



Carrera **Analista Programador Computacional**

Exp 1 – Semana 1

Arquitectura ASY4231

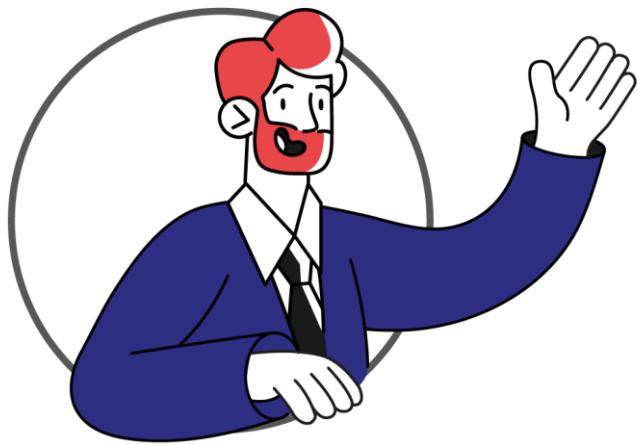
Guía de Aprendizaje

Índice

Introducción a la semana.....	4
Resultado de aprendizaje	6
El estudiante será capaz de:	6
Indicador de logro:.....	6
Conceptos relevantes	6
Preguntas activadoras.....	7
Actividad	7
Que es arquitectura de Software	8
Rol del arquitecto.....	8
Requisitos funcionales.....	10
¿Qué es un requisito funcional?	10
Características de los requerimientos:	10
Técnicas para relevar requisitos funcionales.....	11
Requisitos no funcionales.....	15
Normas para requisitos no funcionales o atributos de calidad.	16
Planillas de requisitos funcionales y no funcionales	18
Especificación de requisitos de software	19
Conceptos de Calidad	22
Atributos de calidad	23
Métricas de producto	23
Métricas de Proceso.....	23
Adecuación funcional	25
Eficiencia de desempeño	25
Compatibilidad.....	26
Usabilidad.....	26
Fiabilidad	27

Seguridad	27
Mantenibilidad	28
Portabilidad	29
Los Stakeholders.....	30
Escenarios de calidad.....	31
Documentando los escenarios de calidad.....	33
Tips y recomendaciones.....	37
Tips para obtener requisitos funcionales.....	37
Otros tips	38
¿Cómo hacer un modelo arquitectónico?.....	40
¿Cómo identificar los atributos de calidad?	41
Cierre de la semana	42
Referencias	43
Lecturas de la semana	43
Apunte	44

Introducción a la semana



calidad.

En esta primera semana iniciaremos el viaje hacia la construcción de la arquitectura del software, para ello introduciremos conceptos como rol de Arquitecto, modelos arquitectónicos, escenarios de calidad y atributos de calidad, información que se desprenderá del análisis de los requisitos que el cliente entrega para desarrollar un producto que tenga un balance entre funcionalidad y

El concepto de arquitectura de software se refiere a la estructuración del sistema que, idealmente, se crea en etapas tempranas del desarrollo. Esta estructuración representa un diseño de alto nivel del sistema que tiene dos propósitos primarios: satisfacer los atributos de calidad (desempeño, seguridad, modificabilidad), y servir como guía en el desarrollo.

del desarrollo puede limitar severamente el que el producto final satisfaga las necesidades de los clientes. Además, el costo de las correcciones relacionadas con problemas en la arquitectura es muy elevado. Es así que la arquitectura de software juega un papel fundamental dentro del desarrollo.

Como primer paso, formarán un grupo de trabajo de un máximo de 3 integrantes, los cuales trabajarán juntos durante el curso, posterior a ello, seleccionaran un proyecto (se adjuntan como anexo 3 proyectos para puedan seleccionar), el cual será utilizado durante el curso, iniciaremos el proceso descubriendo los requisitos funcionales y no funcionales, los que darán forma a la solución que desarrollaremos para el cliente, y serán documentados en un documento Excel llamado “Requerimientos simplificados”.

En una segunda instancia, vamos a generar los escenarios de calidad, los cuales nos harán reflexionar sobre como nuestro futuro sistema/software, se enfrentará a los ambientes

productivos y cómo vamos a regular algunos elementos como la seguridad, rendimiento, compatibilidad, entre otros, esto lo documentarán en el archivo Word “Plantillas escenarios de calidad”.

Para formalizar la etapa y finalizar la semana, llenarán el documento E.R.S (especificación de requisitos de software), documento estándar en el desarrollo de software, el cual contiene información sobre la problemática, objetivos, alcance, requisitos funcionales/no funcionales, interfaces del software (hardware, software, comunicaciones), características de los usuarios que operarán el software, entre otros tópicos.

Resultado de aprendizaje

El estudiante será capaz de:

RA1. Diseña un modelo arquitectónico para soportar la solución de acuerdo a su factibilidad, a los estándares de la industria y considera los riesgos de la solución propuesta.

Indicador de logro:

IL1. Selecciona el modelo arquitectónico, de acuerdo a los requerimientos funcionales y no funcionales de la organización.

IL2. Identifica los atributos de calidad a partir de los requisitos no funcionales y necesidades de los stakeholders.

Conceptos relevantes

	Requisitos del cliente	Alcance	Solución
	Funcionalidades	No funcionalidades	Riesgo informático
	Atributos de calidad	Escenarios	Satisfacción del cliente

Preguntas activadoras

- ¿Para la correcta construcción de una solución computacional, solo la construimos??
- ¿Serán los modelos arquitectónicos, las herramientas correctas para planificar un buen software y con la calidad que el cliente requiere?
- ¿Cuál es la definición "Oficial" de la Arquitectura de software según la IEEE Std 1471 - 2000?
- ¿Qué sabemos de la arquitectura del sistema?
- ¿Qué es un software?

Actividad

Descripción de la actividad

En esta primera semana realizarán una actividad formativa de manera grupal a través de un encargo llamada "Conociendo el modelo arquitectónico" donde a partir de un proyecto revisarán los requisitos funcionales y no funcionales, el ERS para poder reconocer los atributos de calidad involucrados en desarrollo de la solución y propuestos en los escenarios de calidad.

Que es arquitectura de Software

La arquitectura de software es un conjunto de prácticas que responden a la organización y planificación teóricas, que permiten la construcción de un software, la cual permite en forma ordenada, conectar cada componente entre sí, así generar un modelo que puede ser usado en forma genérica para distintos tipos de software.



Importante

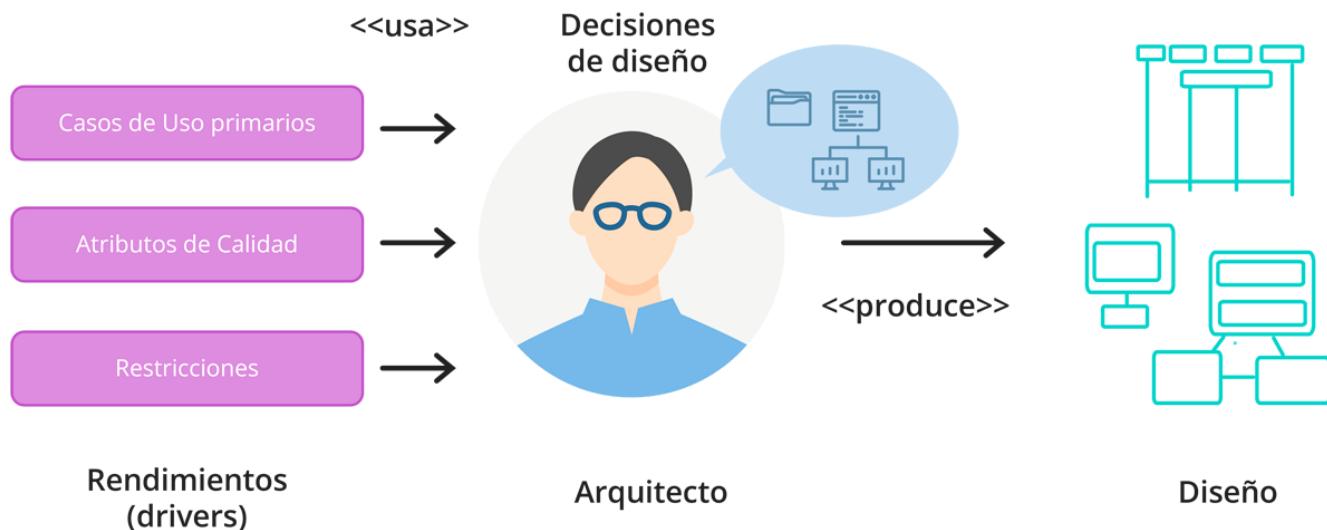
De acuerdo al Software Engineering Institute (SEI), la Arquitectura de Software se refiere a “las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos”. [SEI].

Rol del arquitecto

Quien desarrolla la arquitectura de software, es el arquitecto de software, profesional que tiene la misión y responsabilidad global de diseñar el modelo arquitectónico y las principales decisiones técnicas, éste, debe considerar desde los requisitos del cliente, tomando en consideración los atributos de calidad, los cuales son referidas a como el software se desempeñará cuando esté construido y que responderá a como se desempeñará en ambiente productivo final, por ello deberá considerar la infraestructura, seguridad, rendimiento, entre otros atributos que se verán más adelante.

Figura 1:

Diseño de arquitectura.



Nota: Durante el diseño de la arquitectura, el **arquitecto** toma como entrada los **requerimientos que influyen en la arquitectura (drivers)** y produce un **diseño arquitectónico**.

Algunas funciones del arquitecto:

- Revisa las **necesidades del negocio** junto con los **requerimientos no funcionales**, y los **relaciona con la solución** que se necesita implementar para cubrirlas.
- Se **asegura** de la **calidad de la solución** y de su **mantenimiento en el tiempo**.
- Toma **decisiones de diseño** de alto nivel.
- Dicta **estándares técnicos**, de **codificación**, **herramientas** y **plataformas**
- **Seleccionar las tecnologías**
- **Definir** y **evaluar la arquitectura**
- Entender e implementar los **atributos de calidad**

Requisitos funcionales

¿Qué es un requisito funcional?

Los (RF) definen las funciones que el sistema será capaz de realizar, describen las variaciones que producen por interacciones de usuarios, a través de entradas y salidas. A medida que avanza el desarrollo de software se transforman en algoritmos, lógica y gran parte del código del sistema.

Características de los requerimientos:

- Especificados por escrito
- Posibles de probar y verificar
- Concisos
- Completos
- Consistentes
- No ambiguos
- Fácil de modificar
- Trazabilidad



Importante

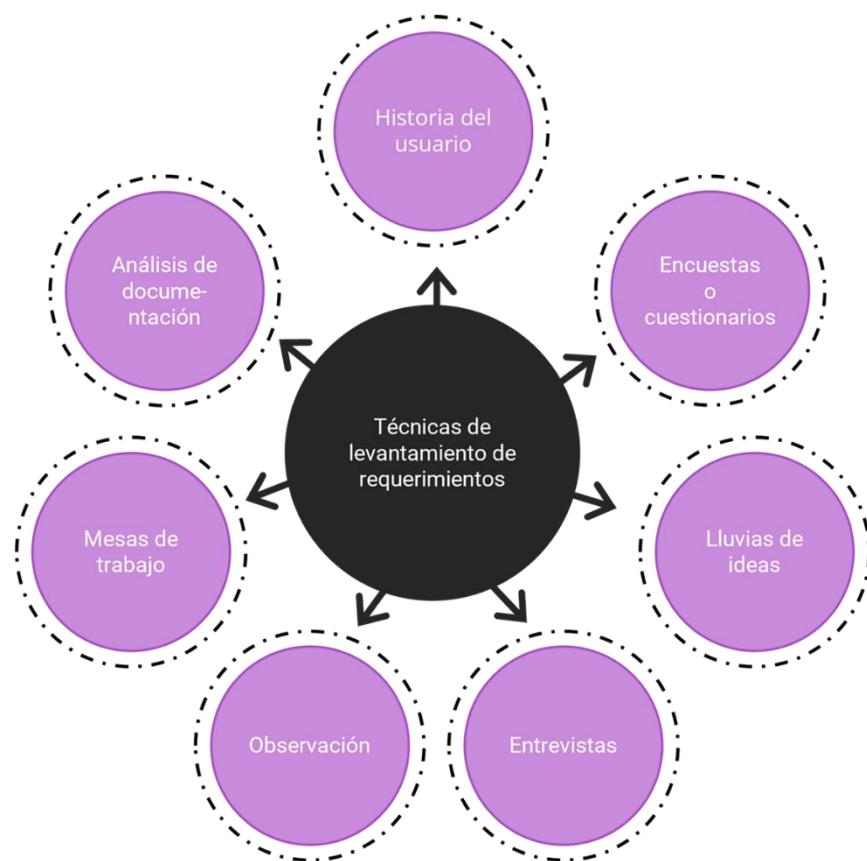
La importancia de los requerimientos en el desarrollo del sistema es la de que cada uno de ellos representa una condición o necesidad que ha sido identificada como necesaria para resolver un problema o alcanzar un objetivo dentro del modelo de negocio en el que funciona la organización que requiere que el software sea implementando, Describen el ¿Qué? hace el sistema [SEI].

Técnicas para relevar requisitos funcionales

En la figura 2 podrán encontrar un gráfico donde se indican 7 actividades que nos ayudarán a obtener los requisitos del sistema/software.

Figura 2:

Técnicas de levantamiento de requerimientos.



Nota: Técnicas de levantamiento de requerimientos; Historia del usuario, Encuestas o cuestionarios, Lluvia de ideas, Entrevistas, Observación, Mesas de trabajo y Análisis de documentación.

Análisis de documentación

Consiste en obtener la información sobre los requerimientos funcionales y requerimientos no funcionales de software a partir de documentos que ya están elaborados. Es útil cuando los expertos en la materia no están disponibles para ser entrevistados o ya no forman parte de la organización. Utiliza la documentación que sea relevante al requerimiento que se está levantando.

Ejemplos de documentación: Planes de negocio, actas de constitución de proyecto, reglas de negocio, contratos, definiciones de alcance, memorándums, correos electrónicos, documentos de entrenamiento, entre otros.

Observación

Consiste en estudiar el entorno de trabajo de los usuarios, clientes e interesados de proyecto (Stakeholders). Es una técnica útil cuando se está documentando la situación actual de procesos de negocio. Puede ser de dos tipos, pasiva o activa.

En observación pasiva, el observador no hace preguntas, limitándose solo a tomar notas y a no interferir en el desempeño normal de las operaciones.

En observación activa, el observador puede conversar con el usuario.

Entrevistas

Se realizan con los usuarios o interesados clave. Direccionan al usuario hacia aspectos específicos del requerimiento a levantar.

Son útiles para obtener y documentar información detallada sobre los requerimientos y sus niveles de granularidad.

Pueden ser entrevistas formales o informales.

- La clave es mantenerse enfocado en los objetivos de la entrevista.
- Las preguntas abiertas son útiles para identificar información faltante.
- Las preguntas cerradas son útiles para confirmar y validar información.

- El éxito de las entrevistas depende del grado de conocimiento del entrevistador y entrevistado, disposición del entrevistado de suministrar información, buena documentación de la discusión y en definitiva de una buena relación entre las partes.

Encuestas o cuestionarios

Es una técnica útil para **recopilar eficientemente los requerimientos de muchas personas**.

La clave para el éxito es que tengan un propósito y audiencia claramente definida, establecer fechas topes para llenar la encuesta, con preguntas claras y concisas.

Deben **enfocarse en los objetivos de negocio que se necesitan identificar**.

Pueden apoyarse con **entrevistas** de seguimiento con usuarios individuales.

Pueden contener tanto **preguntas cerradas** como **preguntas abiertas**

Mesas de trabajo (Workshops)

Es una técnica efectiva para obtener información rápidamente de varias personas. Es recomendable tener una agenda predefinida y preseleccionar a los participantes, siguiendo buenas prácticas para reuniones efectivas.

Se puede utilizar un facilitador neutral y un transcriptor (que no sea el mismo facilitador).

Se puede utilizar un material común sobre el cual enfocar la atención y conversar, por ejemplo, una presentación con un desglose del proceso que se está estudiando o un flujoograma.

Se pueden combinar con otras técnicas como pueden ser las entrevistas y cuestionarios.

Lluvia de ideas (BrainStorming)

Es una sesión de trabajo estructurada orientada para obtener la mayor cantidad de ideas posibles. Es recomendable limitarlas en el tiempo, utilizar ayudas visuales y designar un

facilitador. Las reglas son importantes, por ejemplo, los criterios para evaluar ideas y asignarles un puntaje, no permitir las críticas a las ideas y limitar el tiempo de discusión.

En una primera fase, se deben identificar la mayor cantidad de ideas, para luego evaluarlas.

Todas las ideas deben ser consideradas y deben limitarse que una idea se le ahogue o critique antes de tener tiempo de desarrollarla.

Historia del usuario (User Story)

Las historias de usuario son una aproximación simple al levantamiento de requerimientos de software, en la cual la conversación pasa a ser más importante que la formalización de requerimientos escritos. Es recomendable que sean escritas por el mismo cliente o interesado (con apoyo del facilitador si es necesario), con énfasis en las funcionalidades que el sistema deberá realizar. Al redactar una historia de usuario deben tenerse en cuenta describir el Rol, la funcionalidad y el resultado esperado de la aplicación en una frase corta. Las historias de usuario son una de las técnicas más difundidas para levantar requerimientos de software en metodologías ágiles.

Importante

Como dato importante, en la gestión de proyectos tradicionales habitualmente hablamos de “requisitos de alto nivel” y “requisitos funcionales”, en gestión de proyectos ágiles, es más común hablar de “Épicas e Historias de usuario”. El concepto es el mismo, son las necesidades del cliente



Requisitos no funcionales

Los **requisitos no funcionales** o **atributos de calidad** son las **características que tendrá el software para desempeñar las funcionalidades durante la operación**, es decir indican los criterios con que evaluaremos el comportamiento del software, a través de métricas. Algunos atributos de calidad son Seguridad, Fiabilidad, Usabilidad, Portabilidad, Compatibilidad, entre otros.

Ejemplo: El cliente solicita un formulario para ingresar los nombres del cliente, el cliente espera que puedan visualizarlo en distintos dispositivos.

En este ejemplo “un formulario para ingresar nombres de cliente”, corresponde al requisito funcional, y “que puedan visualizarlo en distintos dispositivos”, corresponde a un requisito no funcional o atributo de calidad, en este caso corresponde el atributo **Portabilidad**.

Los (RNF) tienen que ver con características que de una u otra forma puedan limitar el sistema, como, por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, etc.

Se pueden clasificar en:

- **Atributos de calidad:** características que influyen directamente en la calidad del sistema.
- **Restricciones:** limitaciones impuestas al sistema, recursos, tiempo, presupuesto.



Importante

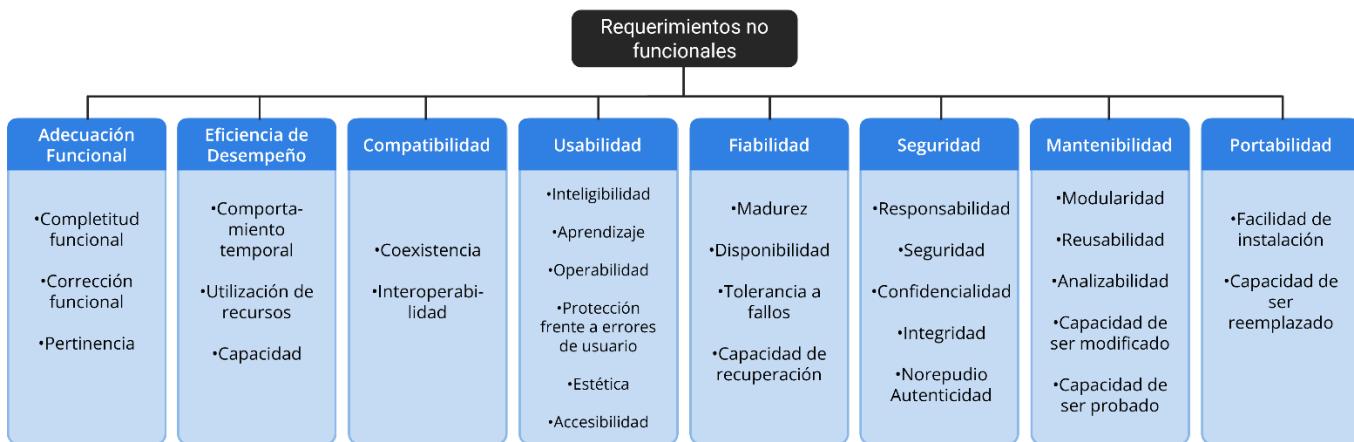
Como dato importante, los requisitos no funcionales pueden ser enlazados a los atributos de calidad, los cuales, dependiendo de su autor, varían en cantidad y organización, pero no en su característica principal, ejemplo de ellos son Seguridad, Fiabilidad, Usabilidad, entre otros, definidos en ISO/IEC 25010. Describen ¿cómo? Lo hace el sistema

Normas para requisitos no funcionales o atributos de calidad.

Los atributos de calidad según la ISO/IEC 25010 son aspectos que de una u otra manera se relacionan con la calidad del sistema, en la figura 3, podemos visualizar los atributos de calidad y las sub características (métricas) según la norma ISO 25010.

