desarrollador diferentes vistas del software.

En este tema encontrarás la descripción general de cada uno de los niveles de diseño de software, así como los diagramas de componentes y despliegue que son herramientas valiosas para detallar el software a desarrollar.

Algunos conceptos claves son:
Diagrama de componentes, diagrama de despliegue, diseño, datos, arquitectura, interfaz de usuario.

Notas para la enseñanza del tema (modalidad presencial & en línea):

El diseño arquitectónico, el diseño de componentes y el diseño de la interfaz, son abordados con mayor detalle en los temas 5, 11 y 14, respectivamente. En este tema se analizan de forma conjunta porque son parte del modelo general de diseño de software.

El participante debe distinguir cada tipo de diseño en su dimensión de abstracción y refinamiento, que le permitirá tener la idea general de lo que incluye el diseño de software.

Encontrará que algunos autores el diagrama de componentes y el diagrama de despliegue los explican en un solo diagrama. En este tema se describen de forma separada, tomado las guías de elaboración de estos diagramas a partir de la versión UML 2.0.

En la explicación del tema se toma la metáfora del proceso de construcción no porque sea un proceso rígido o estático, sino por su nivel de refinamiento, que comienza por lo general y se va detallando poco a poco. Incluso el desarrollo de software exige de una flexibilidad que no existe en el proceso de construcción una vez iniciada la obra.

Notas para la actividad:

- En esta actividad se espera que el participante proponga un diseño básico con un nivel alto de abstracción a modo de ejercicio.
- El diseño debe establecer por componentes las funcionalidades requeridas para resolver el caso.
- El listado de documentos servirá para realizar un repaso sobre los objetivos que persiguen estos documentos desde el análisis del sistema o ingeniería de requerimientos.

Tema 3

Objetivo:
Utilizar los conceptos relacionados con la arquitectura del software para tomar decisiones adecuadas en relación a la estructura básica que tendrá el desarrollo del sistema.

En este tema conocerás los conceptos básicos relacionados con la arquitectura del software que te permitirán generar un diseño arquitectónico basado en los requerimientos funcionales y no funcionales de la ingeniería de requerimientos.

Conceptos claves:
Arquitectura, software, vista, niveles de abstracción, descripciones arquitectónicas, decisiones de arquitectura.

Notas para la enseñanza del tema (modalidad en línea y presencial):

El Modelo de Kruchten de 1995 ha sido el referente de la mayoría de los trabajos sobre vistas publicados hasta este momento. Los diseños de arquitectura actuales siguen utilizando este modelo.

Existen otros modelos de vistas, sin embargo, coincidimos con Kruchten en que pueden estar contenidos en su modelo.

Es importante hacer ver al participante que la responsabilidad de generar una arquitectura adecuada no debe recaer en una sola persona a la que generalmente se le otorga el título de “Arquitecto”, sino es un trabajo en equipo que debe ser revisado y validado en un proceso de iterativo e incremental. Incluso en Internet encontrará que es una opinión generalizada en la comunidad de desarrolladores que este título no sea asignado en los proyectos, pues da la impresión que el trabajo de arquitectura solo se encuentra en una sola persona y que el programador tiene poco que ver con el proceso creativo de crear la arquitectura. Por otro lado, da la impresión de asignar un grado superior que al

resto de la organización, dando a entender que los programadores tomarían el rol de albañiles siguiendo con

Notas para la actividad:
En esta actividad se espera que el participante ofrezca diferentes vistas de un mismo diseño de software para cubrir las necesidades de información de los *stakeholders.*

Los diagramas solicitados tienen provienen del curso 1 de este Certificado, por lo que es posible que requieran realizar una consulta para recordar su construcción.

El caso descrito omite detalles generales sobre los requerimientos funcionales. El participante puede realizar supuestos sobre cómo debe funcionar el sistema.

Tema 4

Objetivo:
Reconocer los tres niveles que forman parte de la arquitectura de software y el papel que desempeñan al momento de diseñar la estructura que tendrá el software en respuesta a los requerimientos del sistema.

En este tema revisarás los 3 niveles esenciales que forman parte de la arquitectura del software y el uso que le puedes dar a cada uno de ellos en la etapa del diseño del sistema.

Conceptos claves:
Géneros, estilos, patrones, diseño, arquitectura, software.

Notas para la enseñanza del tema (modalidad presencial y en línea):
Los géneros arquitectónicos propuestos por Booch no se encuentran divididos. En este tema se propone una división por características similares a fin de facilitar su comprensión.

No existe una estructura única sobre los estilos de arquitectura. En la literatura actual encontrará que no hacen diferencia entre los estilos arquitectónicos y los patrones arquitectónicos, por lo que es necesario utilizar el propuesto por el libro de texto del curso.