Figura 3:

Organigrama de la norma ISO 25010

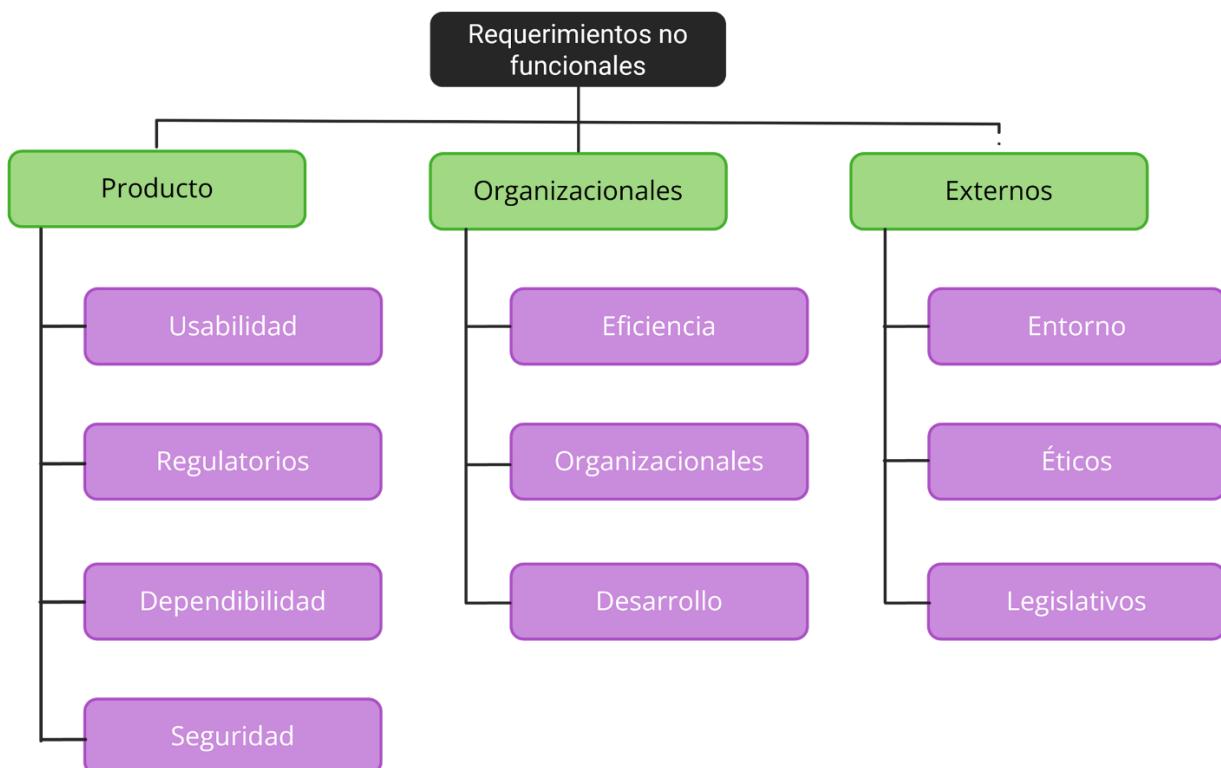


Nota: La calidad del producto de software tiene las siguientes subcaracterísticas: **Adecuación Funcional**: referente a completitud, corrección y pertinencia funcional. **Eficiencia de desempeño**: referente a medir comportamiento temporal, utilización de recursos, capacidad. **Compatibilidad**: referente a medir la coexistencia, interoperabilidad. **Usabilidad**: referente a medir capacidad para reconocer su adecuación, capacidad de aprendizaje, capacidad para ser usado, protección contra errores de usuario, estética de la interfaz de usuario, accesibilidad. **Fiabilidad**: referente a medir madurez, disponibilidad, tolerancia a fallos, capacidad de recuperación. **Seguridad**: referente a medir confidencialidad, integridad, no repudio, responsabilidad y autenticidad. **Mantenibilidad**: referente a medir modularidad, reusabilidad, analizabilidad, capacidad para ser modificado, capacidad para ser probado. **Portabilidad**: referente a medir adaptabilidad, capacidad para ser instalado y capacidad para ser remplazado.

Ian Somerville presenta también una clasificación de los requerimientos no funcionales de acuerdo con los requerimientos del producto, organizacionales y externos.

Figura 4:

Requerimientos no funcionales.



Nota: De los requerimientos no funcionales se despliegan 3 elementos: **Producto** como usabilidad, eficiencia, dependibilidad y seguridad; **Organizacionales** como entorno, organizacionales, desarrollo; **Externos** como regulatorios, éticos y legislativos.

Planillas de requisitos funcionales y no funcionales

El cliente ha entregado sus necesidades a través reuniones, documentación, visitas, etc., estas se deben documentar, pero deben ser especificadas, es decir, se deben transcribir a un lenguaje sencillo, específico y que sea entendible por todas las personas que se relacionan con el proyecto, en especial el cliente quien deberá validar que lo escrito corresponde a la funcionalidad solicitada, por otro lado el desarrollador quien deberá codificar dicha especificación y convertirla en componentes sistémicos.

Estos requerimientos deben poder ser trazables, es decir, podemos encontrarlos a lo largo de la arquitectura, como por ejemplo en las especificaciones de requisitos funcionales, diagramas de clases, diagramas de actividad, componentes, planes de prueba, etc.

A continuación, se adjunta una tabla que ayudará a documentar la planilla de requerimientos simplificado.

Tabla 1:

Tabla de apoyo a la documentación de la planilla de requerimientos simplificado.

Ítem	Descripción
[R-Nº]	Código de identificación de mayor nivel definido para el requisito. Puede definirse con números, por ejemplo 001, 002, 003, y así sucesivamente.
[Nombre del Requerimiento]	Título del requerimiento
"Tipo Requerimiento [Funcional, No Funcional]"	Indicación si es funcional o no funcional.
Clasificación	Funcional de usuario, Funcional de sistema , No funcional de producto, No funcional de la Organización, No funcional Externos
Actores Relacionados	Quienes usarán la funcionalidad
[Descripción corta del requerimiento]	Se proporciona una descripción de que comprende o en qué consiste el requisito. La descripción del requisito depende del tipo que sea, por ejemplo, requisitos del negocio, requisitos de los interesados, requisitos funcionales, requisitos no

	funcionales, requisitos del proyecto o requisitos del producto (solución).
Estado	Puede ser solicitado, aprobado, asignado, completado, cancelado, diferido, aceptado, entre otros.
Criterio de Aceptación	Lista los criterios de aceptación, una lista de puntos o condiciones específicas que deben cumplirse para poder registrar que el requisito ha sido satisfecho cuando se implemente en el Sistema.

Nota. Tabla que muestra los documentos que se requieren para la planilla de requerimientos simplificados.

Especificación de requisitos de software

La especificación de requisitos es una técnica para transcribir las necesidades de los clientes para un sistema específico, en texto entendible, resumido y específico que entrega el comportamiento del sistema que se desarrollará, incluyendo las interacciones con los usuarios finales.

En este siguiente capítulo nombraremos las secciones más importantes del documento E.R.S que deberán llenar en base a la información del proyecto que han seleccionado.

Tabla 2:

Tabla de **secciones más importantes del documento E.R.S.**

Requisito	Descripción
1. Introducción	En esta sección se proporcionará una introducción a todo el documento de Especificación de Requisitos Software (ERS). Consta de varias subsecciones: propósito, ámbito del sistema, definiciones, referencias y visión general del documento.
1.1. Propósito	En esta subsección se definirá el propósito del documento ERS y se especificará a quién va dirigido el documento.
1.2. Ámbito del Sistema	Nombre del sistema, que hará y que no hará, beneficios, objetivos y metas.
1.3. Definiciones, Acrónimos y Abreviaturas	En esta subsección se definirán todos los términos, acrónimos y abreviaturas utilizadas en la ERS.
1.4. Referencias	En esta subsección se mostrará una lista completa de todos los documentos referenciados en la ERS
1.5. Visión General del Documento	En esta subsección se describe brevemente los contenidos y la organización del resto de la ERS.
2. Descripción General	En esta sección se describen todos aquellos factores que afectan al producto y a sus requisitos
2.1. Perspectiva del Producto	Esta subsección debe relacionar el futuro sistema (producto software) con otros productos. Si el producto es totalmente independiente de otros productos, también debe especificarse aquí.
2.2. Funciones del Producto	En esta subsección de la ERS se mostrará un resumen, a grandes rasgos, de las funciones del futuro sistema.
2.3. Características de los Usuarios	Esta subsección describirá las características generales de los usuarios del producto, incluyendo nivel educacional, experiencia y experiencia técnica. Además, debes definir los Tipos de Usuarios con sus perfiles.

2.4. Restricciones	Esta subsección describirá aquellas limitaciones que se imponen sobre los desarrolladores del producto
2.5. Suposiciones y Dependencias	Esta subsección de la ERS describirá aquellos factores que, si cambian, pueden afectar a los requisitos.
2.6. Requisitos Futuros	Esta subsección esbozará futuras mejoras al sistema, que podrán analizarse e implementarse en un futuro
3. Requisitos Específicos	Esta sección contiene los requisitos a un nivel de detalle suficiente como para permitir a los diseñadores diseñar un sistema que satisfaga estos requisitos, y que permita al equipo de pruebas planificar y realizar las pruebas que demuestren si el sistema satisface, o no, los requisitos.
3.1 Requisitos comunes de las interfaces	Descripción detallada de todas las entradas y salidas del sistema de software.
3.1.1 Interfaces de usuario	Describir los requisitos del interfaz de usuario para el producto. Esto puede estar en la forma de descripciones del texto o pantallas del interfaz. Por ejemplo, posiblemente el cliente ha especificado el estilo y los colores del producto. Describa exacto cómo el producto aparecerá a su usuario previsto
3.1.2 Interfaces de hardware	Especificando las características lógicas para cada interfaz entre el producto y los componentes de hardware del sistema. Se incluirán características de configuración.
3.1.3 Interfaces de software	Indicar si hay que integrar el producto con otros productos de software.
3.1.4 Interfaces de comunicación	Describir los requisitos de interfaces de comunicación si hay comunicaciones con otros sistemas y cuáles son los protocolos de comunicación.
3.2 Requisitos funcionales	Definición de acciones fundamentales que debe realizar el software al recibir información, procesarla y producir resultados.

3.3 Requisitos no funcionales	Atributos de calidad que deben ser incorporados para proveer un sistema de buena calidad, seguro, fácil de usar y que sea valorado por el cliente respecto de su rendimiento.
3.4 Otros Requisitos	Cualquier otro requisito.

Nota: Tabla que presenta las secciones más importantes del documento E.R.S.

Conceptos de Calidad

El cliente espera que su producto tenga un justo balance entre funcionalidad y calidad, esta última indica que tanto se ajusta el producto a los requerimientos indicados y las expectativas del cliente.



Importante

Calidad (Diccionario de la Real Academia Española, s.f.). Propiedad o conjunto de propiedades inherentes a una cosa, que permiten apreciarla como igual, mejor o peor que las restantes de su especie.

IEEE (1993). Calidad de software IEEE Std. 610-1990: Grado con el que un sistema, componente o proceso cumple:

- Los requisitos especificados
- Las necesidades o expectativas del cliente o usuario.

Atributos de calidad

Los atributos de calidad son propiedades del software a los cuales se les puede asignar una métrica.

Métrica es el valor con el cual se pueden medir el atributo de calidad, pero no todas las propiedades pueden ser medidas, las métricas se dividen en:

Métricas de producto

Valores que se pueden obtener desde documentos, piezas de código, las cuales nos entregan que tan bien se ajusta el componente respecto del requerimiento no funcional o atributo de calidad.

Métricas de Proceso

Valor numérico que describe un proceso de software.

Ejemplo:

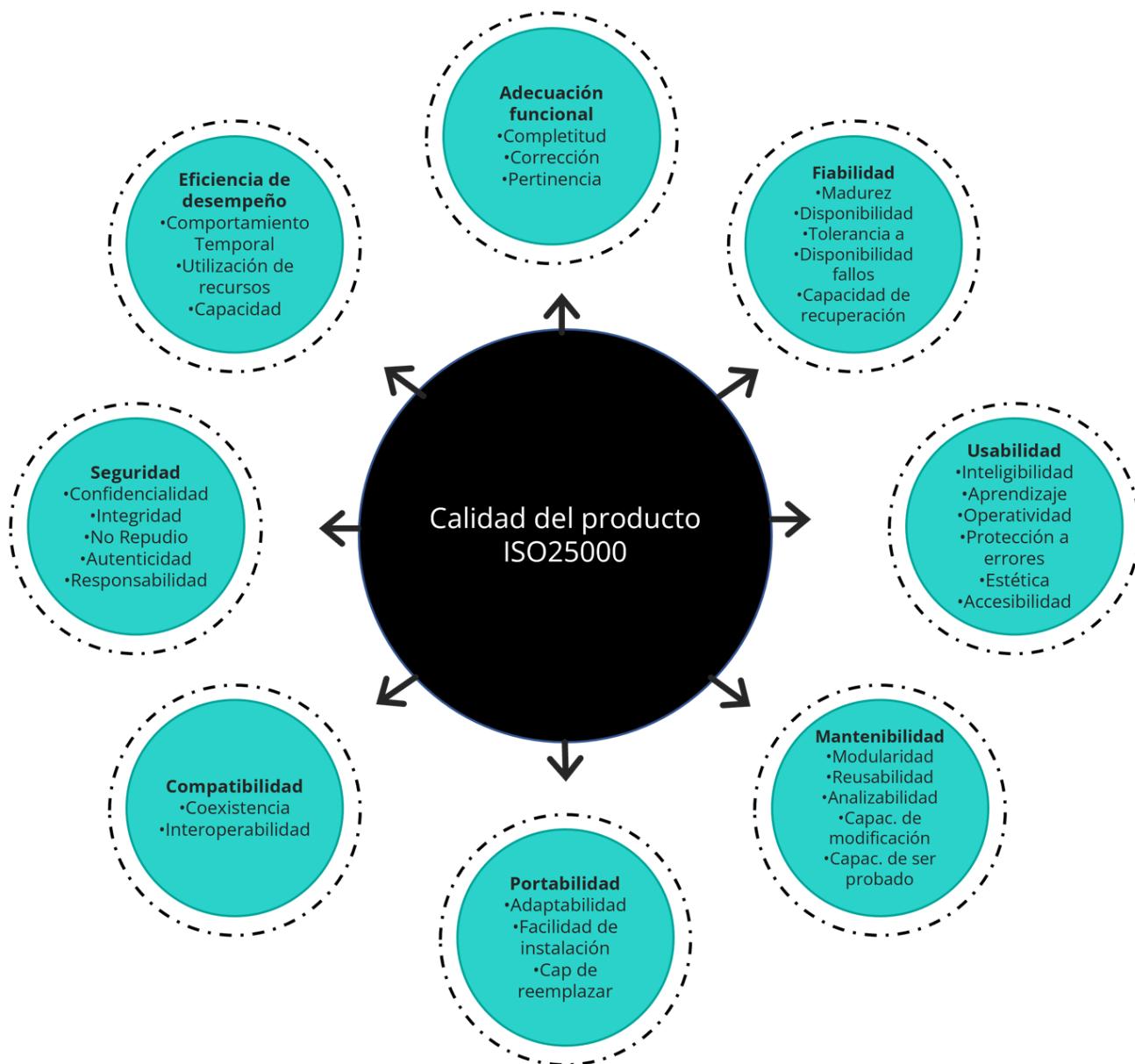
El cliente pide que el ingreso de los datos del formulario solicitado, este activo 24x7 en la Internet.

En este caso el atributo de calidad es FIABILIDAD y su métrica es DISPONIBILIDAD.

La mejor forma de comprender cuales son los atributos de calidad y sus métricas se puede visualizar en la Figura 5, la cual nos muestra cada atributo de calidad y su métrica asociada, esto según la visión de ISO/IEC 25010.

Figura 5:

Atributos de calidad y sus métricas.



Nota: Modelo de calidad del producto ISO25000 que **se define por 8 características de atributos de calidad y métricas**: adecuación funcional, fiabilidad, usabilidad, mantenibilidad, portabilidad, compatibilidad, seguridad y eficiencia de desempeño.

A continuación, se explicarán todos los atributos de calidad y sus métricas asociadas.

Adecuación funcional

Representa la capacidad del producto software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones especificadas.

Esta característica se subdivide a su vez en las siguientes sub características:

- **Compleitud funcional.** Grado en el cual el conjunto de funcionalidades cubre todas las tareas y los objetivos del usuario especificados.
- **Corrección funcional.** Capacidad del producto o sistema para proveer resultados correctos con el nivel de precisión requerido.
- **Pertinencia funcional.** Capacidad del producto software para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario especificados.

Eficiencia de desempeño

Esta característica representa el desempeño relativo a la cantidad de recursos utilizados bajo determinadas condiciones.

Esta característica se subdivide a su vez en las siguientes sub características:

- **Comportamiento temporal.** Los tiempos de respuesta y procesamiento y los ratios de *throughput* de un sistema cuando lleva a cabo sus funciones bajo condiciones determinadas en relación con un banco de pruebas (*benchmark*) establecido.
- **Utilización de recursos.** Las cantidades y tipos de recursos utilizados cuando el software lleva a cabo su función bajo condiciones determinadas.
- **Capacidad.** Grado en que los límites máximos de un parámetro de un producto o sistema software cumplen con los requisitos.

Compatibilidad

Capacidad de dos o más sistemas o componentes para intercambiar información y/o llevar a cabo sus funciones requeridas cuando comparten el mismo entorno hardware o software.

Esta característica se subdivide a su vez en las siguientes sub características:

- **Coexistencia.** Capacidad del producto para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes sin detrimento.
- **Interoperabilidad.** Capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.

Usabilidad

Capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones.

Esta característica se subdivide a su vez en las siguientes sub características:

- **Capacidad para reconocer su adecuación.** Capacidad del producto que permite al usuario entender si el software es adecuado para sus necesidades.
- **Capacidad de aprendizaje.** Capacidad del producto que permite al usuario aprender su aplicación.
- **Capacidad para ser usado.** Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad.
- **Protección contra errores de usuario.** Capacidad del sistema para proteger a los usuarios de hacer errores.
- **Estética de la interfaz de usuario.** Capacidad de la interfaz de usuario de agradar y satisfacer la interacción con el usuario.
- **Accesibilidad.** Capacidad del producto que permite que sea utilizado por usuarios con determinadas características y discapacidades para intercambiar información y utilizar la información intercambiada.

Fiabilidad

Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados.

Esta característica se subdivide a su vez en las siguientes sub características:

- **Madurez.** Capacidad del sistema para satisfacer las necesidades de fiabilidad en condiciones normales.
- **Disponibilidad.** Capacidad del sistema o componente de estar operativo y accesible para su uso cuando se requiere.
- **Tolerancia a fallos.** Capacidad del sistema o componente para operar según lo previsto en presencia de fallos hardware o software.
- **Capacidad de recuperación.** Capacidad del producto software para recuperar los datos directamente afectados y reestablecer el estado deseado del sistema en caso de interrupción o fallo.

Seguridad

Capacidad de protección de la información y los datos de manera que personas o sistemas no autorizados no puedan leerlos o modificarlos.

Esta característica se subdivide a su vez en las siguientes sub características:

- **Confidencialidad.** Capacidad de protección contra el acceso de datos e información no autorizados, ya sea accidental o deliberadamente.
- **Integridad.** Capacidad del sistema o componente para prevenir accesos o modificaciones no autorizados a datos o programas de ordenador.
- **No repudio.** Capacidad de demostrar las acciones o eventos que han tenido lugar, de manera que dichas acciones o eventos no puedan ser repudiados posteriormente.
- **Responsabilidad.** Capacidad de rastrear de forma inequívoca las acciones de una entidad.
- **Autenticidad.** Capacidad de demostrar la identidad de un sujeto o un recurso.

Mantenibilidad

Representa la capacidad del producto software para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas.

Esta característica se subdivide a su vez en las siguientes:

- **Modularidad.** Capacidad de un sistema o software que permite que un cambio en un componente tenga un impacto mínimo en los demás.
- **Reusabilidad.** Capacidad de un activo que permite que sea utilizado en más de un sistema software o en la construcción de otros activos.
- **Analizabilidad.** Facilidad con la que se puede evaluar el impacto de un determinado cambio sobre el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar las partes a modificar.
- **Capacidad para ser modificado.** Capacidad del producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño.
- **Capacidad para ser probado.** Facilidad con la que se pueden establecer criterios de prueba para un sistema o componente y con la que se pueden llevar a cabo las pruebas para determinar si se cumplen dichos criterios.

Portabilidad

Capacidad del producto o componente de ser transferido de forma efectiva y eficiente de un entorno hardware, software, operacional o de utilización a otro.

Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

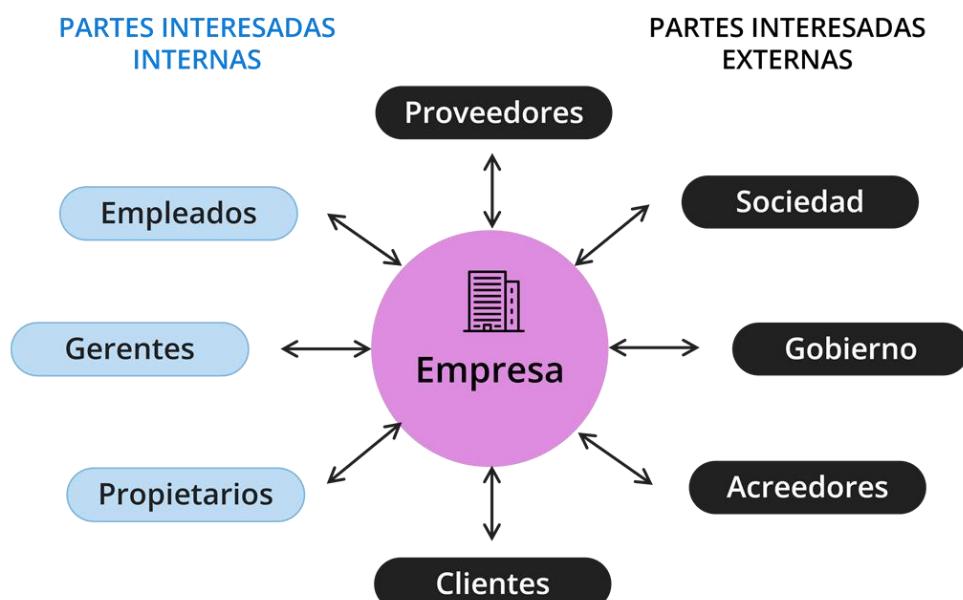
- **Adaptabilidad.** Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.
- **Capacidad para ser instalado.** Facilidad con la que el producto se puede instalar y/o desinstalar de forma exitosa en un determinado entorno.
- **Capacidad para ser reemplazado.** Capacidad del producto para ser utilizado en lugar de otro producto software determinado con el mismo propósito y en el mismo entorno.

Los Stakeholders

Los Stakeholders o “Afectados”, son todas aquellas personas que de alguna manera afectan o se ven afectados por las actividades del proceso, en este caso, su participación en la arquitectura tiene un carácter preponderante, ya que serán ellos quien en conjunto con el arquitecto realizan el proceso de validación de la arquitectura.

Figura 6:

Stakeholders. Partes Interesadas en una organización.



Nota: La figura 6 muestra a la empresa en el centro y a sus partes interesadas internas (empleados, gerentes y propietarios) junto con las partes interesadas externas (proveedores, sociedad, gobierno, acreedores y clientes) tributando hacia la empresa y la empresa hacia ellos.

Dato



El concepto de Stakeholders fue utilizado por primera vez en la obra “Strategic Management: A Stakeholder Approach” del autor Edward Freeman.

Escenarios de calidad

Un escenario mostrará el proceso de negocio, y como se ve reflejada la operación del sistema en construcción, indicará distintas situaciones en la cual el sistema de desempeñará, dejando a la vista posibles problemáticas que se deben profundizar en el desarrollo, por ejemplo si el sistema a construir es altamente transaccional, es decir recibe una gran cantidad de transacciones, deberemos preocuparnos del rendimiento del sistema, de sus canales de comunicación y de la infraestructura donde será ejecutado. Mientras que si un sistema, más bien es informativo o es un catálogo de productos, deberemos preocuparnos por los atributos de cara a la facilidad de uso, operación y estética.

Para poder entender de mejor manera un escenario de calidad, se utilizarán plantillas estándares para documentar los escenarios, a continuación, se presentan los documentos escenarios de calidad.

Los escenarios de calidad permiten detallar en qué condiciones y de qué manera debe cumplirse un atributo de calidad. Son casos precisos en los que podremos evidenciar si el sistema cumple o no con un atributo de calidad, además se centran en lo que concierne al atributo y no en los términos utilizados, en la figura 7, se puede apreciar la secuencia de un escenario de calidad.

Figura 7:

Secuencia de un escenario de calidad.



Nota: Secuencia de un escenario de calidad que va desde la fuente del estímulo (o quién provocará el artefacto), pasando por el estímulo hacia el artefacto (que parte del sistema se

ejecutará o activará) hacia el estímulo que lleva a la medida de la respuesta Criterio para probar el requerimiento.

Para documentar un escenario de calidad se deben llenar la siguiente información

Tabla 3:

Información para documentar un escenario de calidad.

Ítem	Descripción
Fuente del estímulo	Quién o que genera el estímulo para el escenario, pueden ser agentes externos (usuarios, sistema) o internos (procesos).
Estímulo	Qué estímulo es el que va a ocurrir para evidenciar el atributo (eventos de intercambio/request de datos entre las fuentes)
Entorno	Condiciones dentro de las cuales se presenta el estímulo.
Artefacto	Qué parte del sistema va a recibir el estímulo (servicios del sistema).
Respuesta	Actividad que ocurre luego de la llegada del estímulo.
Medida de la Respuesta	Criterio para testear el requerimiento. Atributo de calidad afectado: Atributo de calidad relacionado con el escenario.

Nota: Tabla que representa la información que sirva para documentar un escenario de calidad y su descripción

Documentando los escenarios de calidad

En el siguiente recuadro, se aprecia que el escenario de calidad Número 1, está orientado a comprobar que la aplicación siendo construida se podrá ejecutar sin importar el sistema operativo del dispositivo a utilizar, lo cual requiere de atributos de Portabilidad.

Escenario N°1

Descripción: Se requiere que la aplicación pueda ser utilizada en un dispositivo basado en Android y en otro basado en IOS.

- Afecta: A la aplicación
- Atributo: Portabilidad
- Métrica: Adaptabilidad

Validación del Escenario

- Origen del Estímulo: Usuario
- Estímulo: Usa la aplicación en un dispositivo Android y en un dispositivo IOS.
- Entorno: dispositivos con la última versión de Android e IOS
- Artefacto: Capa de presentación
- Respuesta: La aplicación se despliega normalmente en ambos dispositivos.
- Medida de la Respuesta: Inmediata

En el siguiente recuadro, se aprecia que el escenario de calidad Número 2, está orientado a comprobar que la aplicación siendo construida se podrá utilizar a la hora que el cliente la necesite, lo cual requiere de atributos de Fiabilidad.

Escenario N°2

Descripción: Se requiere que la aplicación pueda ser utilizada 24x7.

- Afecta: A la aplicación
- Atributo: Fiabilidad
- Métrica: Disponibilidad

Validación del Escenario

- Origen del Estímulo: Usuario
- Estímulo: Usa la aplicación a las 08:30, 17:00, 23:00 y 04:00 am.
- Entorno: Ambiente productivo con alta disponibilidad.
- Artefacto: Sistema/Aplicación
- Respuesta: La aplicación se despliega normalmente en todos los horarios probados.
- Medida de la Respuesta: Inmediata

O bien la documentación puede ser usada a través de la plantilla de escenarios de calidad, adjunta como anexo a este documento.

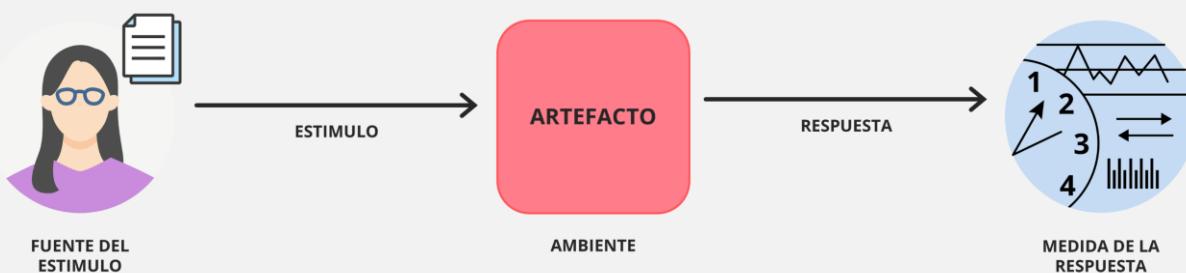
Los siguientes 2 recuadros corresponden a los escenarios anteriores, pero documentados con el formato de plantilla de escenarios de calidad.

Escenario de Calidad N° 1

Atributo de Calidad Asociado (Portabilidad):

Sub característica: Adaptabilidad

Descripción: Se requiere que la aplicación pueda ser utilizada en un dispositivo basado en Android y en otro basado en IOS.



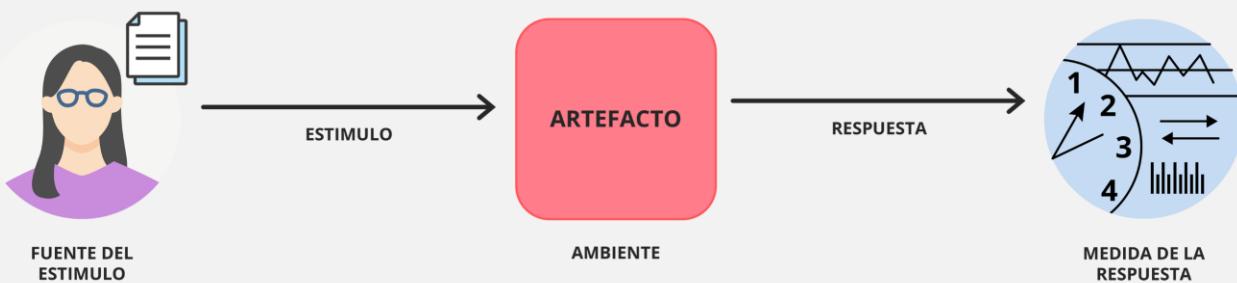
Fuente del Estímulo:	Usuario
Estímulo:	Usa la aplicación en un dispositivo Android y en un dispositivo IOS
Artefacto:	Capa de presentación
Ambiente:	Dispositivos con la última versión de Android e IOS
Respuesta:	La aplicación se despliega normalmente en ambos dispositivos.
Medida de Respuesta:	Inmediata

Escenario de Calidad N° 2

Atributo de Calidad Asociado (Fiabilidad):

Sub característica: Disponibilidad

Descripción: Se requiere que la aplicación pueda ser utilizada 24x7.



Fuente del Estímulo:	Usuario
Estímulo:	Usa la aplicación a las 08:30, 17:00, 23:00 y 04:00 am.
Artefacto:	A la aplicación
Ambiente:	Ambiente productivo con alta disponibilidad.
Respuesta:	La aplicación se despliega normalmente en todos los horarios probados.
Medida de Respuesta:	Inmediata

Tips y recomendaciones

Tips para obtener requisitos funcionales

- **Revisión documental:** Revisar con detalle toda la información que entrega el cliente, esto puede ocurrir a través del product owner, en caso de que la empresa considere agilidad como metodología de gestión de proyectos. Estos documentos deben ser leídos con cuidado ya que entre líneas puede haber algún requisito funcional o un atributo de calidad, documéntalo y busca a los responsables y actores.
- **Realizar cuestionarios:** Prepara cuestionarios simples, a los cuales los usuarios entrevistados puedan responder en forma clara.
- **Realizar visitas pasivas:** Visita las instalaciones del cliente, observa como los usuarios trabajan, anota y realiza esquemas del trabajo, busca posibles retardos, reprocesos y cualquier otra actividad que pueda ser mejorada, muy importante no realices preguntas a los usuarios, en esta oportunidad solo debes observar.
- **Realizar visitas activas:** Una vez que tomaste la información de las visitas pasivas, preparaste los cuestionarios, ve con los usuarios (previo acuerdo con el cliente) y realiza preguntas sobre el funcionamiento, tiempos involucrados, y documenta esto, es importante no hacer juicios de ningún tipo, ni cuestionar la forma en que trabajan.
- **Realizar preguntas abiertas y cerradas:** esta técnica puede ser útil para recolectar información, solo debes ser cuidadoso con las preguntas abiertas, ya que los usuarios, pueden explayarse demasiado y entregar una gran cantidad de información de la cual poco podrá ser usado.
- **Prefiere preguntas cerradas**, en la cual el usuario responda “Si”, “No”, “Correcto”, “es así”, etc.
- De existir, **revisa sitios web**, intranets, y aplicaciones, las cuales pueden entregarte puntos de vista como velocidad, seguridad, la estética, rendimiento, entre otros.
- **Estudia la industria** donde el cliente está inserto, esto te dará oportunidad de revisar si hay regulaciones ambientales, impositivas y/o legales.
- Es muy importante **trabajar sobre los procesos de negocio**, identifica y documenta los procesos de negocio, esto te dará una visión global de cómo se realizan las

actividades en la empresa. El levantamiento de procesos, si bien agregará una carga adicional de trabajo para ti, será una gran oportunidad de promover mejoras a los procesos, recuerda que un proyecto tecnológico debe ser implementado sobre procesos que estén bien estructurados (TO-BE)

Otros tips

- Detectar los **atributos de calidad adecuados** para los requerimientos funcionales y no funcionales implica comprender las necesidades y expectativas de los usuarios y otras partes interesadas, así como considerar las características del sistema y su entorno. Aquí hay algunas recomendaciones para ayudarte en este proceso:
- **Realizar entrevistas y consultas:** Habla con los usuarios finales, administradores, expertos en dominio y otras partes interesadas relevantes para comprender sus necesidades y expectativas. Pregunta sobre los aspectos importantes del sistema, como la seguridad, el rendimiento, la escalabilidad, la usabilidad, la disponibilidad, etc.
- **Realizar análisis de casos de uso:** Examina los casos de uso y los escenarios de uso del sistema para identificar los requisitos funcionales y los atributos de calidad asociados. Por ejemplo, si el sistema debe procesar pagos en línea, la seguridad y la confiabilidad son atributos de calidad relevantes.
- **Consultar estándares y regulaciones:** Considera los estándares de la industria y las regulaciones aplicables al dominio del sistema. Estos pueden proporcionar orientación sobre los requisitos de calidad necesarios. Por ejemplo, si el sistema maneja datos de salud, es posible que deba cumplir con la normativa de privacidad de la salud.
- **Utilizar listas de verificación de atributos de calidad:** Existen listas de verificación y catálogos de atributos de calidad disponibles, como el modelo de calidad ISO/IEC 25010 (anteriormente conocido como ISO/IEC 9126). Estas listas pueden ayudarte a identificar atributos de calidad comunes y relevantes para tu sistema.
- **Realizar análisis de riesgos:** Identifica los posibles riesgos asociados con el sistema y considera los atributos de calidad necesarios para mitigar esos riesgos. Por ejemplo,

si hay un alto riesgo de ataques ciberneticos, la seguridad y la resistencia pueden ser atributos de calidad críticos.

- Considerar restricciones técnicas y presupuestarias:** Evalúa las limitaciones técnicas y presupuestarias del proyecto. Algunos atributos de calidad pueden requerir inversiones significativas en recursos y tecnologías específicas. Asegúrate de considerar estas restricciones al definir los atributos de calidad.
- Obtener retroalimentación y validación:** Una vez que hayas identificado los atributos de calidad propuestos, revisa y valida esta lista con las partes interesadas relevantes. Obtén su retroalimentación y asegúrate de que los atributos de calidad sean consistentes con sus necesidades y expectativas.
- Recuerda que los atributos de calidad pueden evolucionar a lo largo del tiempo y durante el desarrollo del proyecto.** Mantén una comunicación abierta con las partes interesadas y asegúrate de adaptar y ajustar los atributos de calidad según sea necesario.

¿Cómo hacer un modelo arquitectónico?

Para poder armar tu modelo arquitectónico de la manera óptima te dejamos las siguientes recomendaciones:

1. **Identifica los componentes principales:** Supongamos que estás diseñando un software de gestión de proyectos. Los componentes principales podrían incluir una interfaz de usuario, una capa de lógica de negocio, una base de datos y servicios de integración externos.
2. **Define las responsabilidades de cada componente:** La interfaz de usuario se encargaría de mostrar la información y permitir la interacción del usuario. La capa de lógica de negocio se encargaría de procesar la lógica del sistema y realizar las operaciones necesarias. La base de datos almacena y recupera los datos del sistema. Los servicios de integración externos se utilizarían para comunicarse con otros sistemas o servicios externos.
3. **Considera los patrones de diseño:** Para optimizar la arquitectura del software, podrías utilizar el patrón de diseño Modelo-Vista-Controlador (MVC) para separar claramente la lógica de presentación (Vista), la lógica de negocio (Modelo) y la lógica de control (Controlador). Esto permitiría una mayor modularidad y mantenibilidad del sistema.
4. **Diseña la comunicación entre componentes:** Define cómo se comunicarán los diferentes componentes entre sí. Por ejemplo, la interfaz de usuario podría enviar solicitudes al controlador, que a su vez interactuaría con el modelo para realizar las operaciones necesarias. También podrías utilizar eventos o mensajes para comunicar cambios entre componentes.
5. **Considera los atributos de calidad:** Piensa en los atributos de calidad que deseas que tenga tu software, como rendimiento, seguridad, escalabilidad y usabilidad. Asegúrate de tener en cuenta estos atributos al diseñar la arquitectura. Por ejemplo, podrías implementar una capa de seguridad para proteger los datos sensibles del sistema o considerar la escalabilidad al diseñar la estructura de la base de datos.

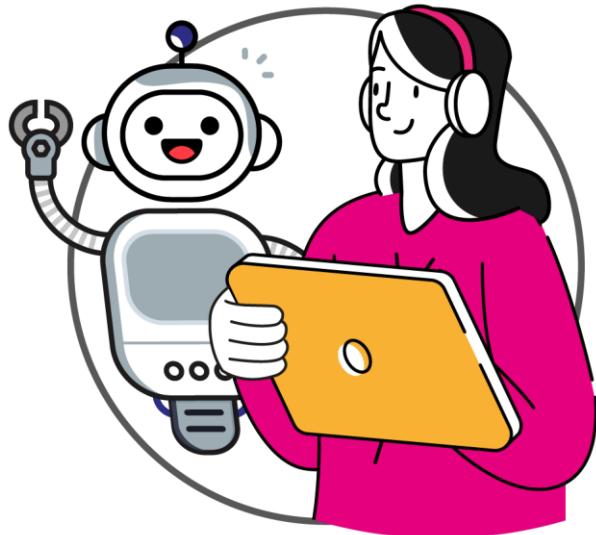
¿Cómo identificar los atributos de calidad?

1. **Comprende los requisitos del sistema:** Analiza las necesidades y expectativas de los usuarios, así como los objetivos del software. Esto te ayudará a identificar los atributos de calidad más importantes.
2. **Realiza un análisis de stakeholders:** Identifica a las partes interesadas en el software, como los usuarios finales, los propietarios del negocio y los desarrolladores. Escucha sus perspectivas y expectativas sobre los atributos de calidad que consideran importantes.
3. **Utiliza listas de atributos de calidad estándar:** Existen listas de atributos de calidad reconocidas, como los atributos de calidad ISO/IEC 25010, que pueden servir como punto de partida para identificar los atributos relevantes para tu software. Estas listas pueden ayudarte a no pasar por alto aspectos importantes.
4. **Prioriza los atributos de calidad:** No todos los atributos de calidad tendrán la misma importancia. Evalúa y prioriza los atributos en función de su relevancia para tu software y las necesidades de los usuarios.
5. **Realiza pruebas y evaluaciones en el sistema desarrollado:** A medida que avanzas en el desarrollo del software, realiza pruebas y evaluaciones para verificar si se cumplen los atributos de calidad identificados. Esto te ayudará a identificar posibles áreas de mejora y ajustar el sistema en consecuencia.
6. **Organiza sesiones de brainstorming con tu equipo:** Reúnete con tu equipo de desarrollo para generar ideas sobre posibles atributos de calidad. El intercambio de ideas puede ayudar a descubrir aspectos importantes que podrían haber sido pasados por alto inicialmente.

Cierre de la semana

Durante esta primera semana, has aprendido acerca de la arquitectura de software, reforzando conceptos como roles del “Arquitecto” y “Stakeholders”, del mismo modo has aprendido sobre requerimientos funcionales, que es lo que el sistema hace, requerimientos no funcionales o atributos de calidad, los que dan las características de cómo se ejecutará el software y bajo qué condiciones lo hará, lo que se representa a través de un escenario de calidad, finalizando la semana, tuviste la oportunidad de aprender sobre riesgos del proyecto y como poder evaluar el impacto que tendríamos si alguno de estos riesgos se materializara.

Esperamos haya sido una gran semana de aprendizaje, los espero en la siguiente semana, donde seguiremos profundizando en los diferentes temas referidos a la arquitectura de software, tengan una buena semana.