Los patrones arquitectónicos son descritos con mayor detalle en el módulo 2 de este curso.
En este tema se abordan brevemente a modo de establecer la arquitectura general del software.

Pressman (2010) aborda las estructuras arquitectónicas canónicas basándose el libro de Bass, Clemens y Kazma, titulado *Software Architecture in Practice*; sin embargo, en la tercera edición del 2012 de este mismo libro ya no aparecen, razón por la cual se decidió no incluirse en este tema.

Notas para la actividad:
El participante pude buscar en Internet documentación sobre el diseño de la arquitectura aplicada a un caso y a partir de esta documentación explicar la estructura arquitectónica utilizada.

Se ofrecen 3 ejemplos, sin embargo, no son limitativos. El participante podría buscar algún otro ejemplo.

El participante podrá observar que existen diferentes formatos para documentar la arquitectura del software. Lo importante será que sea capaz de identificar diferentes estilos de documentación y diagramas UML utilizados.

Tema 5

Objetivo:
Practicar métodos de diseño arquitectónicos basados en diagramas de contexto.

En este tema revisarás dos métodos para realizar un diseño que establezca una estructura de arquitectura de software adecuada: el diseño arquitectónico basado en la funcionalidad y el diseño estructurado.

Conceptos claves:
Diseño, arquitectura, diseño estructurado, basado en funcionalidad, flujo de datos, refinamiento, arquetipos.

Notas para la enseñanza del tema:

- Los métodos descritos en este tema están basados en los diagramas de contexto generados durante la ingeniería de requerimientos que se vieron en el primer curso de este certificado.
- El objetivo fundamental de incluirlos aquí es darle continuidad a lo aprendido y seguir construyendo el aprendizaje del participante. Para ello es importante que el participante practique la teoría con diagramas de flujo ya elaborados.
- Su función como maestro/tutor será ayudarle a analizar el diseño arquitectónico que genere el participante y apoyarle durante el refinamiento.

Notas para la actividad:

- La actividad está enfocada en el diseño estructurado dado, partiendo de la construcción del diagrama de flujo de datos. Si lo cree conveniente puede mostrar el siguiente diagrama para darles una idea: BestProjectsIdea. (2013). *Supermarket Management System*. Recuperado de http://www.bestprojectsidea.com/vb-6-0-synopsis/supermarket-management-system/

- Para aquellos participantes que requieran un repaso sobre la construcción de un DFD se recomienda revisar el capítulo 7 del libro de Kendall E., Kendall, J. (2011). *Systems analysis and design* (8ª ed.). EE. UU: Prentice Hall. O bien el tema 8 del curso de Ingeniería de Requerimientos.
- Se espera un diseño sencillo en el que el participante demuestre que ha practicado los 6 pasos para construir un diseño estructurado y pueda hacer un rediseño de refinamiento.

Tema 6

Objetivo:
Analizar los conceptos básicos de los patrones de diseño y sus aplicaciones como parte del trabajo que realiza un ingeniero de software en la etapa del desarrollo de sistemas llamada “diseño”.

En este tema analizarás algunos conceptos básicos de los patrones de diseño y conocerás la importancia que tienen los patrones en el desarrollo de sistemas.

Conceptos claves:
Patrón, Reusabilidad, Antipatrón de diseño, Gang of four, Lenguajes de patrón, estructura, patrones de diseño.

Notas para la enseñanza del tema:
En este tema se abordan los patrones de diseño como un concepto necesario para establecer la arquitectura adecuada. Existen 23 patrones de diseño en el libro GoF, sin embargo, los patrones han ido creciendo conforme se resuelven problemas comunes al desarrollar aplicaciones.

Sería poco práctico solicitarle al participante conocer cada uno de los patrones. Se recomienda hacerle ver que es necesario que conozcan de su existencia y sugerirles consultarlos cada vez que se

necesite establecer un diseño y al mismo tiempo llevar un propio catálogo que abone a su experiencia profesional. Es por ello que la actividad de este tema propone un ejercicio en grupo en el que generan un catálogo de estos patrones.

Notas para la actividad:
Los patrones de diseño más comunes se encuentran mencionados en la tabla del tema 6.3. Usted puede formar equipos de tal manera que a cada participante le toquen al menos 3 patrones diferentes.

Para crear una wiki de patrones, puede hacer uso la herramienta wikis de Blackboard, se encuentra en botón Comunicación > Wikis.
Aparecerá una pantalla con el botón "Create Wiki". Le sugerimos nombrar la Wiki con el título: Patrones de diseño. Dentro de esta Wiki es posible crear otras páginas haciendo uso del botón "Create Wiki Page". Le sugerimos hacer tres páginas con los siguientes títulos: Creacional, Estructural y Comportamiento, de esa manera los participantes deberán ubicar el tipo del patrón al que corresponde.