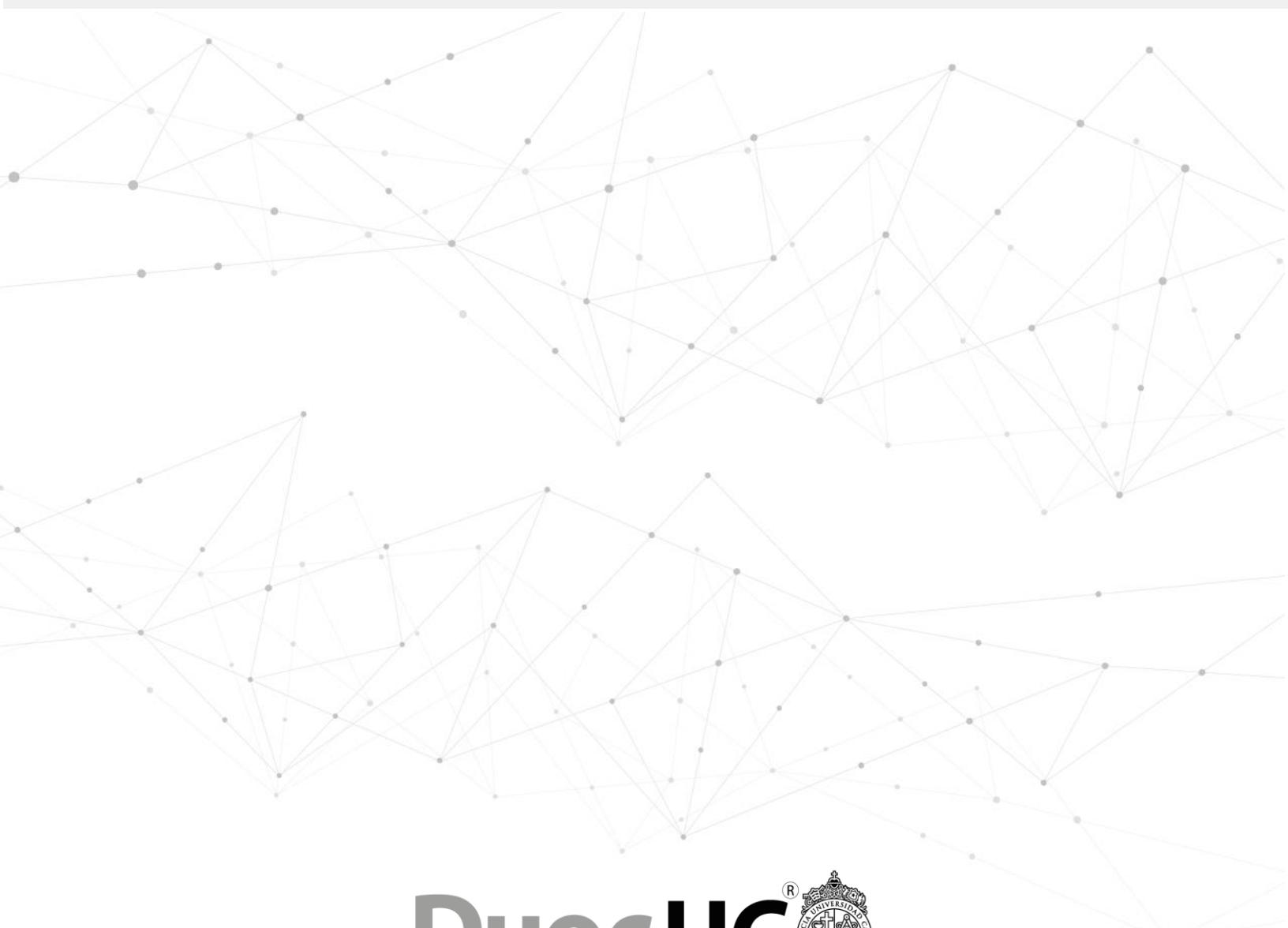
Referencias

- ✓ ISO (2010). ISO 25010: *Systems and software engineering - Systems and software Quality Requirements and Evaluation* (SQuaRE) - System and software quality models.
- ✓ Cervantes, H., Velasco, P., & Castro, L. (2016). *Arquitectura de software: Conceptos y desarrollo* (1a ed.). Cengage Learning Editores, S.A.
<https://www.iso.org/standard/43808.html>.
- ✓ Pressman, R. S. (2006). *Ingeniería del Software*: Un enfoque práctico (7th ed.).
- ✓ Bass, L., Clements, P., & Kazman, R. (2006). *Software Architecture in Practice* (2nd ed.). Addison-Wesley.
- ✓ Rozanski, N., & Woods, E. (2005). *Software Systems Architecture*. Addison-Wesley.

Lecturas de la semana

- ✓ Capítulo 7: Comprensión de los requerimientos
Fuente: Pressman, R. S. (2021). Ingeniería del Software: Un enfoque práctico (9th ed.). McGraw Hill. Páginas: 102 a la 125.
- ✓ Calidad del Software
Fuente: Sommerville, I. (2011). Ingeniería de software. D. F: Pearson. Páginas: 655-657.

Apunte



DuocUC®
ONLINE

Duoc UC

Facilitador disciplinar: Juan Alberto Gana.

Asesor par: Domingo Sanz.

Reservados todos los derechos Fundación Instituto Profesional Duoc UC. No se permite copiar, reproducir, reeditar, descargar, publicar, emitir, difundir, de forma total o parcial la presente obra, ni su incorporación a un sistema informático, ni su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia, grabación u otros) sin autorización previa y por escrito de Fundación Instituto Profesional Duoc UC La infracción de dichos derechos puede constituir un delito contra la propiedad intelectual.



Carrera **Analista Programador Computacional**

Exp 1 – Semana 2

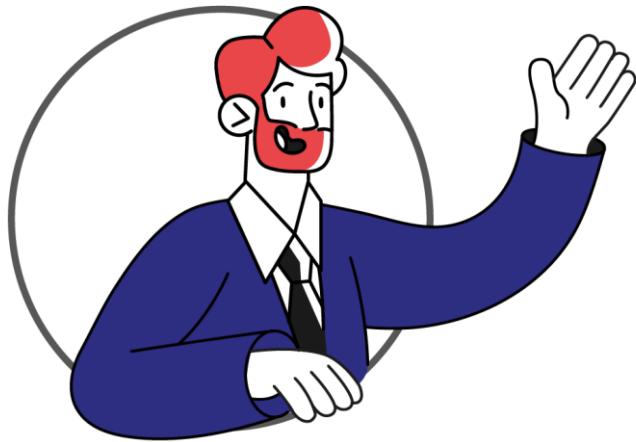
Arquitectura ASY4231

Guía de Aprendizaje

Índice

Introducción a la semana.....	.3
Resultado de aprendizaje	4
El estudiante será capaz de:	4
Indicador de logro:.....	4
Conceptos relevantes	4
Preguntas activadoras.....	4
Actividad	5
Conociendo el modelo arquitectónico.....	6
Modelo de arquitectónico	6
Modelo de vistas 4+1	7
Gestión de riesgo	17
Cierre de la semana	22
Referencias	23
Lecturas de la semana	24
Apunte	25

Introducción a la semana



En esta segunda semana diagramarás la arquitectura del software, para ello deberás conocer los conceptos de modelos arquitectónicos, vistas 4+1 y riesgos.

De esta manera darás el soporte requerido como solución a la construcción de tu arquitectura.

Tendrás que diferenciar cada uno de estos diagramas y cuáles son sus funciones, uso y transversalidades.

Tendrás que usar herramientas de base de datos, como conocimiento previo a la diagramación, igualmente darle continuidad con los componentes de persistencia.

Para esto deberás entregar un formato de respuesta y una plantilla de registro de los riesgos. Donde podrás identificar y documentar los distintos riesgos del proyecto, indicando su plan de mitigación para aplicar en caso de que estos se materialicen.

Así que te invitamos a conocer sobre los diagramas y a trabajar con los riesgos de tu proyecto.

Resultado de aprendizaje

El estudiante será capaz de:

RA1. Diseña un modelo arquitectónico para soportar la solución de acuerdo a su factibilidad, a los estándares de la industria y considera los riesgos de la solución propuesta.

Indicador de logro:

IL3. Implementa el modelo arquitectónico seleccionado a través de la diagramación del modelo de vistas 4+1, para soportar los atributos de calidad requeridos de la organización.

IL4. Evalúa los impactos de los riesgos identificados del modelo diseñado, de acuerdo a la arquitectura propuesta.

Conceptos relevantes

	Modelo	Riesgo	Punto de vista
	Diagrama	Impacto	Magnitud
	Caso de Uso	Escenarios	Modelo arquitectónico

Preguntas activadoras

- ¿Qué es un modelo?
- ¿Qué es un punto de vista?
- ¿Qué es un riesgo en un proyecto?

Actividad

Descripción de la actividad

En esta segunda semana realizarán una actividad formativa de manera grupal, llamada "Implementando los primeros pasos en el diseño del modelo arquitectónico" en el cual implementaremos el modelo de vistas 4+1, que da soporte a la construcción de la solución, para ello utilizarán el lenguaje orientado a objetos, herramientas de base de datos y componentes de persistencia. Igualmente tendrán que identificar y documentar los riesgos del proyecto.

Conociendo el modelo arquitectónico

Modelo de arquitectónico

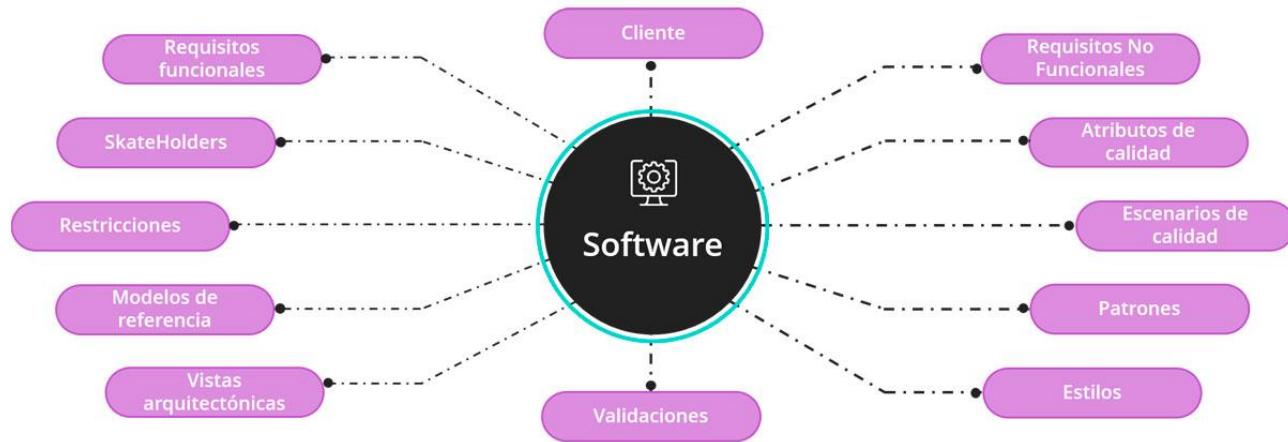
¿Qué es un modelo?

Un modelo según la RAE es (entre otras definiciones) “Arquetipo o punto de referencia para imitarlo o reproducirlo.”, por lo tanto, un modelo arquitectónico, es el conjunto de características, técnicas, componentes, restricciones y decisiones que dictarán la base de la construcción del software proceso que repetiremos en cada proyecto de construcción de software, dado que contiene las mejores prácticas de la industria, normas y recomendaciones, para complementar, utilizaremos modelos de referencia como aporte, pudiendo generar nuestros propios modelos ad-hoc a las soluciones que implementaremos.

En la figura 1 podemos apreciar un modelo arquitectónico que tiene como componentes, cliente, restricciones, requerimientos funcionales, atributos de calidad, restricciones, entre otros componentes.

Figura 1:

Modelo.



Nota: En el centro de la imagen se presenta la palabra Software, como un modelo arquitectónico que tiene como componentes, cliente, restricciones, requerimientos funcionales, atributos de calidad, restricciones, entre otros componentes.

Modelo de vistas 4+1

Una vez iniciado el proceso de construcción del modelo arquitectónico, y habiendo identificado y documentado los requisitos funcionales y los atributos de calidad, debemos diseñar la arquitectura del software, para ello debemos preguntarnos quienes serán los principales consumidores de nuestra arquitectura, por lo tanto debeos preocuparnos de los clientes y usuarios, desarrolladores, ingenieros de sistemas, integradores, especialistas en procesos, administradores de base de datos, entre otros roles de un proyecto de desarrollo de software.

Philippe Kruchten, en 1995, creó un modelo para describir la arquitectura del software, desde el punto de vista de los roles que trabajarán con cada parte de la arquitectura, es así como el diagrama siguiente nos muestra el modelo completo de las vistas según Kruchten, ver Figura

Figura 2:

Modelo de vistas 4 + 1.



Nota. Se presenta un esquema, en el centro +1 vista de escenarios y alrededor tributando la vista lógica, vista de despliegue, vista física y vista de procesos.

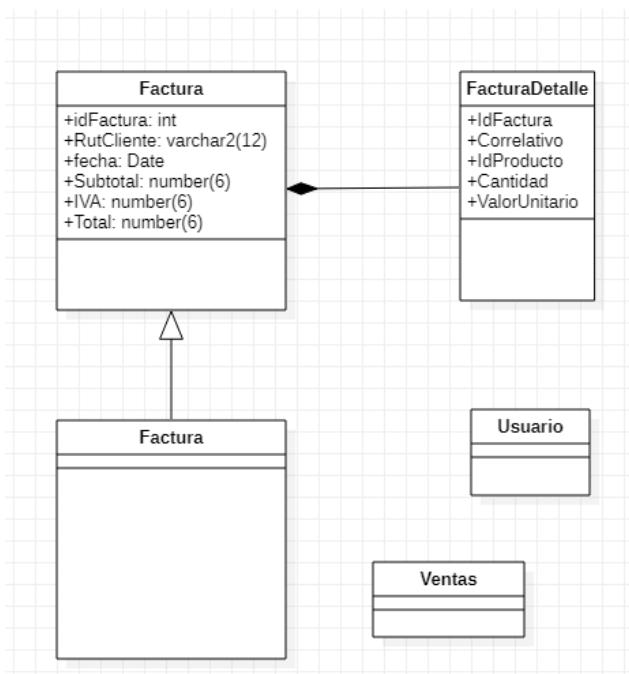
Vista Lógica

La vista lógica representa los requisitos funcionales que el sistema considera y que entregará a los usuarios finales cada acción solicitada por el cliente.

La vista lógica considera los siguientes diagramas:

Figura 3:

Diagrama de clases.



Nota. en el diagrama se identifican las pestañas usuario y ventas en la esquina inferior derecha de manera aislada y las pestañas “Factura” (la cual no tiene información) y “Factura detalle” (la cual tiene información +Id Factura +Correlativo +IdProducto +Cantidad +ValorUnitario) indicando con una flecha hacia “Factura”, el cual contiene los mismos elementos que “Factura detalle”.

Video

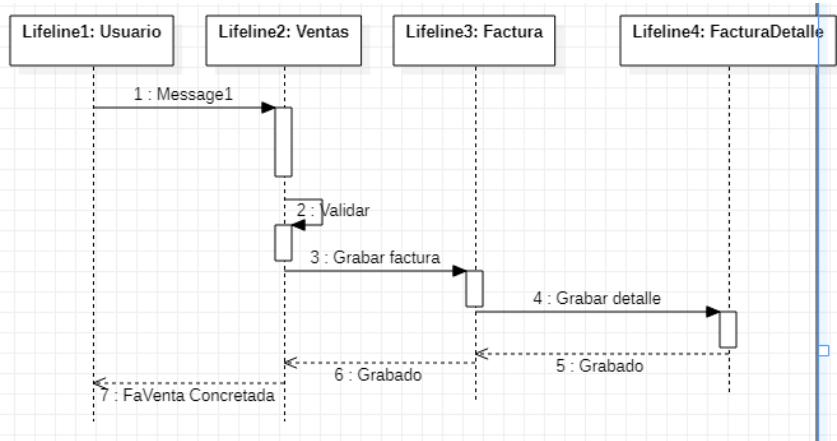


En el siguiente video podrás observar este tipo de Vista Lógica en el Diagrama de clases:

https://videosduoc.duoc.cl/media/t1_fimqq6hl

Figura 4:

Diagrama de secuencia.



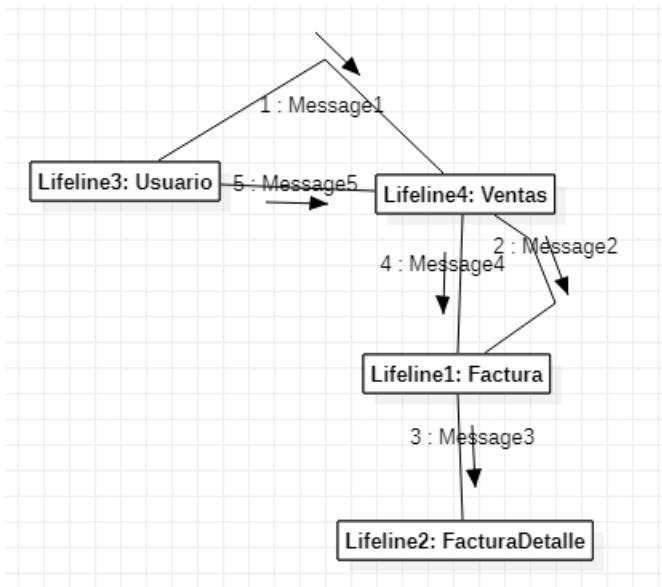
Nota. El diagrama de secuencia indica en distintos lifelines, las relaciones entre los usuarios, ventas, factura y factura detalle.

Video



En el siguiente video podrás observar este tipo de Vista Lógica en el Diagrama de secuencia:

https://videosduoc.duoc.cl/media/t1_u11wmj0i

Figura 5:*Diagrama de comunicación.*

Nota. El diagrama de comunicación indica al Usuario como mensaje a Ventas y Ventas como mensaje a Factura. Luego factura como mensaje a FacturaDetalle.

Video

En el siguiente video podrás observar este tipo de Vista Lógica en el Diagrama de comunicación: https://videosduoc.duoc.cl/media/t/1_shmfjfyc

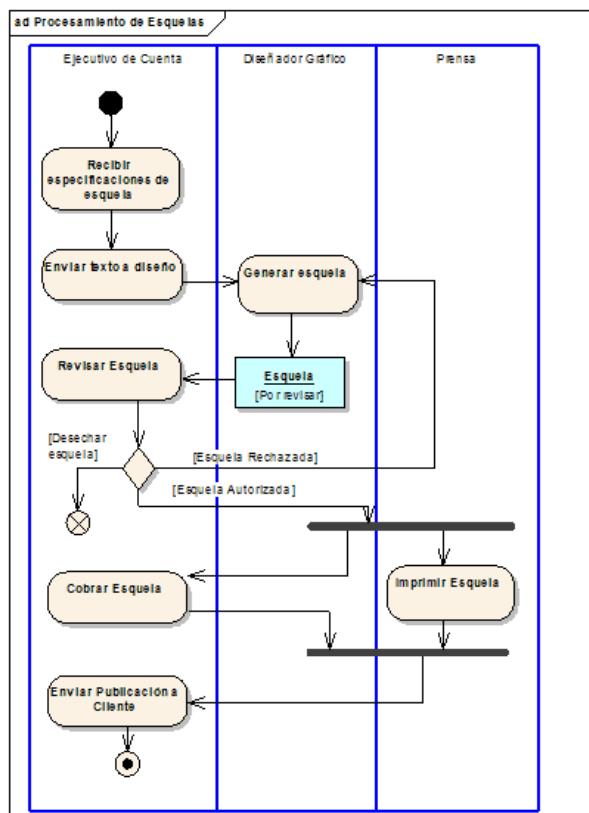
Vista de Procesos

Esta vista muestra los procesos de negocios de la empresa, y como se representarán en el sistema, indicando sus comunicaciones e integraciones. La vista de procesos esta orientada a los Integradores de sistemas, quienes tienen la misión de diseñar como el sistema se comunicará con el resto de sistema u otros sistemas a través de APIs, webservices y buses de integración.

La vista de Proceso considera los siguientes diagramas:

Figura 6:

Ejemplo de un Diagrama de actividades.



Nota. El diagrama de actividades presenta en tres áreas el procesamiento de esquemas: En este ejemplo: El ejecutivo de cuenta, el diseñador gráfico y la prensa los cuales se relacionarán en diferentes etapas del proceso.

Video



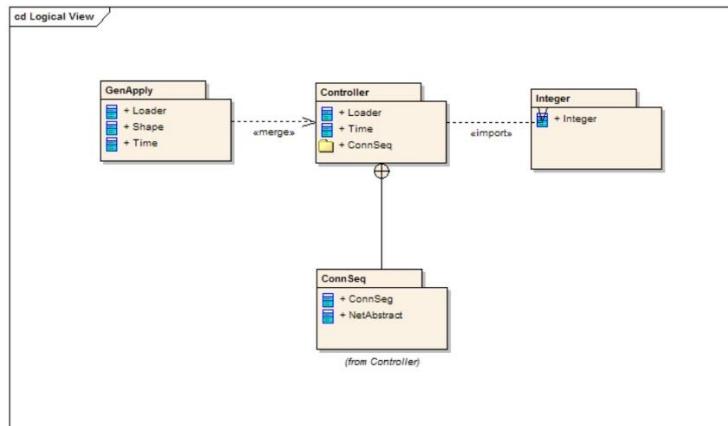
En el siguiente video podrás observar este tipo de Vista de Procesos en el Diagrama de actividades https://videosduoc.duoc.cl/media/t/1_v979wnn1

Vista de Desarrollo

La vista de desarrollo visualiza el sistema desde el punto de vista de los desarrolladores, por lo tanto, se preocupa de la gestión del desarrollo, indicando los componentes del sistema, como se comunican, sus dependencias. La vista de desarrollo considera los siguientes diagramas:

Figura 7:

Diagrama de componentes.



Nota. Diagrama de componentes en donde se aprecia una carpeta con GenApply que indica un paso a Controller y desde esta carpeta a Integer. Desde la carpeta Controller se despliega la opción ConnSeq.

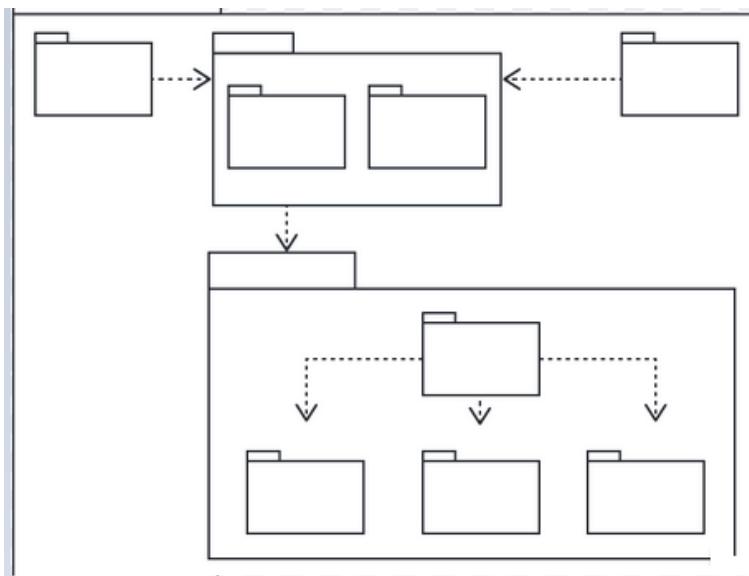
Video



En el siguiente video podrás observar este tipo de Vista de Despliegue en el Diagrama de componentes: https://videosduoc.duoc.cl/media/t1_n3dfotzl

Figura 8:

Diagrama de paquetes.