Es recomendable que genere un botón con el nombre Wiki para que los participantes tengan un acceso directo. Lo puede realizar desde el panel de control, haciendo clic al boton [+] > Create tool link. Puede compartir el siguiente link con un video explicativo para los participantes:
Utmartin. (2011, marzo 2011). *Using Wikis - UT Martin - Blackboard 9.1 Student Tutorial*. [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=Vb_SaYOMgyI

Las participaciones registrando automáticamente en la parte de arriba de la wiki. De esa manera usted puede detectar quién ha modificado el contenido de la wiki.

Tema 7

Objetivo:
Analizar las características de los patrones arquitectónicos del software más comunes, aplicados al diseño de sistemas.

En este tema revisarás los patrones de arquitectura genéricos más comunes que puedes utilizar en el diseño de un sistema.

Conceptos claves:
Arquitectura de capas, repositorio, MVC, modelo-vista-controlador, tuberías y filtros.

Notas para la enseñanza del tema:
Los patrones arquitectónicos explicados en este tema son frecuentemente incluidos como parte de la literatura de ingeniería del software actual, por lo que el participante no tendrá problemas en complementar la explicación de cada patrón con otras fuentes.

Es importante que el participante reconozca sus características generales para llevarlo a aplicarlos a casos particulares.

Un buen ejercicio para el participante es buscar arquitecturas de software de casos reales y compararlas con alguno de los patrones incluidos en este módulo.

Notas para la actividad:
Se recomienda que una vez evaluados los trabajos presentados por los participantes se elijan algunos ejemplos que puedan ser analizados en el grupo. El autor de cada diseño puede presentarlos frente al grupo a modo de junta técnica, en la que el diseñador presenta al equipo técnico el diseño que seguirá el sistema.

Este ejercicio permite que compartir diferentes perspectivas del diseño de software que solucionan el mismo problema y da pie a intercambiar opiniones que se generen en el mismo grupo.

Tema 8

Objetivo:
Analizar las generalidades de las arquitecturas más comunes de los sistemas distribuidos, sus ventajas, desventajas y limitantes, para que puedas utilizarlas como parte de un diseño de software.

En este tema revisarás tres arquitecturas de sistemas distribuidos que podrás encontrar frecuentemente en aplicaciones empresariales.

Conceptos claves:
Cliente-servidor, *peer to peer*, entre iguales, arquitectura orientada al servicio.

Notas para la enseñanza del tema:
Las arquitecturas p2p suelen crear polémica, ya que en muchos casos son consideradas para intercambiar contenido de dudosa procedencia o que atenta con los derechos de autor.

Esta controversia está basada en la interpretación de la Ley de derechos de autor, ya que no establece con claridad si habiendo comprado legalmente un contenido no se permite compartirlo, algo que no sucede con otras obras intelectuales como libros, CD's, DVD's y que es permitido prestar o incluso vender a terceras personas.

Las arquitecturas revisadas en este tema también son consideradas como arquitecturas de hardware. Es difícil hacer la distinción ya que en la mayoría de los casos una arquitectura del software se impone a la infraestructura que lo soporta (el hardware).

Notas para la actividad:
En esta actividad los participantes pueden elegir algunos de estas marcas para describir la arquitectura de software que utilizan: Netflix, Claro Video, Itunes, Spotify, xBox online store.

Algunos de estos servicios son una combinación de las arquitecturas de software explicadas en el tema.

Existen algunos servicios P2P que han sido cerrados en respuesta a demandas por violaciones a los derechos de autor. Por ejemplo Napster, Megaupload, Pirate bay, Suprbay. Sin embargo, en España han declarado que este tipo de redes son legales.

Zetter, K. (2014). *Pirate Bay Has Been Raided and Taken Down: Here's What We Know*. Recuperado de http://www.wired.com/2014/12/pirate-bay-raided-taken-down/

Eldiario.es (2014). *Las redes P2P son legales en España: sentencia a favor de Pablo Soto*. Recuperado de http://www.eldiario.es/turing/propiedad_intelectual/Justicia-victoria-Pablo-Soto-discograficas_0_247775662.html

Tema 9

Objetivo:
Identificar las características generales de las arquitecturas de aplicación.

En este tema revisarás las arquitecturas de aplicaciones desde las más básicas como los sistemas de procesamiento de transacciones hasta llegar al procesamiento de lenguaje natural.

Conceptos claves:
Aplicaciones transaccionales, sistemas de información, procesamiento por lotes, asignación de recursos, edición basados en eventos, procesamiento de lenguaje

Notas para la enseñanza del tema:
La mayoría de las arquitecturas de sistemas de información ERP o CRM están basadas en *multi-tier*. Un buen ejercicio para los participantes consiste en pedirles investigar algunos ejemplos que pueden encontrar en internet como por ejemplo SAP, People Soft, Siebel o Microsoft Dynamics. Esta arquitectura será analizada en el tema 10 del curso.

Las arquitecturas de sistemas de procesamiento del lenguaje imponen ciertos lenguajes de programación de última generación como son Java y Python. Inclusive existen paquetes que contienen clases que pueden utilizarse para lograr sacar provecho del análisis del lenguaje natural.

Para aquellos participantes que se encuentren interesados en el avance de los sistema NLP, es recomendable motivarlos a explorar artículos generados por algunas universidades de prestigio que han formado grupos de investigación dedicados al NLP de sus departamentos de Ciencias Computacionales, como por ejemplo Stanford, Universidad de California, Universidad de Pensilvania, Universidad de Rochester, por mencionar algunas.

Notas para la actividad:
Los procesos transaccionales son sencillos de modelar ya mantienen una estructura secuencial que corresponde a lo que sucede en el proceso. El participante puede describir cualquier tipo de proceso: ya sea el registrar una venta, generar una factura, procesar una petición a un cajero automático, etc.

Por lo genera los sistemas de información gerencial presentan arquitecturas por capas o multiniveles ya que son más complejos que cualquier otro tipo de sistemas. En esta actividad el participante deberá reconocer los patrones arquitectónicos de software que hasta ahora ha analizado.

La mayor dificultad que enfrentan los sistemas de procesamiento del lenguaje radica en la interpretación que debe hacer el sistema para responder con exactitud.
Para ejemplificarlo puede revisar el siguiente video que, con una pizca de humor, muestra cómo se pone a prueba el sistema SIRI.
Stuff mom never told you. (2013, noviembre 4). *Am I Pretty or Ugly? Ask Siri.* [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=mo-5K8z_yTA

Tema 10

Objetivo:
Identificar las características de los sistemas embebidos o sistemas de tiempo real, así como los patrones que los describen y el patrón multinivel.

En este tema conocerás los patrones que describen a los sistemas embebidos y sistemas en tiempo real, así como los patrones multinivel que son frecuentemente utilizados en la industria del software para representar la complejidad de aplicaciones empresariales conocidas como ERP´s y CRM's.

Conceptos claves:
Control Ambiental, Observar y Reaccionar, Segmentación de proceso, *process pipeline*, multiniveles, *multi-tier*, embebidos.

Notas para la enseñanza del tema:

Por lo general cuando se habla de sistemas en tiempo real se piensa en sistemas que muestran información tan pronto como es originada. En el caso de sistemas embebidos, Sommeville hace una distinción importante, el tiempo real va más relacionado a la respuesta que generan los sistemas ante un estímulo, y la rapidez de respuesta está acotada a las condiciones físicas que prevalecen es este instante. Un sistema embebido podría generar una respuesta aunque no fuera inmediata e incluso así considerarse un sistema de tiempo real.

Existen otros patrones arquitectónicos que no fueron incluidos en este tema porque no aparecen tan frecuentemente en la literatura de la ingeniería de software. Entre ellos Map-reduce, Publish-subscribe y Broker. Dejamos a consideración del profesor abordarlos como material extra. Estos y otros patrones son explicados en el siguiente libro:
Bass, L., Clements, P. y Kazman, R. (2012). *Software Architecture in Practice*. EE. UU: Addison - Wesley Professional.

Notas para la actividad:
Esta práctica es muy sencilla y puede generar en los participantes curiosidad sobre qué más pueden hacer con la tarjeta Arduino. En YouTube es posible replicar proyectos cuyo código es *open source*.

Las tarjetas *Arduino Uno* cuestan alrededor de $200. Se pueden conseguir en mercado libre o en tiendas de electrónica. Le recomendamos dar las instrucciones a los participantes con tiempo para que puedan conseguir lo necesario para esta práctica.

Para esta práctica es recomendable revisar el siguiente video antes de iniciar. TechTeacher. (2011, agosto 6). *Arduino: Lesson 1 - Blinking an LED*. [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=dnPPoetX0uw

Se requiere la sala de PC para esta práctica, sugiero separarla con anticipación.

Tema 11

Objetivo:
Establecer los fundamentos del diseño de software por componentes a través de la perspectiva orientada a objetos y la tradicional.

En este tema descubrirás las diferentes perspectivas del diseño de componentes que ayudarán a pasar el trabajo elaborado en el análisis al diseño del software.

Conceptos claves:
Componentes, módulos, vista orientada a objetos, vista tradicional, clases, interfaces, control, dominio, infraestructura, principios, abierto-cerrado, sustitución de Liskov. Inversión de la dependencia, segregación la interfaz, lineamientos.

Notas para la enseñanza del tema:
Existe una tercera perspectiva para el diseño de componentes relacionada con el proceso. De forma breve se reduce a utilizar patrones de diseño ya existentes, en lugar de tener que realizar un diseño desde cero. Este tema no fue abordado dado que el módulo 2 del curso incluye la descripción de los patrones de diseño.