Nota. El diagrama muestra carpetas que llevan a una carpeta principal, de la cual se despliega otro grupo de carpetas.

Video



En el siguiente video podrás observar este tipo de Vista de despliegue en el Diagrama de paquetes: https://videosduoc.duoc.cl/media/t1_d8u2sfjt

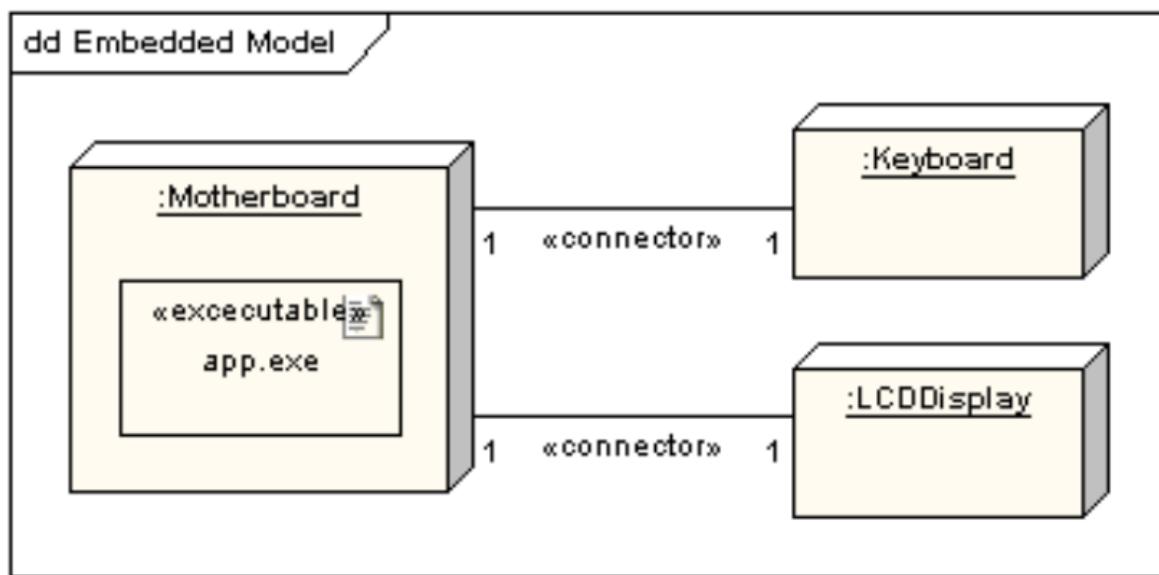
Vista Física

La vista física, muestra la perspectiva de los ingenieros de sistemas, quienes administran y diseñan la infraestructura donde el sistema será ejecutado, sus conexiones, redes, protocolos, servicios, topología en general.

La vista física considera los siguientes diagramas:

Figura 9:

Diagrama de despliegue.



Nota. El diagrama de despliegue tiene como centro :Motherboard que usa como connector :keyboard y :LCDDisplay

Video



En el siguiente video podrás observar este tipo de Vista Física en el Diagrama de despliegue:

https://videosduoc.duoc.cl/media/t/1_i1vffmb8

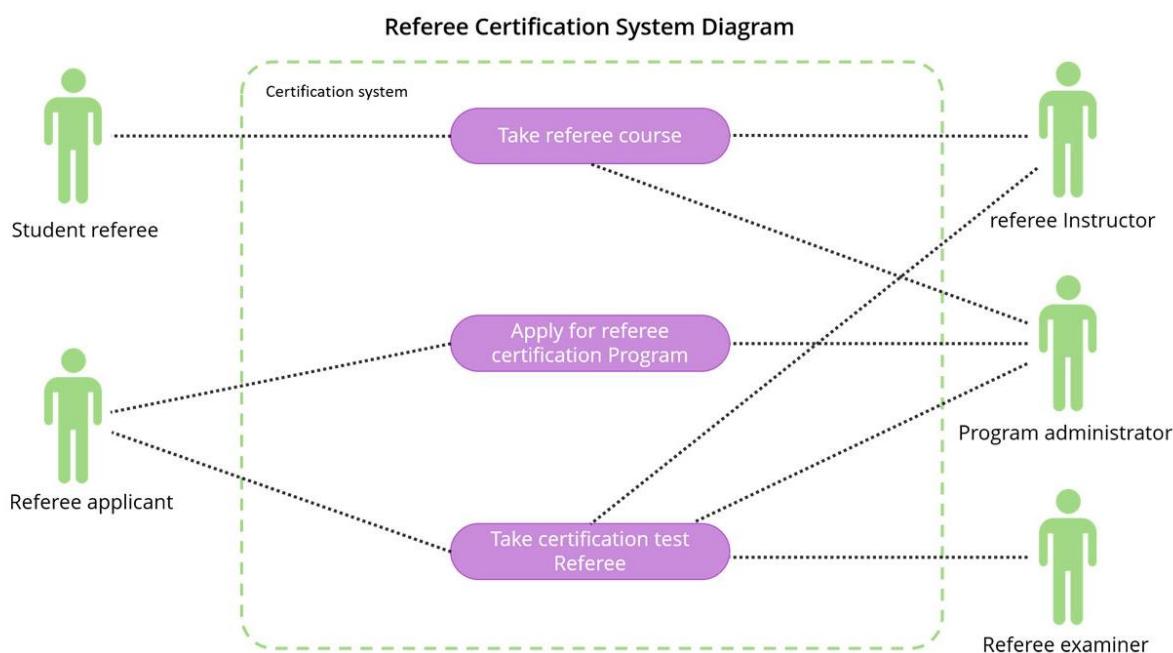
Vista de escenarios

La vista de escenarios llamada “+1”, es la encargada de coordinar a las 4 vistas anteriores, ya que, a través de expresar la funcionalidad con los casos de uso, triangula el resto de las vistas y diagramas. Siendo los casos de uso los encargados de explicar de forma sencilla los requerimientos funcionales del sistema.

La vista de escenarios considera el siguiente diagrama:

Figura 10:

Diagrama de clases.



Nota. Los diagramas de casos de uso muestran el comportamiento esperado del sistema. No muestran el orden en que se realizan los pasos.

Video



En el siguiente video podrás observar este tipo de Vista de Escenarios en el Diagrama de clases: https://videosduoc.duoc.cl/media/t/1_h1ritdxu

Importante

Para realizar los diagramas, deberás usar los softwares de StarUML o Visio, a continuación, observa los siguientes links que te darán información sobre su uso para que puedas hacer la actividad.

Links de Interés

Descargar StarUML, revisa el siguiente tutorial: https://videosduoc.duoc.cl/media/t/1_zi0hcjss

Para la creación de clases con StarUML haz click:

https://videosduoc.duoc.cl/media/t/1_jvshiokf

Aquí un ejemplo de cómo crear un modelo entidad-relación

https://videosduoc.duoc.cl/media/t/1_5iew4pna

Links de Interés

Para acceder a Visio 365, lo puedes hacer desde tu cuenta DuocUC de Office 365, es muy fácil acceder ya que es una aplicación más de la familia Office 365.

- Visio 365: Sesión 4.3: Crear un Diagrama de Bloques
https://videosduoc.duoc.cl/media/t/1_e71vfdkj
-

Gestión de riesgo

“El mayor riesgo de todos sería no aceptar ninguno”

Mellody Hobson

Un riesgo es un evento posible en el futuro, que en caso de ocurrir tendrá un efecto sobre las metas de proyecto.

- Se caracteriza por la probabilidad de que ocurra y por el impacto que tendrá en las metas del proyecto
- Una buena administración de proyectos contempla el seguimiento de los riesgos muy probables con potenciales impactos significativos.
- Por su naturaleza, cualquier riesgo asociado a la arquitectura tiene casi siempre una consecuencia muy alta y negativa en las metas del proyecto.
- Según sea el proyecto se pueden clasificar en:
 - Requerimientos funcionales
 - Requerimientos no funcionales
 - Desconocimiento técnico
 - Decisiones arquitectónicas erradas
 - Recursos humanos y monetarios

Los riesgos son eventos que pueden ser negativos (amenazas) o positivos (oportunidades), en el caso de los riesgos negativos que pueden traer consecuencias no deseadas para el proyecto, muy probablemente provocarán desviaciones en este, en tiempo, presupuesto y forma.

Las evaluaciones permiten tener visibilidad sobre los riesgos más probables y costosos a los que pueda enfrentarse el proyecto.

Esto facilita la toma de decisiones para evitar o disminuir el impacto de los riesgos, con lo cual se establecen estrategias proactivas de la administración del proyecto para que este pueda alcanzar sus objetivos.

En la tabla 1 se muestran algunas categorías de posibles riesgos asociados a proyectos.

Tabla 1:

Categorías de posibles riesgos asociados a proyectos.

Categorías	Descripción
Requerimientos funcionales	Las funciones construidas en el producto no satisfacen lo que los usuarios necesitan
Requerimientos no funcionales	El sistema es incapaz de satisfacer los atributos de calidad que espera el usuario
Desconocimiento técnico	El personal no cuenta con el conocimiento necesario para plantear una solución técnica que satisfaga los requerimientos de los usuarios.

Nota. La Tabla muestra las categorías de posibles riesgos asociados a proyectos, como lo son los requerimientos funcionales, no funcionales y desconocimiento técnico.

Tabla 2:
Planilla de riesgos

Ítem	Descripción
Fecha Identificación	Fecha en la que se identifica el riesgo.
Nro.	Una identificación, índice, etc.
Riesgo/Evento de Riesgo	Descripción del riesgo
Categoría	Categoría, proyecto, recursos humanos, presupuesto, etc.
Fuente/Causa/Condición	Quien, como, donde se produce el riesgo
Impacta a/ Consecuencias /Objetivos Proyecto	Como afecta, al proyecto, al cronograma, al presupuesto, etc
Descripción y Comentarios	Descripción de tallada del riesgo
Probabilidad	Que tan probable es que ocurra, ALTA, MEDIA, BAJA
Valor de Probabilidad	Valor de la probabilidad, ALTA=3, MEDIA = 2 o BAJA=1
Impacto	Que tanto nos impacta, ALTA, MEDIA, BAJA
Valor Impacto	Valor del impacto, ALTA=3, MEDIA = 2 o BAJA=1
Magnitud	Es la multiplicación del valor del impacto y la probabilidad
Asignado a (responsable)	Quien se hará responsable del monitoreo del riesgo.
Cuando / Fecha estimada de ocurrencia	Si es posible, indicar en qué fecha es probable que ocurra
Mitigación o Plan de Contingencia	Qué acciones, se planificarán para tratar de minimizar la magnitud del riesgo.
Estado	En qué estado se encuentra el riesgo.
Fecha de Compromiso	Fecha en la cual se revisará el riesgo.

Nota. Se muestra en la tabla la planilla de riesgos con sus elementos y descripciones de cada uno.

Para documentar una planilla de riesgo será necesario completar la planilla de riesgos, documentando cada columna según se indica en la tabla anterior.

Tipos de desviaciones

Existen diferentes tipos de desviaciones que se pueden presentar:

- **Desviaciones en las necesidades del cliente:** Pueden ser causados por un mal entendimiento de los requisitos del cliente, requisitos mal entregados, no reflejan la real necesidad, mala interpretación, ambiguos.
- **Desviaciones en la construcción:** Causados básicamente por una mala arquitectura, codificación deficiente y falta de control. Aquí las pruebas son necesarias para validar la arquitectura y otros componentes como el código.

Si existen riesgos que pueden ser categorizados como positivos, habrá que explotarlos al máximo para obtener beneficios.

Una vez identificado el riesgo, cualquiera sea este, se debe documentar, para ello utilizaremos una plantilla para identificación de riesgos, la cual se adjunta al material del curso.

Este registro debe claramente identificar el riesgo, su probabilidad e impacto, magnitud, entre otros datos como responsable y plan de mitigación, ver figura 11.

Tabla 3

Plantilla de riesgo

Riesgo	En que consiste	Impacto, probabilidad y magnitud	Manejo y Mitigación
RG1: Caída del servidor principal	Nuestro servidor principal puede tener alguna falla producto de un parche aplicado, una baja de voltaje, etc.	El Impacto sería ALTO , la probabilidad sería MEDIA , el Impacto sería ALTO .	Es un riesgo cuya magnitud para el negocio podría ser alta, por lo que se propone como medida de mitigación, dar disponibilidad a un nuevo servidor que esté funcionando en paralelo al principal.

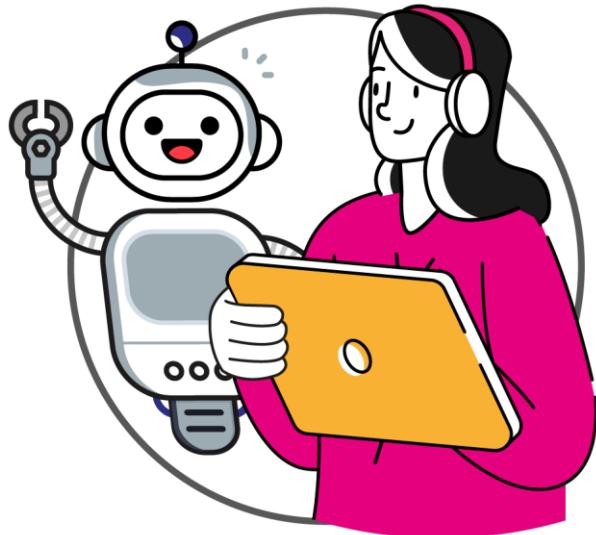
			El encargado del monitoreo e implementación será el área de operaciones.
RG2: Problemas de disponibilidad financiera del cliente.	El modelo arquitectónico definió Oracle como base de datos, sabemos que el valor de las licencias de la base de datos tiene un alto costo, lo que podría impactar en la renovación de estas en caso de tener problemas financieros.	El Impacto sería ALTO , la probabilidad sería BAJA, el impacto sería Medio.	En esto caso, si bien es cierto, es lejano, por lo que el plan de mitigación sería que el diseño pudiese incorporar una base de datos Oracle Community la cual es libre de costo o bien buscar una alternativa de bajo costo.

Nota. Tabla que muestra la planilla de riesgo, pero haciendo énfasis en aspectos como: en qué consiste, el impacto que tiene, su probabilidad, magnitud y el manejo y mitigación.

Cierre de la semana

Durante esta segunda semana, has aprendido acerca de la arquitectura de software, reforzando conceptos como modelos, también aprendiste que un software, debe generar la documentación pertinente y adecuada a los distintos puntos de vista, lo cual revisamos con el modelo de vistas 4+1, donde se explicó el concepto de vistas y diagramas, entre las cuales explicamos la vista lógica, vista de proceso, vista de despliegue, vista física y vista de escenarios, finalizando la semana, tuviste la oportunidad de aprender sobre riesgos del proyecto y como poder evaluar el impacto que tendríamos si alguno de estos riesgos se materializara.

Esperamos haya sido una gran semana de aprendizaje, los espero en la siguiente semana, donde seguiremos profundizando en los diferentes temas referidos a la arquitectura de software, tengan una buena semana.



Referencias

- Bass, L., Clements, P., & Kazman, R. (2006). *Software Architecture in Practice* (2nd ed.). Addison-Wesley.
- Cervantes Humberto, Velasco Perla, y Castro Luis. (2016). *Arquitectura de Software: Conceptos y desarrollo*. Primera edición. Cengage Learning Editores, SA.
- Morales, P. (2020, April 21). Vista lógica [Video].
<https://www.youtube.com/watch?v=Qlh989ihCmQ>
- Morales, P. (2020, Junio 15). *Vista de Escenarios* [Video]. YouTube.
<https://www.youtube.com/watch?v=kZYcgl96cJ4>
- Morales, P. *Vista de Procesos*. (2020, marzo 17).
https://www.youtube.com/watch?v=yPp_jGFnnMY&list=PLjK8W0kxXIDaUwKGNKTjvj0RzqJOR9bwE&index=3
- Morales, P. (2020, Aug 3). *Vista de Despliegue* [Video]. Youtube.
<https://www.youtube.com/watch?v=xkp4CLtlpb0>
- Morales, P. *Vista Física* [Video]. YouTube.
<https://www.youtube.com/watch?v=f0lG-QY7ciQ&list=PLjK8W0kxXIDaUwKGNKTjvj0RzqJOR9bwE&index=4>
- Pressman, R. S. (2006). *Ingeniería del software: Un enfoque práctico* (7th ed.). McGraw Hill.
- Rozanski, N., & Woods, E. (2005). *Software Systems Architecture*. Addison-Wesley.

Lecturas de la semana

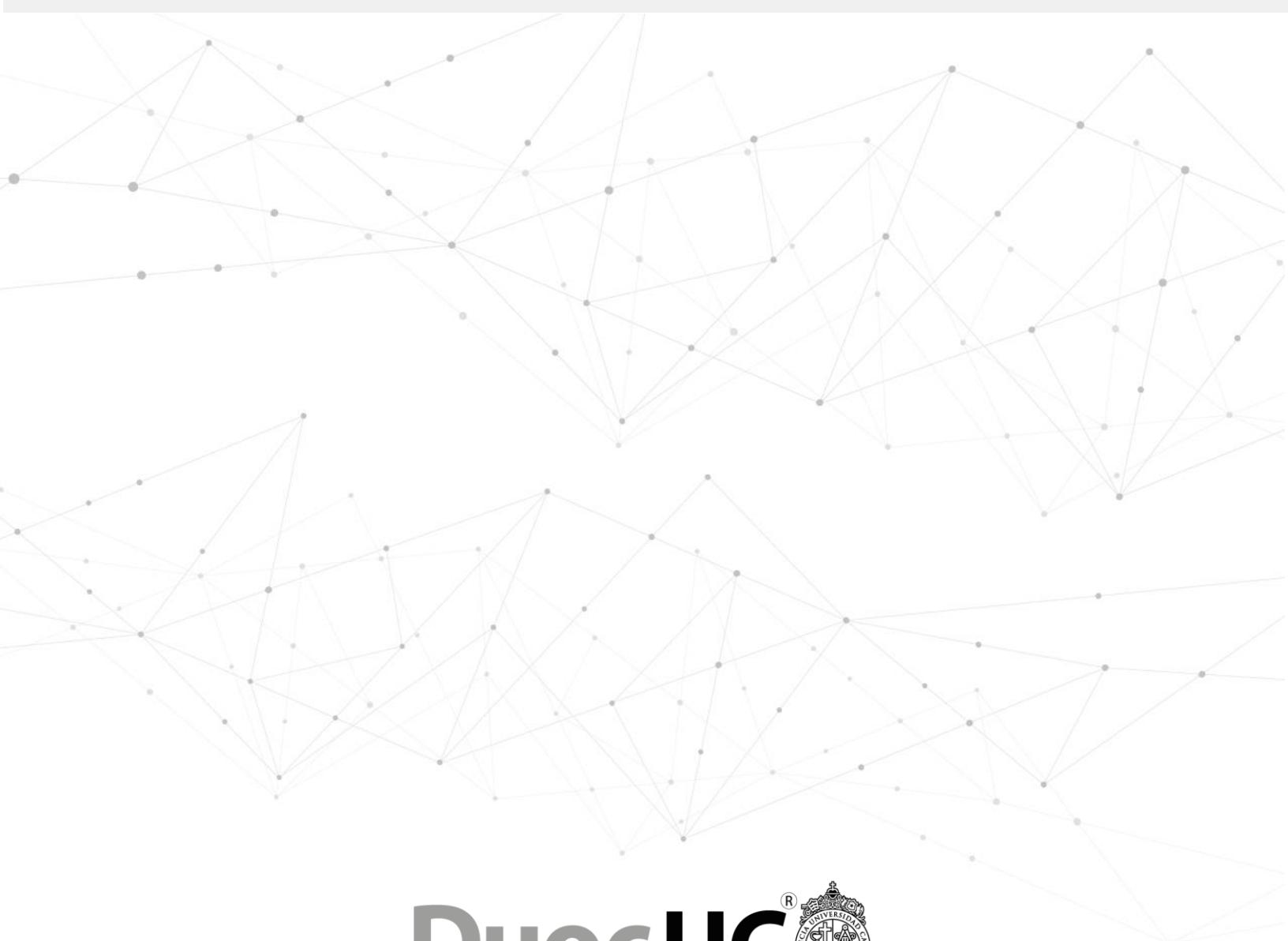
- Vistas arquitectónicas. Patrones arquitectónicos

Fuente: Sommerville, I. (2011). Ingeniería de software. México, D. F. Pearson, Páginas 153-158

- Capítulo 8: Modelado de requerimientos: Un enfoque recomendado

Fuente: Pressman, R. S. (2021). *Ingeniería del Software: Un enfoque práctico* (9th ed.). McGraw Hill. Páginas 126 a 155.

Apunte



DuocUC[®] 
ONLINE

Duoc UC

Facilitador disciplinar: Juan Alberto Gana.

Asesor par: Domingo Sanz.