En este tema se abordan los principios del diseño de componentes más comunes, sin embargo, Roger Pressman incluye otros tres principios del diseño relacionados con la reutilización de código o principios de la arquitectura de empaquetamiento. Principio de equivalencia, de cierre común y reutilización común. Ponemos a su consideración abordarlos en clase. En el siguiente recurso puede también utilizar principios del diseño a través de patrones:
Martin, R. (2000). Design principles and Design Patterns. Recuperado de http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf

Notas para la actividad:
Puede utilizar el siguiente trabajo como base para realizar el diagrama de clases.
Bello, A. (2013). TFC.NET. *Aplicación de gestión escolar.* Recuperado de:
http://openaccess.uoc.edu/webapps/o2/bitstream/10609/23641/6/abello81TFC0613memoria.pdf

En el módulo 2 del curso se explicó el diagrama de despliegue.

Tema 12

Objetivo:
Reconocer las características que debe tener el diseño de software basado en componentes.

En este tema analizarás las características que debe tener un componente para poder ser reutilizado, los principios que debes aplicar para su diseño y los elementos que debe tener una biblioteca de componentes.

Conceptos claves:
Cohesión, acoplamiento, componentes, reutilización, principios, biblioteca de componentes, ingeniería de dominio.

Notas para la enseñanza del tema:
En la edición 7 del libro de Pressman (2010) aparecen otros tipos de acoplamiento: de molde, de datos, de rutina de llamada, de tipo de uso, y acoplamiento de inclusión o importación. Estos tipos ya no aparecen en la edición 8 (2015) razón por la cual no fueron incluidos en este tema.

Pressman (2010) ofrece una explicación sobre el diseño de componentes tradicionales que se basa en programación secuencial. Se decidió no incluirse en este tema dado que la mayoría de los actuales lenguajes de programación tienen el paradigma Orientado a Objetos. Dejamos a su consideración el abordar el tema como material adicional.

Notas para la actividad:
En esta actividad se espera que el participante realice un ejercicio aplicando el proceso de diseño de componentes. Se proponen las clases para adelantar el trabajo que normalmente se realiza en la etapa de análisis del ciclo de vida del desarrollo de sistemas.

Si usted considera que el planteamiento del caso no está a la altura de un reto para el participante, puede sugerir realizar un sistema para otro tipo de escenario, o bien proponer varios tipos de tal manera que en el grupo se trabaje con diferentes casos.

El proceso de diseño se encuentra descrito paso a paso en el capítulo 10 del libro de texto (Pressman, 2010), por si algún participante tiene dificultades para llevar a cabo esta actividad.

Tema 13

Objetivo:
Utilizar los conceptos del diseño de la interfaz de usuario, así como los principios del diseño para poderlos tomar como herramientas que permitan generar un producto útil para el usuario.
En este tema conocerás los conceptos y principios sobre los que se fundamenta el diseño de la interfaz de usuario.

Conceptos claves:
Interfaz, usuarios, consistencia, detectabilidad, estética, usabilidad, accesibilidad, principios, diseño.

Notas para la enseñanza del tema:
Le recomendamos revisar en clase ejemplos sobre los patrones de diseño de interfaces a través del siguiente recurso:
Talboe, A. (2012). *User Interface Design Patterns*. Recuperado de http://ui-patterns.com/patterns

Una buena práctica para los participantes es solicitarles ejemplos del diseño de interfaces de usuario que no cumplan con algún principio y realizar una crítica constructiva. Algunas preguntas pueden ser las siguientes: ¿Cuál ha sido el mejor diseño de interfaz que conoces? ¿Cuál ha sido el peor y por qué? ¿Cómo mejorarías la interfaz?

Notas para la actividad:
El diccionario de elementos que pueden utilizarse en el diseño de una interfaz debe ser extenso ya que existe una gran cantidad de elementos.

Si los participantes tienen algún problema sobre la investigación de casos en los que se haya aplicado el proceso de diseño de la interfaz de usuario, puede solicitarles una investigación de tres autores donde describan el proceso y realicen una comparación.

Los dispositivos de entrada o salida que eligen los participantes pueden ser un mouse, una pantalla o una impresora.

Tema 14

Objetivo:
Realizar un análisis adecuado que servirá de base para generar un diseño de interfaz de usuario utilizando los principios del diseño.

En este tema profundizarás las actividades relacionadas con el análisis y diseño de la interfaz de usuario, así como los principios que norman un buen diseño de software.

Conceptos claves:
Interfaz de usuario, principio, diseño, análisis de interfaz, objeto blanco, objeto fuente, objeto aplicación, tiempo de respuesta, manejo de errores, ayuda, menús, comandos.

Notas para la enseñanza del tema:
Es recomendable que el participante realice ejercicios que le ayuden a generar prototipos de interfaces de usuario de una forma ágil. Recuerde que un prototipo no necesariamente debe ser funcional, puede inclusive ser un bosquejo y servir para este propósito.

Existen algunas aplicaciones para ayudar diseñar interfaces de usuario de una manera rápida. Una de ellas es https://sketch.io/sketchpad/ o https://www.gliffy.com.

Gliffy permite utilizar algunas plantillas prefabricadas para elaborar prototipos de aplicaciones de una manera muy sencilla, y los participantes podrían aprender a utilizarla por sí mismos.

Es posible que los participantes encuentren otros principios del diseño en fuentes de información. Como sugerencia podrían generar una WIKI en Blackboard para compartir con el resto del grupo aquellos principios que consideren una buena idea para mejorar sus diseños de interfaces de usuario. Por ejemplo, en el mismo libro de texto, al final del capítulo 11, Pressman incluye los siguientes principios aplicables a la interfaz de webapps, como previsión, Ley de fit, autonomía controlada, navegación visible, flexibilidad, por mencionar algunos.

Notas para la actividad:

En esta actividad el participante puede proponer estaciones fijas para que los meseros puedan registrar las órdenes de los clientes, o bien hacer uso de aplicaciones móviles donde los meseros registren los pedidos y sean transmitidos por WIFI hasta la cocina, o bien que los mismos clientes realicen el pedido de sus órdenes directamente.

Un buen ejercicio es que cada participante presente al grupo sus diseños y establecer una mecánica de intercambio de ideas sobre los principios aplicados al diseño.

Tema 15

Objetivo:
Reconocer los elementos relacionados con un buen diseño de una aplicación web.

En este tema revisarás los elementos que permite diseñar una aplicación web de forma que pueda ser útil para el usuario.

Conceptos claves:
WebApp, contenido, navegabilidad, consistencia, diseño, aplicaciones web, seguridad, disponibilidad, escalabilidad. MVC, arquitectura

Notas para la enseñanza del tema:
Es importante reconocer que existen ciertas diferencias entre las aplicaciones web (WebApps) y las páginas web o sitios web. Mientras que las primeras están diseñadas y construidas a través de un proceso de desarrollo y pretenden cumplir con necesidades específicas de un grupo de usuarios, las páginas web suelen ser portales de información de la empresa que no incluyen funcionalidad. En este tema nos enfocaremos en aplicaciones web en lugar de sitios web, sin embargo, en ambos casos se pueden aplicar los mismos principios del diseño.

Notas para la actividad:
Esta actividad requiere la elaboración de un par de prototipos en los que se espera que el participante aplique elementos de un buen diseño.
La propuesta de los participantes debe ser innovadora y creativa.
Las preguntas, que serán parte de la encuesta de evaluación, deben redactarse de tal manera que puedan evaluar elementos del diseño, sin la necesidad de expresar directamente el concepto a evaluar. Si a una persona le preguntan su opinión sobre la “usabilidad” del prototipo, probablemente no sepa responder. Pero si le preguntan su opinión sobre las características relacionadas con la usabilidad, entonces obtendrán mejores resultados.
Es muy importante que, como parte de la evaluación de la interfaz de usuario, los participantes obtengan opiniones de personas que no se encuentren relacionadas con la clase y decidan si deberán realizar ajustes a los prototipos generados, basados en los resultados de las encuestas.

Evidencia

El participante deberá elaborar una evidencia (producto final) por medio de la cual demuestre el dominio de la competencia del curso, como elemento indispensable para conseguir la acreditación del mismo. Es decir, lo plasmado en la evidencia es aquello que buscamos que los estudiantes sean capaces de hacer bien.

Es importante insistir en que los participantes se tomen en serio la elaboración de las evidencias de sus certificados, pues con ellas pueden armar un portafolio interesante de proyectos que les servirá mucho al momento de buscar ingresar al mercado laboral.

Las instrucciones para la realización de la evidencia son las siguientes:

Tomando como base el siguiente artículo que contiene el análisis de un sistema de banca por internet, realiza lo siguiente:

AlAbdullah, F., Alshammari, F., Alnaqeib, R., Jalab, H., Zaidan, A. y Zaidan, B. (2010). *Analytical Study on Internet Banking System*. Recuperado de http://arxiv.org/pdf/1006.4559.pdf

PARTE I. (Avance 1 evidencia)

A continuación se describen los pasos para el primer avance:

1. Describe el sistema que estás analizando. Genera un antecedente y contexto.

2. Documenta los siguientes diagramas UML.

a. Casos de uso. Se espera que describas los casos de uso que servirán para describir las acciones que realizan las entidades que interactúan con el sistema.
b. Diagramas de actividad, donde se muestren la secuencia de pasos que realiza el usuario al interactuar con el sistema.
c. Diagramas de estado, en donde se describan los cambios de estado que sufre una cuenta bancaria y la cuenta de acceso de un cliente que sirve para autenticarse.
d. Diagrama de clases. Indica las clases, dependencias, relaciones de multiplicidad, herencia e interfaces necesarias del sistema.
e. Genera un diagrama de componentes en el que se detallen los módulos necesarios del sistema.
f. Crea el diagrama de despliegue, con el que se describas componentes de la infraestructura tecnológica del sistema.