Reservados todos los derechos Fundación Instituto Profesional Duoc UC. No se permite copiar, reproducir, reeditar, descargar, publicar, emitir, difundir, de forma total o parcial la presente obra, ni su incorporación a un sistema informático, ni su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia, grabación u otros) sin autorización previa y por escrito de Fundación Instituto Profesional Duoc UC La infracción de dichos derechos puede constituir un delito contra la propiedad intelectual.



Carrera
**Analista Programador
Computacional**

Exp 1 – Semana 3

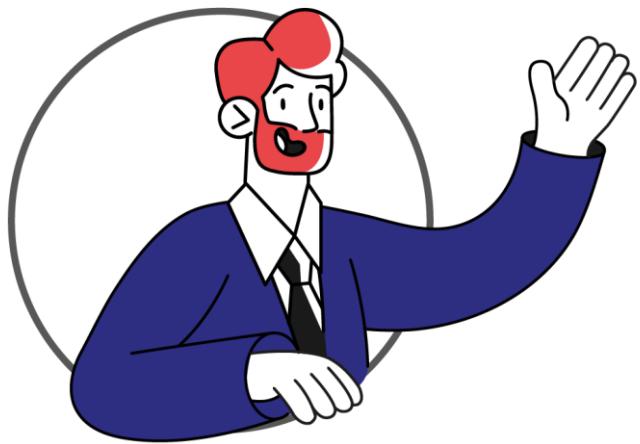
Arquitectura – ASY4231

Guía de aprendizaje

Índice

Introducción a la semana.....	3
Resultado de aprendizaje	4
El estudiante será capaz de:	4
Indicador de logro:.....	4
Conceptos relevantes.....	5
Preguntas activadoras.....	5
Actividad	5
Determinando los requisitos funcionales y atributos de calidad que iniciarán el proceso del modelo arquitectónico.....	6
Determinando los atributos de Calidad.....	7
Documentando los escenarios de Calidad	8
Desarrollando las Vistas 4+1	10
Vista de escenarios	10
Vista Lógica.....	12
Vista de Procesos.....	16
Vista de desarrollo.....	17
Vista física	19
Análisis de riesgos.....	21
Cierre de la semana	23
Referencias	24
Apunte	25

Introducción a la semana



En esta semana realizaremos una actividad sumativa en la cual pondremos a prueba lo aprendido en las semanas anteriores.

A partir del caso entregado sobre el sistema de ventas, sus requisitos funcionales y atributos de calidad, vamos a generar los escenarios de calidad iniciales que evaluaremos, para posteriormente generar una serie de diagramas que generan la

implementación de las vistas 4+1, finalmente iniciaremos la gestión de riesgos del proyecto.

Como primer paso, analizaremos la planilla de requisitos simplificados, y determinaremos cuales requisitos funcionales iniciarán el proceso de construcción, posteriormente haremos la generación de escenarios de calidad, siguiendo con la diagramación de cada elemento y finalizaremos analizando los riesgos de la arquitectura inicial que estamos proponiendo.

Resultado de aprendizaje

El estudiante será capaz de:

RA1. Diseña un modelo arquitectónico para soportar la solución de acuerdo a su factibilidad y a los estándares de la industria.

Indicador de logro:

IL1. Selecciona el modelo arquitectónico, de acuerdo a los requerimientos funcionales y no funcionales de la organización.

IL2. Identifica los atributos de calidad a partir de los requisitos no funcionales y necesidades de los stakeholders

IL3. Implementa el modelo arquitectónico seleccionado a través de la diagramación del modelo de vistas 4+1, para soportar los atributos de calidad requeridos de la organización.

IL4. Evalúa las consecuencias de los riesgos identificados del modelo diseñado, de acuerdo a la arquitectura propuesta.

Conceptos relevantes

	Modelo	Riesgo	Punto de vista
Diagrama	Impacto	Magnitud	
Caso de Uso	Escenarios	Modelo arquitectónico	

Preguntas activadoras

- ¿Qué es un modelo?
- ¿Qué es un punto de vista?
- ¿Qué es un riesgo en un proyecto?

Actividad

En esta tercera semana realizará la actividad grupal sumativa de la Experiencia 1, está dividida en 3 partes, para la Parte I deberán entregar la documentación requerida será el E.R.S (Especificación de Requisitos de Software), planilla de requisitos funcionales y no funcionales, escenarios de calidad, diagramas de arquitectura, planilla de identificación de riesgos del proyecto, con los aportes solicitados en las semanas anteriores. Para la Parte II deberán presentar el trabajo realizado.

Para la Parte III, deberán generar un espacio de reflexión sobre los aprendizajes adquiridos de manera individual. Para la Parte III, deberán generar un espacio de reflexión sobre los aprendizajes adquiridos de manera individual.

Determinando los requisitos funcionales y atributos de calidad que iniciarán el proceso del modelo arquitectónico.

A partir del caso entregado, donde nuestro cliente desea implementar mejoras tecnológicas en su modelo de ventas, seleccionaremos los siguientes requisitos funcionales (Tabla 1):

Tabla 1:

Requisitos funcionales.

ID	Requisito funcional	Actor	Descripción
RF001	Registro de usuarios	Cliente	El cliente debe poder registrarse en el sistema.
RF002	Iniciar sesión	Cliente	El cliente puede autenticarse en el sistema
RF003	Gestión de productos	Administrador	El administrador puede crear la ficha de los productos

Nota. La tabla muestra los requisitos funcionales, con su ID, el tipo, el actor y su descripción.

Del mismo modo, seleccionaremos los atributos de calidad (requisitos no funcionales, Tabla 2)

Tabla 2*Requisitos no funcionales*

ID	Descripción
RNF005	El sistema debe estar disponible las 24 horas del día, los 7 días de la semana, con un tiempo de inactividad mínimo planificado para tareas de mantenimiento programadas.
RNF006	Se deben implementar medidas de tolerancia a fallos para garantizar que el sistema siga funcionando en caso de errores o fallas en los componentes.

Nota. La tabla muestra los requisitos no funcionales con su ID y descripción.

Determinando los atributos de Calidad

El Cliente ha determinado los requisitos no funcionales RNF005 y RNF006, los cuales hacen mención a que el sistema debe estar disponible las 24 horas y además que se implementen medidas en caso de fallas.

Para ello usaremos los atributos de calidad:

- **Disponibilidad:** Grado en el que los datos tienen atributos que permiten ser obtenidos por usuarios y/o aplicaciones autorizadas en un contexto de uso específico [ISO25010, ISO25010.COM].

En nuestro caso definiremos la Disponibilidad como el tiempo en que el sistema debe estar funcionando, en el caso del cliente, ha solicitado que el sistema esté funcionando las 24 horas del día,

- **Tolerancia a fallos:** Capacidad del sistema o componente para operar según lo previsto en presencia de fallos hardware o software. [ISO25010, ISO25010.COM].

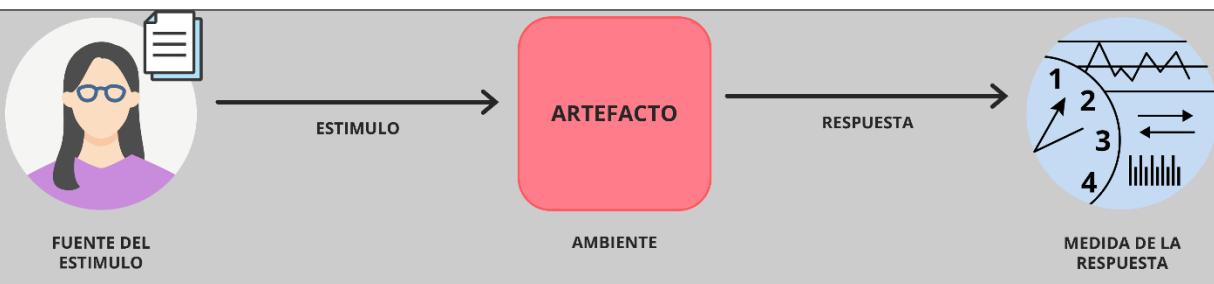
En nuestro caso definiremos Tolerancia a fallo como un mecanismo en el cual, si falla alguna parte de la infraestructura, esta deberá estar lista para operar con algún mecanismo secundario.

Documentando los escenarios de Calidad

A continuación, se documentan los 2 escenarios de calidad que soportan los atributos de calidad solicitados por el cliente: (Tabla 3 y Tabla 4)

Tabla 3:

Escenario de calidad 1

Escenario de Calidad N° 1 – RNF005		
Atributo de Calidad Asociado (Característica): FIABILIDAD		
Sub característica: DISPONIBILIDAD		
<p>Descripción: Se requiere que el cliente pueda ingresar a la aplicación WEB en cualquier momento para realizar las compras, esto puede ocurrir en cualquier horario, es decir el sistema debe estar disponible las 24 horas del día.</p> <p>Excepcionalmente se dejará una ventana para mantenimientos, la cual será analizada por el equipo de gestión de cambios.</p>		
 <p>FUENTE DEL ESTIMULO</p>	<p>ESTIMULO →</p> <p>ARTEFACTO</p> <p>AMBIENTE</p> <p>RESPUESTA →</p>	<p>MEDIDA DE LA RESPUESTA</p>
Fuente del Estímulo:	El Cliente ingresa al sitio para realizar una transacción comercial.	
Estímulo:	El cliente revisa los productos y agregar un producto al carrito de compras.	
Artefacto:	La página de ventas, perteneciente al módulo de ventas web.	
Ambiente:	Se requiere simular la peor condición, una semana con Cyber Day.	
Respuesta:	El sistema debe estar disponible cuando el cliente lo requiera.	

Medida de Respuesta:	INMEDIATA.
-----------------------------	------------

Nota. Tabla que muestra el escenario de calidad 1, correspondiente al Atributo de Calidad Asociado (Característica): FIABILIDAD

Sub característica: DISPONIBILIDAD

Tabla 4

Escenario de calidad 2

Escenario de Calidad N° 2 – RNF006

Atributo de Calidad Asociado (Característica): FIABILIDAD

Sub característica: TOLERANCIA A FALLOS

Descripción: Ante la eventual falla del servidor web (Servidor de aplicaciones), deberá el sistema ser operado en un segundo servidor, el cual debe estar en línea y preparado para entrar en el proceso como servidor primario.

FUENTE DEL ESTIMULO	ESTIMULO	ARTEFACTO
		AMBIENTE
		RESPUESTA
		MEDIDA DE LA RESPUESTA
Fuente del Estímulo:	Una falla eléctrica provoca que el servidor de aplicaciones falle.	
Estimulo:	Caída de la energía eléctrica.	
Artefacto:	Servidor de aplicaciones principal.	
Ambiente:	Ambiente productivo en semana de Cyber Day.	
Respuesta:	Un segundo servidor debe estar en línea.	
Medida de Respuesta:	INMEDIATA.	

Nota. Tabla que muestra el escenario de calidad 2, correspondiente al Atributo de Calidad Asociado (Característica): FIABILIDAD. Sub característica: TOLERANCIA A FALLOS

Desarrollando las Vistas 4+1

Una vez seleccionado los requisitos funcionales, los atributos de calidad y hemos generado los escenarios de calidad requeridos, estamos en condiciones de generar nuestros diagramas de Vistas 4+1.

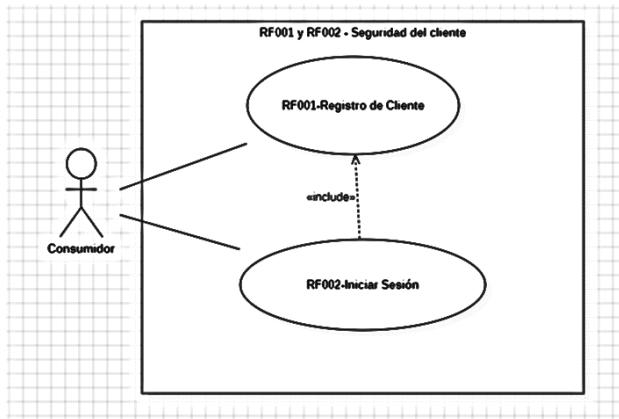
Vista de escenarios

Diagrama de casos de uso

El diagrama de casos de uso como se aprecia en la figura 3, nos muestra las funcionalidades del sistema, en otras palabras, representa a los requisitos funcionales. Los casos de uso no deben expresar una secuencia de pasos, solamente la acción principal. Los casos de uso pueden estar relacionados entre sí, y dependiendo del grado de compromiso entre ellos, su conexión puede ser **<<Include>>** u obligación, donde el caso de uso podría ejecutarse mal o tener un resultado no deseado si el otro caso de uso no se ejecuta, en la figura 4, se muestra que la relación entre el caso de uso RF002, depende del caso de uso RF001, o en otras palabras, un usuario no puede iniciar sesión si no se ha registrado primero. Si la dependencia de los casos de uso es opcional entonces se diagrama como **<<Extend>>**, donde un caso de uso padre hereda parte de si en el otro, ejemplo, Pagar la cuenta, con debito o con crédito, los 2 últimos son extensiones del pago, pero si no pago con crédito puedo pagar con débito u otro medio de pago.

Figura 1

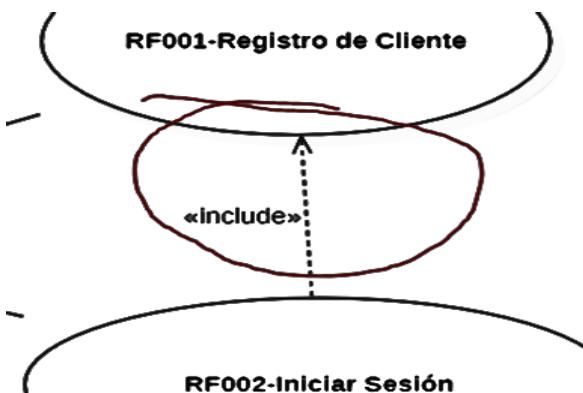
Casos de uso



Nota. Diagrama de casos de uso.

Figura 2

Dependencia entre casos de uso



Nota. Relación de dependencia entre casos de uso.

Vista Lógica

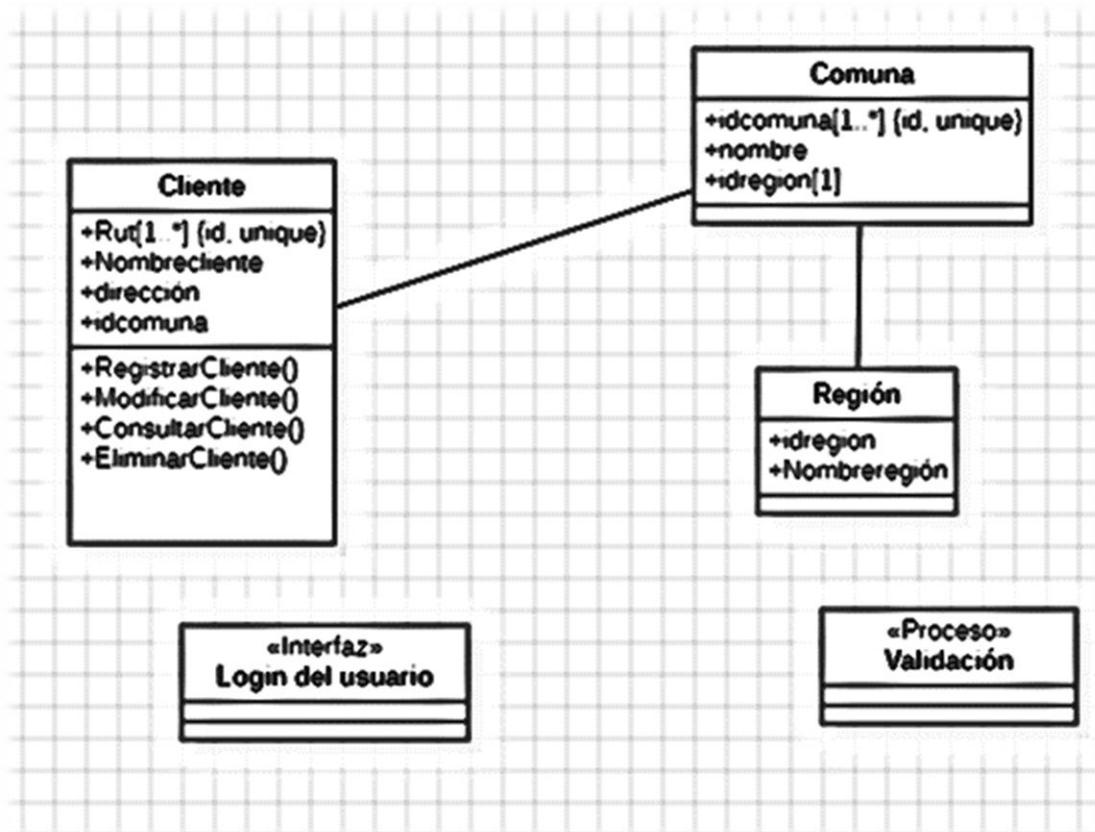
Diagrama de clases

En este caso hemos detectado que, para el RF001, se requiere crear la ficha del cliente, por lo que vamos a crear una clase Cliente, la cual contendrá los datos del cliente, la clase comuna, ya que un cliente reside en una sola comuna (en nuestro caso), y la comuna pertenece a solo una región, por lo que creamos la clase región.

Además, crearemos una clase de tipo interfaz, la cual será el formulario de ingreso de los datos del cliente y la clase validación la cual será un proceso que validará que el usuario no existe con anterioridad, tal como se muestra en la Figura 3

Figura 3

Diagrama de clases



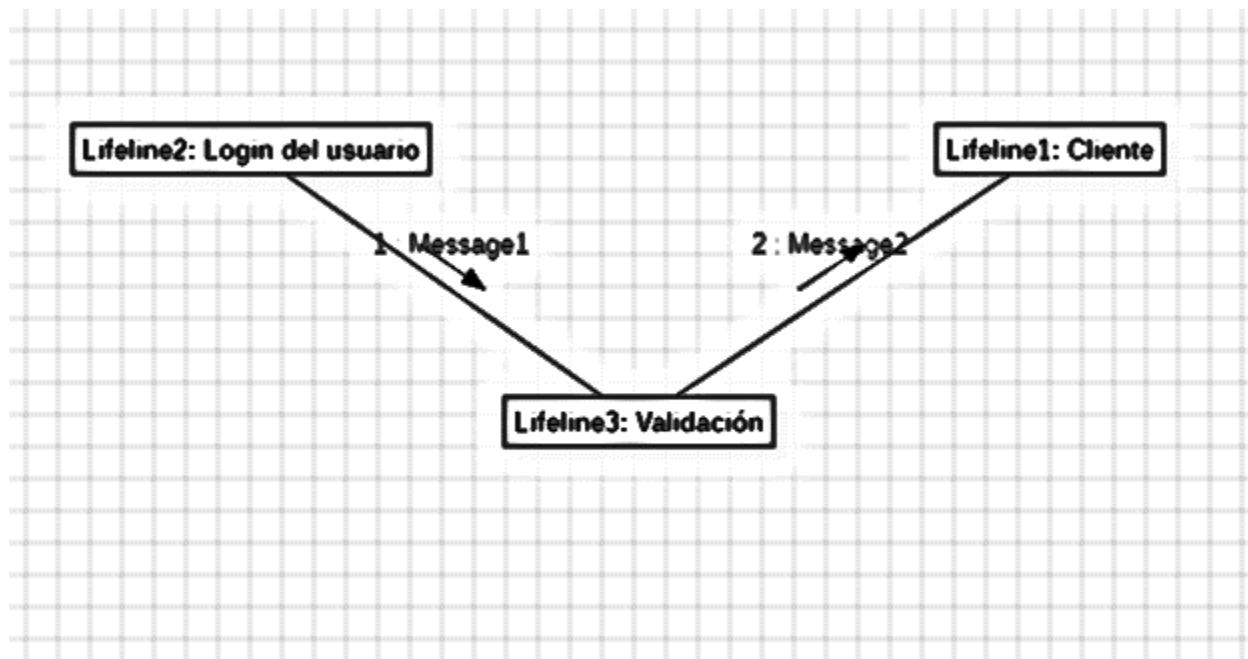
Nota. Diagrama de clases.

Diagrama de comunicaciones

El diagrama de comunicaciones como hemos aprendido nos muestra las interrelaciones entre las clases creadas, para nuestro ejemplo se adjunta el siguiente diagrama de comunicación, el cual indica la clase interfaz del inicio de sesión, comunicándose con la clase que, valida al usuario, finalmente la clase de validación le pregunta a la clase Cliente por la existencia del usuario, esto antes de crear la ficha del usuario, Figura 4

Figura 4

Diagrama de comunicaciones



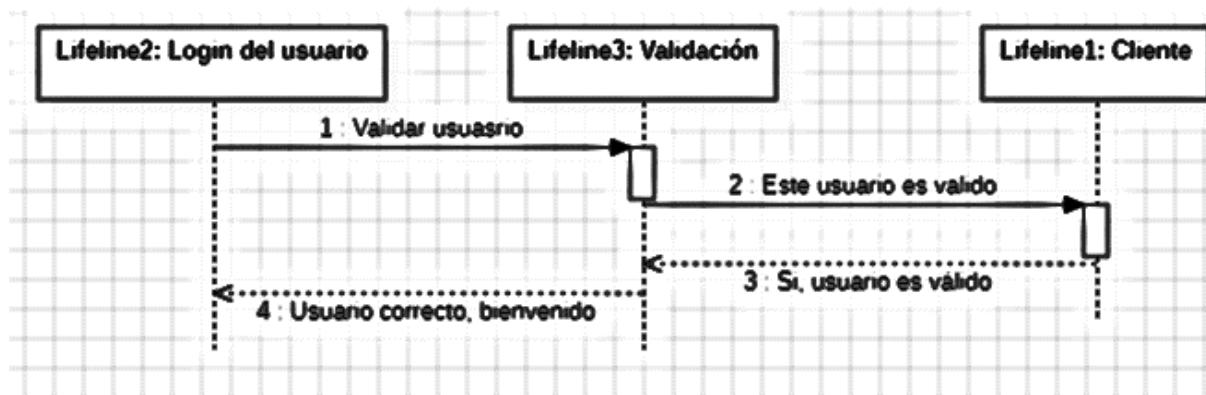
Nota. Diagrama de comunicaciones.