Importante: revisa los criterios de evaluación de la PARTE I. (Avance 1 evidencia) en la rúbrica.

PARTE II (Evidencia final)

Describe la arquitectura de software más adecuada para el sistema.

a. Ofrece un diagrama donde se muestre la arquitectura de software.
b. Describe cuál es el género arquitectónico que aplica para este desarrollo.
c. Determina cuál es el patrón arquitectónico utilizado, justificando su respuesta.
d. Describe los requerimientos no funcionales que impactan en el diseño arquitectónico.

3. Establece un diseño de la interfaz de usuario.

a. Selecciona a los usuarios que involucrarías para realizar el análisis de la interfaz de usuario.
b. Recopila información que te ayude a diseñar una interfaz de usuario. Puedes seleccionar cualquier técnica de recopilación de información. Deberás describir los pasos que realizaste.
c. Diseña un prototipo de la interfaz de usuario de cada una de las funciones del software.
d. Crea una interfaz funcional que pueda servir para darle una idea al usuario cómo sería el sistema.
e. Valida con el usuario la interfaz. Haz los ajustes necesarios, tomando en consideración los comentarios recibidos.

Importante: revisa los criterios de evaluación de la PARTE II. (Evidencia final) en la rúbrica.

El documento deberá contener los siguientes apartados:

Análisis del sistema:
Casos de uso del sistema de banca por Internet.
- Acceder a la aplicación/salir de la aplicación
- Consultar cuenta
- Transferir fondos
- Realizar pagos
- Acceder a servicios de chequera
- Acceder a las opciones de configuración de cuenta

Diagrama de actividades de:
- Acceder a la aplicación/salir de la aplicación
- Consultar cuenta
- Transferir fondos
- Realizar pagos
- Acceder a servicios de chequera
- Acceder a las opciones de configuración de cuenta

Diagramas de estado de:
- Cuentas bancarias
- Cuenta de acceso al sistema

Diseño de software:
- Diagrama de clases
- Diagrama de componentes
- Diagrama de despliegue

Arquitectura de software
Descripción de la arquitectura del software y su justificación.

Análisis y diseño de la interfaz de usuario
- Descripción del análisis realizado para determinar los atributos de la interfaz de usuario.
- Diseño de la interfaz de usuario que incluye el prototipo no funcional y funcional del sistema.
- Descripción de las actividades realizadas para generar, validar y establecer la aceptación del usuario.

La rúbrica con la que usted deberá evaluar la evidencia final es la siguiente:

Rúbrica	Descriptores						
	Excelente	**Sobresaliente**	**Aceptable**	**Suficiente**	**Insuficiente**	**No cumple**	**Puntos totales**
Avance 1							
1. Análisis del sistema	**Equivalencia: 40 puntos** 1. Describe el sistema que está desarrollando, incluyendo un antecedente y contexto. 2. Documenta de forma completa los seis casos de uso, seis diagramas de actividad y dos diagramas de estado.	**Equivalencia: 35 puntos** 1. Describe de forma general el sistema que está desarrollando, incluyendo un antecedente y contexto. 2. Documenta de forma parcialmente completa los seis casos de uso, seis diagramas de actividad y dos diagramas de estado.	**Equivalencia: 30 puntos** 1. Describe de forma general el sistema que está desarrollando, incluyendo sólo el antecedente. 2. Documenta de forma completa al menos cinco casos de uso, cinco diagramas de actividad y dos diagramas de estado.	**Equivalencia: 20 puntos** 1. Describe de forma general el sistema que está desarrollando, incluyendo sólo el contexto. 2. Documenta de forma completa los cuatro casos de uso, cuatro diagramas de actividad y un diagrama de estado.	**Equivalencia: 10 puntos** 1. Describe de forma vaga el sistema que está desarrollando. 2. Documenta de forma completa al menos tres casos de uso, tres diagramas de actividad y un diagrama de estado.	**Equivalencia: 0 puntos** No cumple con los requisitos.	40
2. Diseño del sistema	**Equivalencia: 60 puntos** 1. Describe de forma completa y correcta los siguientes diagramas: Clases, componentes y despliegue.	**Equivalencia: 50 puntos** 1. Describe de forma parcialmente completa y correcta los siguientes diagramas: Clases, componentes y despliegue.	**Equivalencia: 40 puntos** 1. Describe vagamente los siguientes diagramas: Clases, componentes y despliegue.	**Equivalencia: 30 puntos** 1. Describe de forma completa y correcta los siguientes al menos los diagramas: Clases y componentes	**Equivalencia: 20 puntos** 1. Describe de forma completa y correcta los siguientes al menos los diagramas: Clases y despliegue	**Equivalencia: 0 puntos** No cumple con los requisitos.	60
Evidencia final							
3. Arquitectura de software	**Equivalencia: 20 puntos** 1. Describe la arquitectura de software basándose en un patrón de arquitectura. 2. Justifica la aplicación del patrón de forma adecuada.	**Equivalencia: 15 puntos** 1. Describe la arquitectura de software basándose en un patrón de arquitectura. 2. Justifica la aplicación del patrón de forma parcialmente adecuada.	**Equivalencia: 10 puntos** 1. Describe la arquitectura de software sin basarse en un patrón de arquitectura. 2. Justifica la aplicación del patrón de forma parcialmente adecuada.	**Equivalencia: 8 puntos** 1. Describe la arquitectura de software no aplicable para el caso propuesto. 2. Justifica la aplicación del patrón de forma parcialmente adecuada.	**Equivalencia: 6 puntos** 1. Describe la arquitectura de software sin basarse en un patrón de arquitectura. 2. No justifica la aplicación del patrón de arquitectura.	**Equivalencia: 0 puntos** No cumple con los requisitos.	20
	Equivalencia: 30 puntos	**Equivalencia: 25 puntos**	**Equivalencia: 20 puntos**	**Equivalencia: 15 puntos**	**Equivalencia: 10 puntos**	**Equivalencia: 0 puntos**	

4. Análisis de interfaz de usuario	1. Describe de forma completa el perfil de usuarios adecuados para el análisis de la interfaz. 2. Describe de forma completa y adecuada la técnica de recopilación de información. 3. Lista al menos 10 atributos asequibles de calidad del diseño de la interfaz.	1. Describe de forma parcialmente adecuada el perfil de usuarios adecuados para el análisis de la interfaz. 2. Describe de forma parcialmente adecuada la técnica de recopilación de información. 3. Lista al menos ocho atributos asequibles de calidad del diseño de la interfaz.	1. Describe vagamente el perfil de usuarios para el análisis de la interfaz. 2. Describe de forma general la técnica de recopilación de información. 3. Lista al menos seis atributos asequibles de calidad del diseño de la interfaz.	1. Describe el perfil de usuarios para el análisis de la interfaz sin relación con la aplicación web. 3. Describe de forma general la técnica de recopilación de información. 3. Lista al menos cuatro atributos asequibles de calidad del diseño de la interfaz.	1. Describe el perfil de usuarios para el análisis de la interfaz sin relación con la aplicación web. 2. No incluye la descripción de la técnica de recopilación de información. 3. Lista al menos dos atributos asequibles de calidad del diseño de la interfaz.	No cumple con los requisitos.	30
5. Diseño de la interfaz de usuario	**Equivalencia: 50 puntos** 1. Crea un bosquejo de la interfaz de usuario que puede validar el usuario apegándose a los principios del diseño. 2. Crea un prototipo funcional de acuerdo a las especificaciones del caso. 3. Describe las actividades realizadas para validar el diseño de la interfaz y el proceso de aceptación.	**Equivalencia: 40 puntos** 1. Crea un bosquejo de la interfaz de usuario que puede validar el usuario apegándose a algunos principios del diseño. 2. Crea un prototipo parcialmente funcional de acuerdo a las especificaciones del caso. 3. Describe algunas de las actividades realizadas para validar el diseño de la interfaz y el proceso de aceptación.	**Equivalencia: 30 puntos** 1. Crea un bosquejo de la interfaz de usuario que puede validar el usuario, sin embargo no es claro el apego a los principios del diseño. 2. Crea un prototipo parcialmente funcional de acuerdo a las especificaciones del caso. 3. Describe algunas de las actividades realizadas para validar el diseño de la interfaz y el proceso de aceptación.	**Equivalencia: 20 puntos** 1. Crea un bosquejo de la interfaz de usuario que puede validar el usuario, sin apegarse a los principios del diseño. 2. Crea un prototipo no funcional y parcialmente de acuerdo a las especificaciones del caso. 3. Describe vagamente algunas de las actividades realizadas para validar el diseño de la interfaz y el proceso de aceptación.	**Equivalencia: 10 puntos** 1. Crea un bosquejo de la interfaz con errores en cuanto a la aplicación de los principios del diseño. 3. Crea un prototipo no funcional sin tener en cuenta las especificaciones del caso. 3. No describen las actividades realizadas para validar el diseño de la interfaz y el proceso de aceptación.	**Equivalencia: 0 puntos** No cumple con los requisitos.	50