Diagrama de secuencia

El diagrama de secuencia nos indica en que instante del proceso las clases de comunican entre sí, en el ejemplo de la Figura 5, observamos el proceso de validación del usuario, primero el usuario trata de iniciar sesión, ante lo cual se genera un mensaje a la clase de validación, la cual a su vez le consulta a la clase cliente (la cual ya fue creada como tabla del sistema), dependiendo si el usuario existe o no se generan respuestas a los mensajes enviados

Figura 5

Diagrama de secuencia



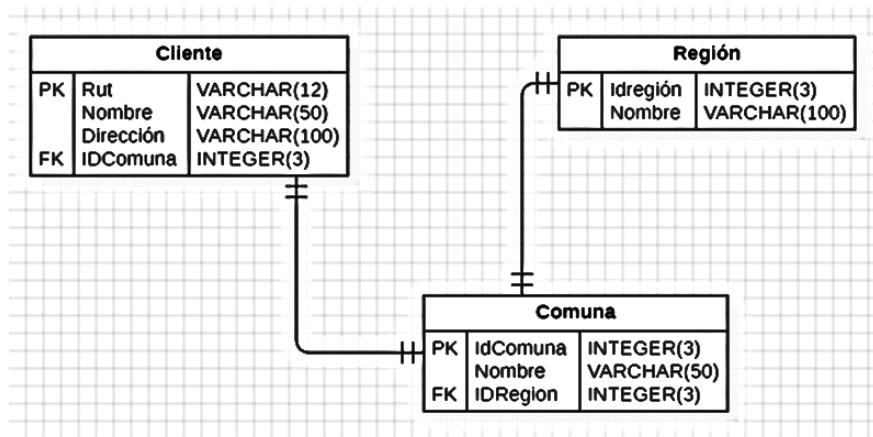
Nota. Diagrama de secuencia.

Diagrama de entidad relación

En la figura 6, podemos observar el resultado del proceso de convertir clases de datos (tablas de base de datos) a tablas en la base de datos, las cuales corresponden al modelo de base de datos relacional llamado comúnmente como MER. Se indican las llaves primarias y foráneas en caso de ser necesario.

Figura 6

Diagrama de entidad relación



Nota. Diagrama de entidad relación.

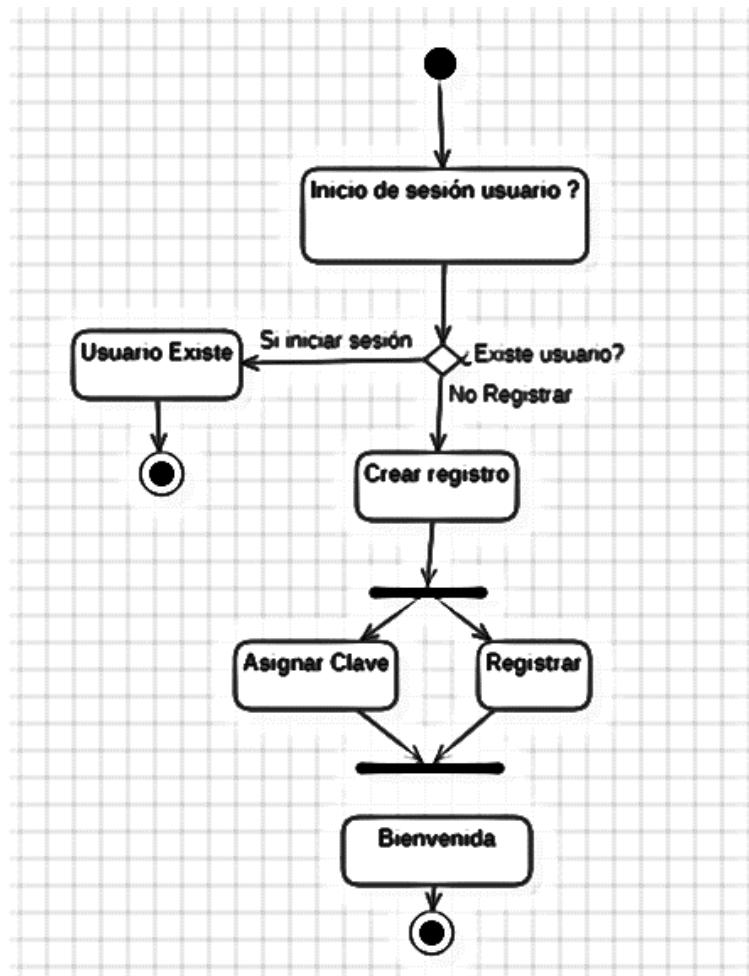
Vista de Procesos

Diagrama de actividad

El diagrama de actividades nos muestra la secuencia de pasos para completar una actividad, en el ejemplo de la figura 7, nos indica los pasos a seguir para el registro de un usuario nuevo, luego de solicitar los datos del usuario, si este existe el sistema termina el registro y lo invita a iniciar sesión, de lo contrario si no existe, realiza el proceso de creación de la cuenta.

Figura 7

Diagrama de actividad



Nota. Diagrama de actividades.

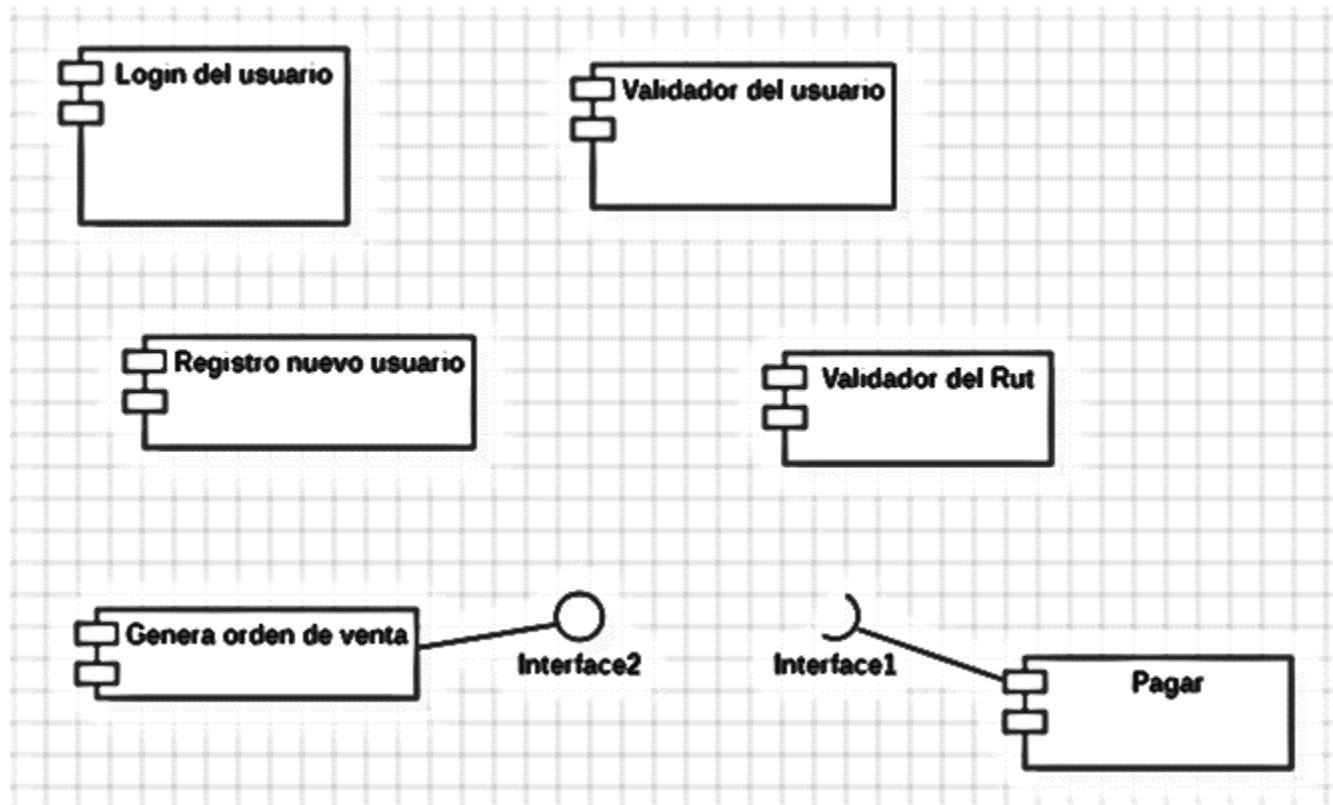
Vista de desarrollo

Diagrama de componentes

El diagrama de componentes muestra los componentes a crear, ya sean programas, funciones, procedimientos, interfaces, servicios web, etc., todo aquel componente que es parte del sistema, en la figura 8 se muestran algunos de los componentes que incluirá el sistema.

Figura 8

Diagrama de componentes



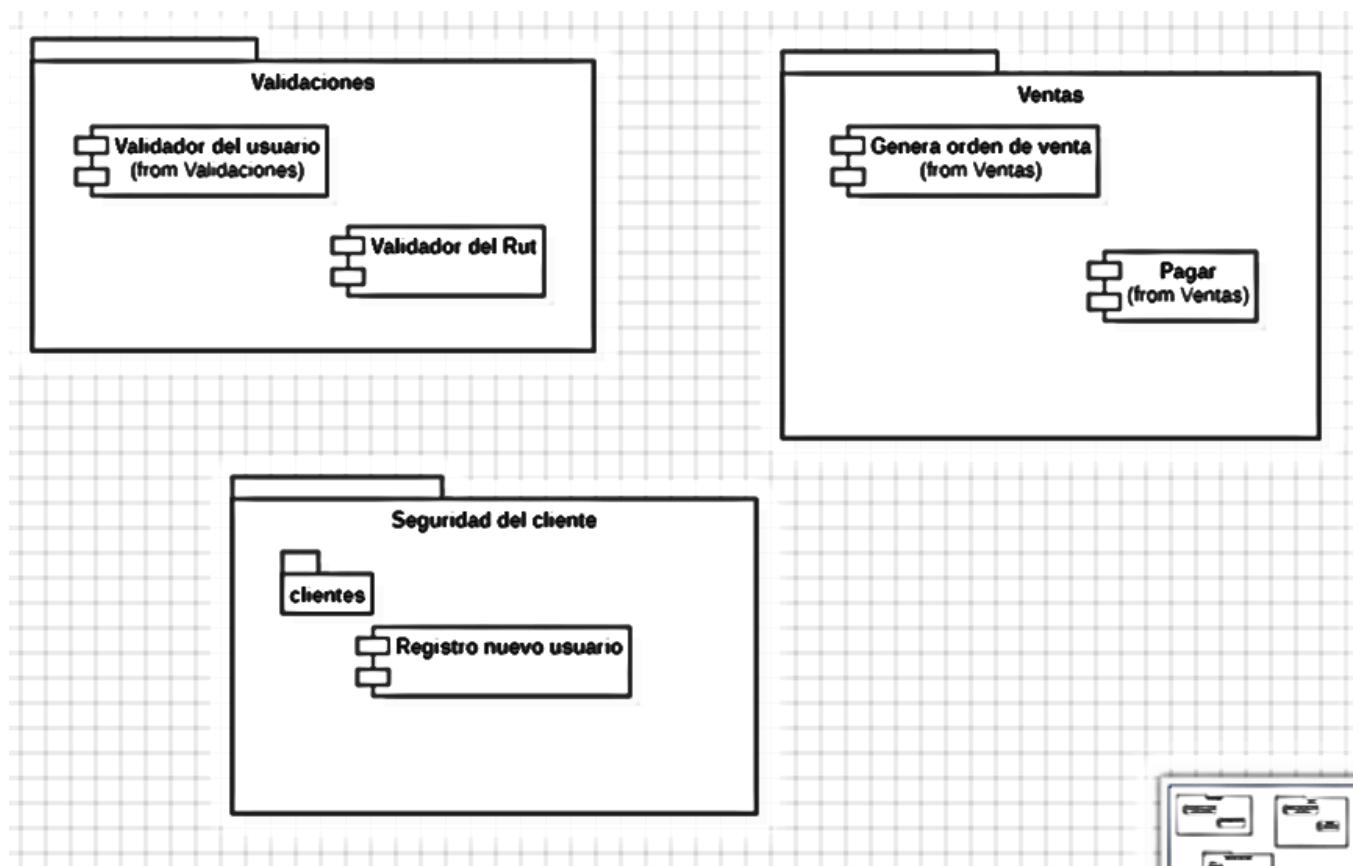
Nota. Diagrama de componentes.

Diagrama de paquetes

El diagrama de paquetes es el encargo de organizar los componentes, en este podemos encontrar los componentes del sistema, ordenados por temas, fases, etc., es un orden en general de todos nuestros objetivos del sistema, tal como se aprecia en la figura 9.

Figura 9

Diagrama de paquetes.



Nota. Diagrama de paquetes.

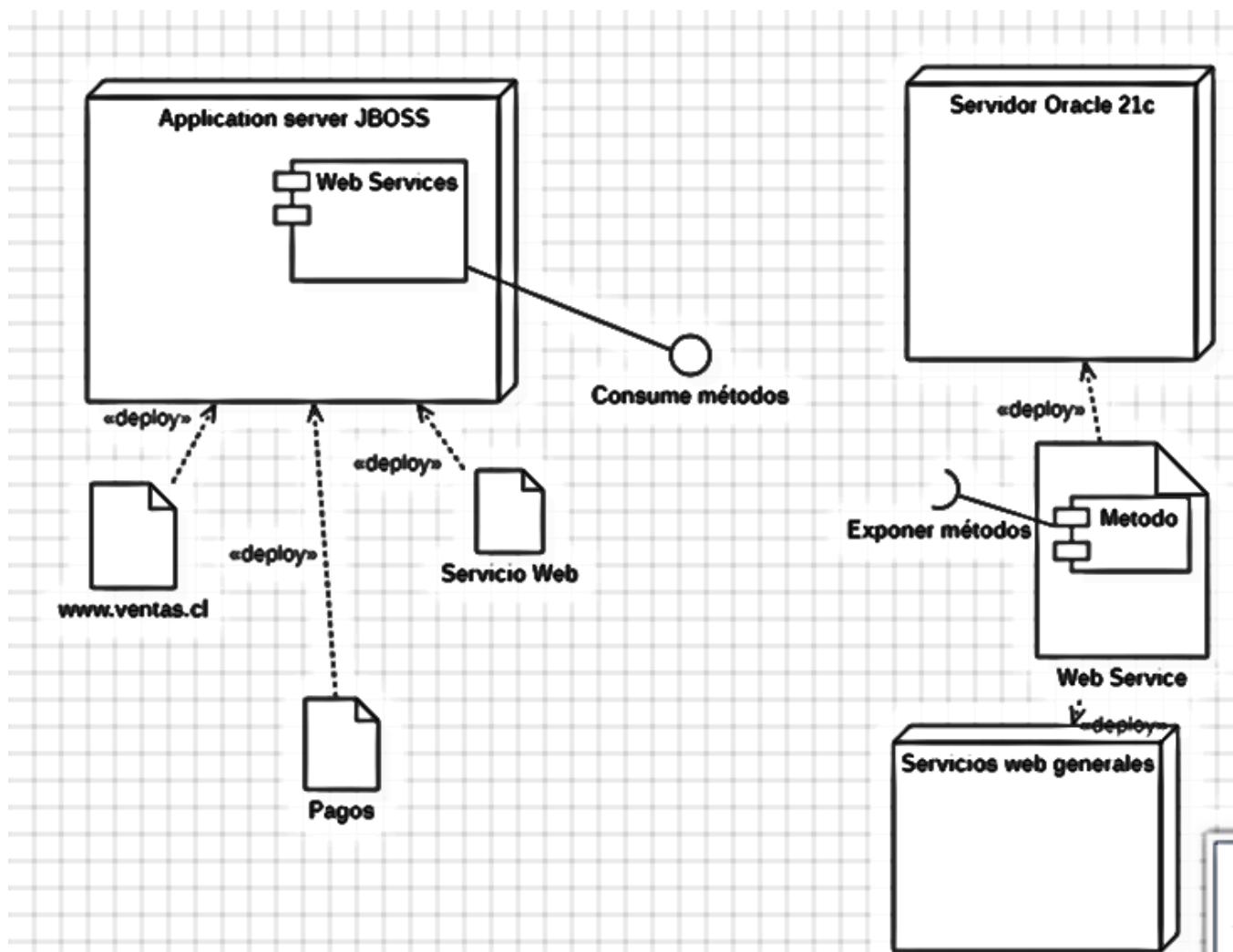
Vista física

Diagrama de topologías

El diagrama de topología nos muestra sobre la infraestructura en general, si hay servidores de aplicación, servidores de base de datos, componentes de interfaces, etc., la idea es diagramar el ambiente operativo del sistema en términos de la infraestructura, como se muestra en la figura 10.

Figura 10

Diagrama de topología



Nota. Diagrama de topología.

Importante

Las vistas 4+1 tienen como objetivo principal, llegar a cada persona clave en la construcción de la arquitectura y del sistema, en este sentido la vista lógica esta de cara a desarrolladores y arquitectos, la vista de procesos es usada por los Integradores, la vista de despliegue es usada por los desarrolladores, la vista física es orientada a los ingenieros de sistema y finalmente la vista de escenarios está orientada a analistas de sistemas, y a cualquier usuario que tenga relación con la definición de requisitos.

Análisis de riesgos

Primero recordemos que es un riesgo, son todas aquellas situaciones que nos podrían afectar el desempeño del proyecto o el sistema en caso de materializarse a futuro, un riesgo no se puede eliminar, solo podemos mitigarlo, transferirlo o bien aprender a vivir con este.

En nuestro ejemplo, visualizaremos los siguientes riesgos, Tabla 5.

Tabla 5

Riesgos del Proyecto

Riesgo	En que consiste	Impacto, probabilidad y magnitud	Manejo y Mitigación
RG1: Caída del servidor principal	Nuestro servidor principal puede tener alguna falla producto de un parche aplicado, una baja de voltaje, etc.	El Impacto sería ALTO , la probabilidad sería MEDIA	Es un riesgo cuya magnitud para el negocio podría ser alta, por lo que se propone como medida de mitigación, dar disponibilidad a un nuevo servidor que esté funcionando en paralelo al principal. El encargado del monitoreo e implementación será el área de operaciones.
RG2: Problemas de disponibilidad financiera del cliente.	El modelo arquitectónico definió Oracle como base de datos, sabemos que el valor de las licencias de la base de datos tiene un alto costo, lo que podría	El Impacto sería ALTO , la probabilidad sería BAJA	En esto caso, si bien es cierto, es lejano, por lo que el plan de mitigación sería que el diseño pudiese incorporar una base de datos Oracle Community la cual es libre de costo o bien buscar una alternativa de bajo costo.

	impactar en la renovación de estas en caso de tener problemas financieros.		
--	--	--	--

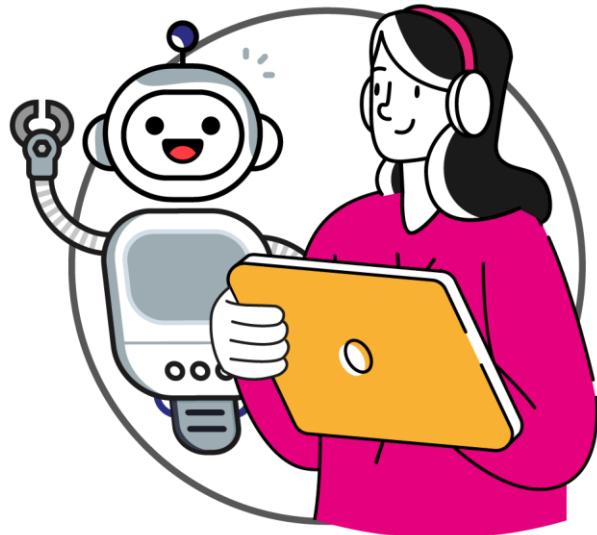
Nota. La tabla describe cada uno de los riesgos.

Como todo plan de riesgo, no basta con crearlo y compartirlo, cada cierto tiempo se debe revisar, buscar cambios en los riesgos identificado, actualizar con nuevos riesgos y/o procedimientos de mitigación.

Cierre de la semana

Durante esta tercera semana, hemos reforzado nuestros conocimientos , desarrollando un caso práctico para la implementación del modelo arquitectónico de un sistema de ventas, en este ejercicio hemos determinado los atributos de calidad a utilizar, construimos los escenarios de calidad, realizamos el proceso de diagramación de las vistas 4+1 y hemos determinado algunos riesgos de nuestras decisiones arquitectónicas, las cuales profundizaremos al final del curso en la evaluación de la arquitectura usando la metodología ATAM.

Espero haya sido una semana provechosa y los invito a completar esta actividad con el resto de los requisitos funcionales de caso de ventas que hemos desarrollado acá.



Referencias

- Riesgos, Análisis Cuantitativo y cualitativo, fuente [Liliana Buchtik, 2021],
https://www.youtube.com/watch?v=xx-l_nF7juM

Apunte



Duoc UC 
ONLINE

Duoc UC

Facilitador disciplinar: Juan Alberto Gana.

Asesor par: Domingo Sanz.

Reservados todos los derechos Fundación Instituto Profesional Duoc UC. No se permite copiar, reproducir, reeditar, descargar, publicar, emitir, difundir, de forma total o parcial la presente obra, ni su incorporación a un sistema informático, ni su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia, grabación u otros) sin autorización previa y por escrito de Fundación Instituto Profesional Duoc UC La infracción de dichos derechos puede constituir un delito contra la propiedad intelectual.



Carrera **Analista Programador Computacional**

Exp 2 – Semana 4

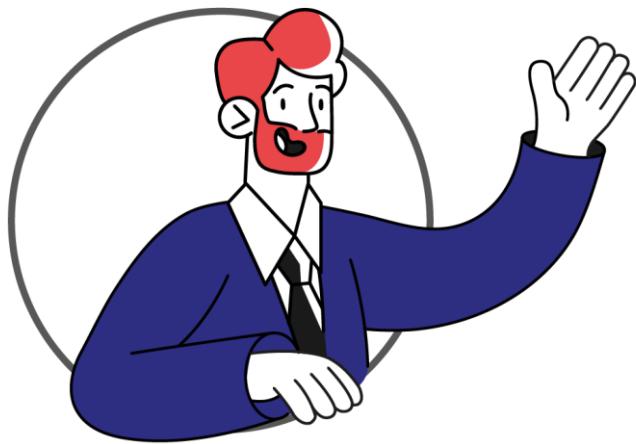
Arquitectura ASY4231

Guía de Aprendizaje

Índice

Introducción a la semana.....	.3
Resultado de aprendizaje	4
El estudiante será capaz de:	4
Indicador de logro:.....	4
Conceptos relevantes	4
Preguntas activadoras.....	4
Actividad	5
La vista lógica	6
Diagrama de Clases	6
Implementando la vista lógica en JAVA usando el IDE NetBeans 8.2	13
Implementando el sistema de ventas a través de la vista lógica	20
Cierre de la semana	22
Referencias	23
Apunte	24

Introducción a la semana



En esta cuarta semana pondremos en práctica lo aprendido sobre cómo documentar la arquitectura de nuestro sistema a través de los distintos diagramas de vistas 4+1.

En esta oportunidad vamos a generar los diagramas de clases, de comunicación y secuencia de la vista lógica, a través del uso de un lenguaje de programación orientado a objetos, en este caso será Java. Junto con

ello, veremos una alternativa para la creación de clases usando para ello el software starUML, el cual hemos estado usando para crear los distintos diagramas de vistas 4+1.

Usaremos para la experiencia, algunos de los requisitos funcionales entregados al inicio del curso y los pondremos en práctica.

Como material adicional, tendrás disponible el video “**Implementando vista lógica**”, el cual explicará cómo crear clases a través del software starUML, generando finalmente código Java para su implementación, que tengan una excelente semana de aprendizaje.

Resultado de aprendizaje

El estudiante será capaz de:

RA2. Implementa el modelo arquitectónico en el lenguaje de programación seleccionado para soportar la solución sistémica requerida por el cliente.

Indicador de logro:

IL5 Implementa la vista lógica del proyecto, en base a los requerimientos funcionales del proyecto.

Conceptos relevantes

	4+1	Objeto	Diagrama
	Comunicación	Secuencia	Clase
	Atributo	Operaciones	Relaciones

Preguntas activadoras

- ¿Qué son objetos, y lenguaje orientado a objetos?
- ¿Qué es el diagrama de comunicación y secuencia?
- ¿Qué son las clases?
- ¿Qué son los atributos?
- ¿Qué son las operaciones?

Actividad

Implementando la arquitectura - Vista lógica.

En la cuarta semana realizarán una actividad formativa con encargo grupal llamada "Implementando la arquitectura - Vista lógica" para generar el diagrama de clases, diagrama de secuencia y diagrama de comunicación.

La vista lógica

Diagrama de Clases

Como ya hemos aprendido, las clases son plantillas que permiten describir las características y sus comportamientos. Una clase posee datos los que son llamados atributos o propiedades, y del mismo modo posee funciones, llamadas métodos o acciones que la clase puede realizar.

Las clases no son instancias del objeto, son plantillas, las cuales posteriormente usaremos creando diferentes instancias de este objeto creado, ejemplo:

Clase **Cla_Tipo_Cliente**

Atributos

(Rut, nombre, edad)

Métodos

([CrearCliente](#), ModificarCliente, EliminarCliente)

Cla_Tipo_Cliente miCliente = NEW **Cla-Tipo-Cliente('12100908-7', 'Juan', 55);**

En este ejemplo, se crea la clase **Cla_Tipo_Cliente**, luego definimos una instancia de la clase llamada miCliente, la cual adopta los atributos y métodos de la clase, por lo cual usamos la clase para definir nuevas instancias de ella y aquellas instancias nuevas, son las que usaremos en los programas como tal.

En nuestro ejemplo de esta semana, crearemos las clases necesarias para cubrir los requerimientos RF001 y RF002 del caso de Ventas entregado al inicio del curso.

Recordemos, RF001, el cual corresponde a “Registro del usuario”, en este caso se requiere que un nuevo cliente pueda registrarse en el sistema, para ello necesitará ciertos datos para su registro como:

- Rut
- Nombre

- Correo
- Contraseña

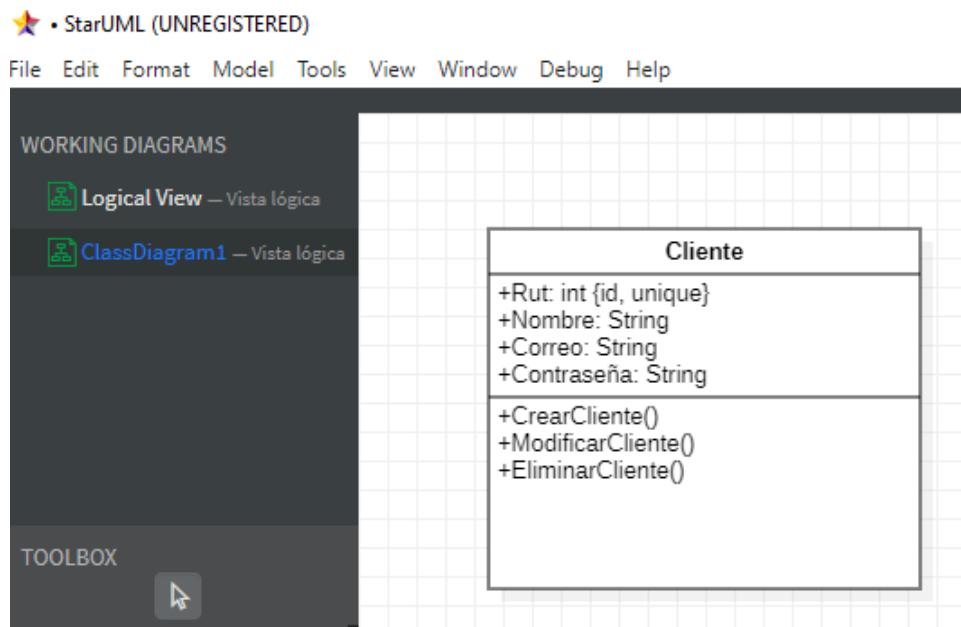
Primero, usaremos starUML para crear la clase

Paso 1: Usando starUML para crear la clase “Cliente”

Como ya hemos visto en starUML, cree la clase cliente, indicando que posee los atributos rut de tipo Int, nombre, correo y contraseña, estos últimos de tipo String, Figura 1.

Figura 1:

Clase Cliente.



Nota: En la clase cliente se despliegan los atributos +Rut: de tipo Int, +Nombre, +Correo y +Contraseña, estos últimos de tipo String. Abajo se despliegan +CrearCliente(), +ModificarCliente() y +EliminarCliente().

Paso 2: a partir del diagrama, implementar la clase en Java.

Seleccionar los atributos (Rut, nombre, correo y Contraseña) una a una, primero seleccione Rut, luego desde el menu principal, hacer clic en Tools, luego “**Generate Getters Setters**”, Figura 2, esto agregará los getters y setters requeridos para el atributo Rut, podrá visualizar el cambio en la estructura de la clase, como se muestra en la figura 3.

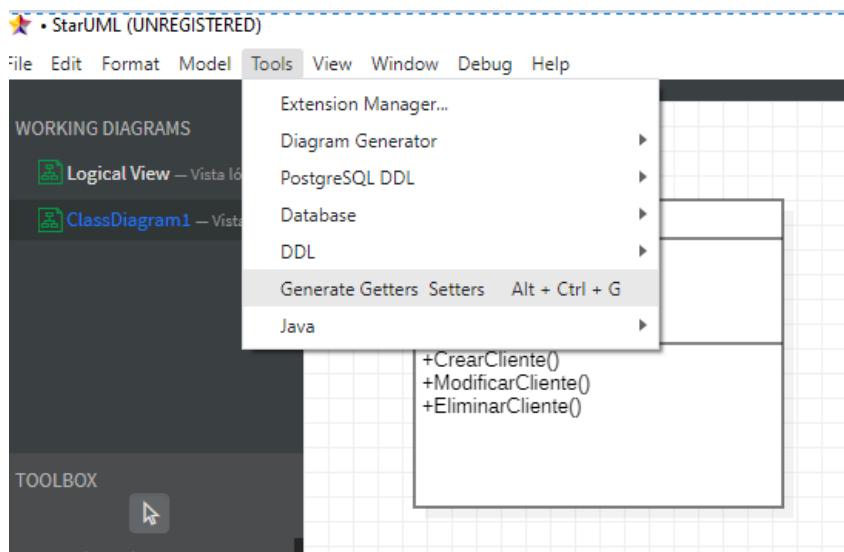
Importante

Getters y Setters son métodos que permitirán tener acceso a los atributos de las clases y así poder manipular sus valores desde el exterior de la clase, en un programa, por ejemplo.



Figura 2:

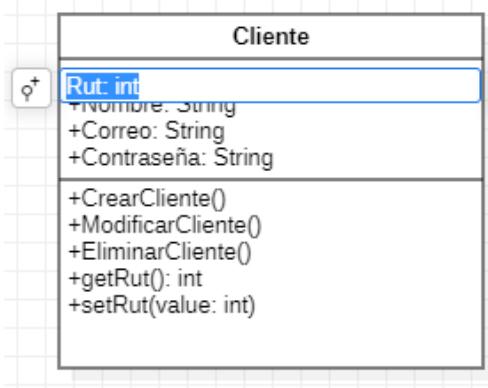
Generate Getters Setters.



Nota: En la pestaña Tools, se debe seleccionar “Generate Getters Setters”.

Figura 3:

Cambio en la estructura de clases.

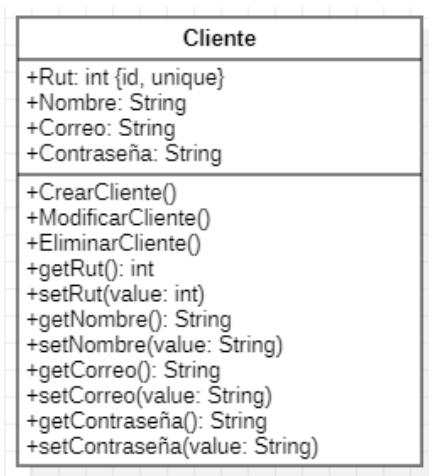


Nota: En la clase “Cliente”, podrás editar los atributos.

Repita estos pasos para todos los atributos de la clase, hasta que quede como en la figura 4

Figura 4:

Ejemplo de como debe quedar

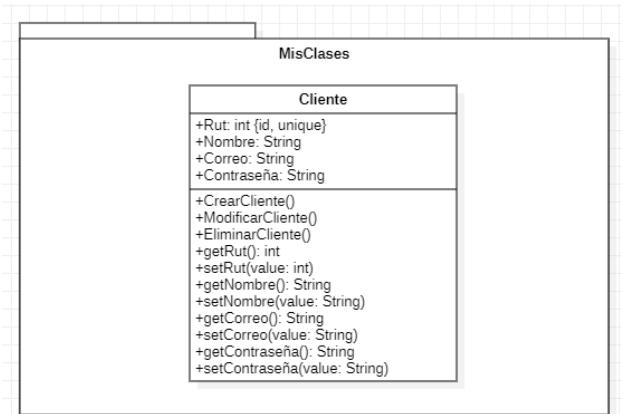


Nota: Los atributos de la Clase “Cliente” queda completa con +Rut: de tipo Int, +Nombre, +Correo y +Contraseña, estos últimos de tipo String. Abajo se despliegan +CrearCliente(), +ModificarCliente(), +EliminarCliente(), +getRut(value: int), +getNombre(): String, +getNombre(value:String), +getCorreo(): String, +setCorreo(value:String), +getContraseña(): String, +setContraseña():String, +setContraseña(value:String).

Es conveniente crear un objeto Package y luego dejar dentro de este la clase recien creada, como se ve en la figura 5.

Figura 5:

Package conteniendo a la clase Cliente.



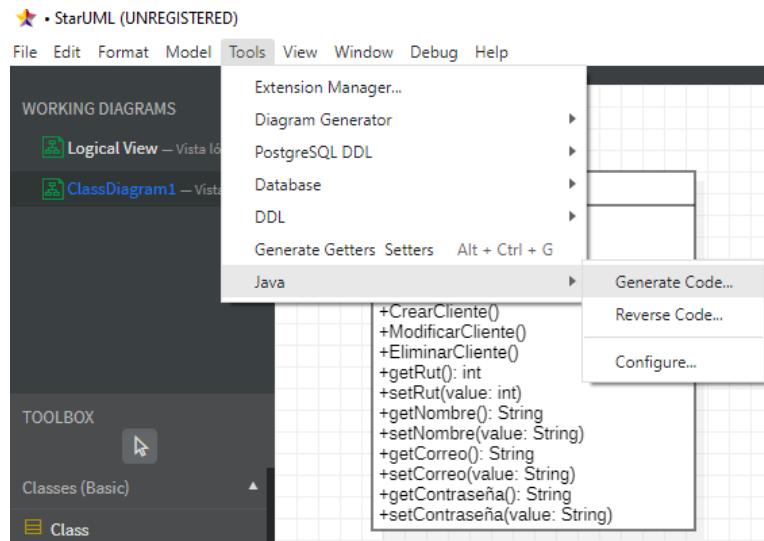
Nota: Los atributos de la Clase “Cliente” Están dentro de una carpeta.

Paso 3: Generar el código para JAVA.

Una vez finalizó con la definición de atributos, sus Getters y Setters, grabe el modelo starUML en una carpeta donde este documentando su sistema. Ahora, generar la clase en Java, para ello desde el menu principal, selecciones “Tools”, luego “Java” y “Generate Code”, tal como se aprecia en la figura 6.

Figura 6:

Generando código Java

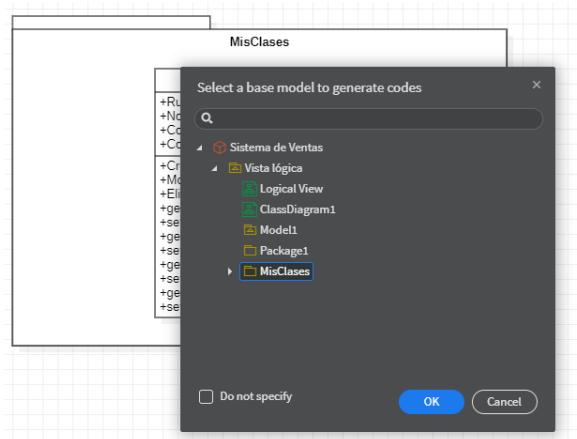


Nota: Dentro de la pestaña Tools, seleccionar “Java” y luego “Generate Code”.

StarUML, nos consultara que modelo vamos a generar, en este caso seleccionaremos el Package MisClases, luego indicar “OK”, Figura 7, esto abrirá una nueva ventana para solicitar en que carpeta dejaremos el código, ver Figura 8.

Figura 7:

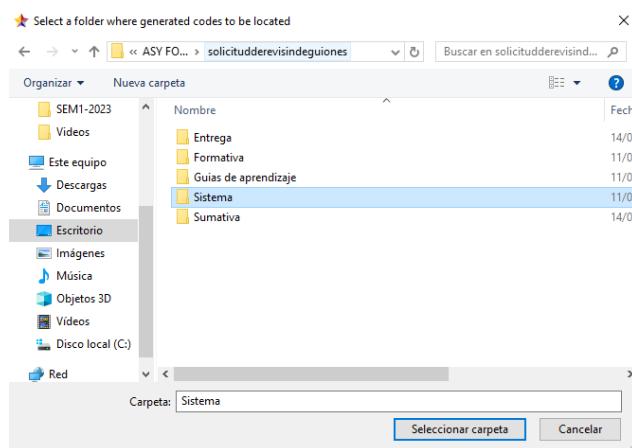
Selección del Package MisClases.



Nota: En Vista lógica selecciona Misclases.

Figura 8

Seleccionar carpeta “sistema”.



Nota: Seleccionar una carpeta donde dejar el código, en nuestro ejemplo escogeré “Sistema”.

Una vez indicada la carpeta, el código fue generado en dicha carpeta, con el explorador de Windows, dirijase a la carpeta y puede abrir el archivo usando un IDE o bien alguna herramienta para editar texto. El ejemplo generado se encuentra disponible como anexo al final de este documento en “ANEXO 1: Código JAVA para la clase Cliente”.

Importante



- Para profundizar un poco más sobre clases, revise el anexo 2, al final de este documento.
- Para revisar el paso a paso completo, revise el VIDEO “Generando clases”, el cual se encuentra disponible en el material de esta semana.

Implementando la vista lógica en JAVA usando el IDE NetBeans

8.2

A continuación, utilizaremos el IDE Netbeans 8.2, para desarrollar un ejemplo de cómo implementar la vista lógica en el lenguaje JAVA, para ello usaremos la clase Cliente, creada en el capítulo anterior, el cual a través de starUML creo un código con la clase.

En esta oportunidad, crearemos la clase directo desde JAVA para visualizar un nuevo método para implementarla.

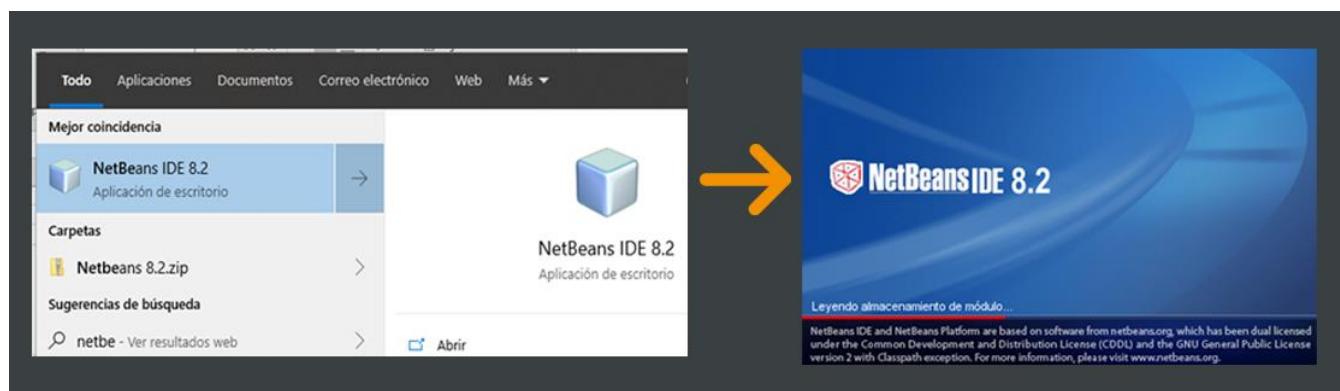
Paso 1: Abrir Netbeans 8.2

Como parte del material disponible, existe un video de como instalar Netbeans en tu máquina, además en el repositorio, está disponible el instalador de Netbeans 8.2 para que lo puedas utilizar rápidamente.

Abre desde Windows Netbeans 8.2, digitando desde la línea de comandos o a través del menú de Windows.

Figura 9

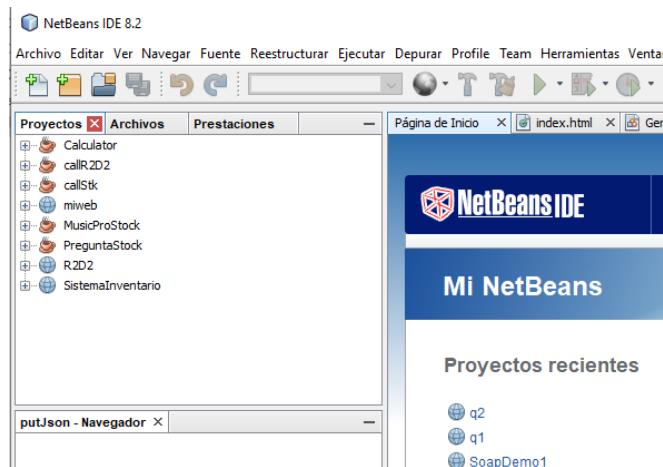
Abriendo Netbeans.



Nota: Al seleccionar el programa Netbeans se abrirá y se podrá visibilizar su pantalla de inicio.

Una vez en Netbeans, observaremos lo siguiente:

Figura 10
Netbeans.



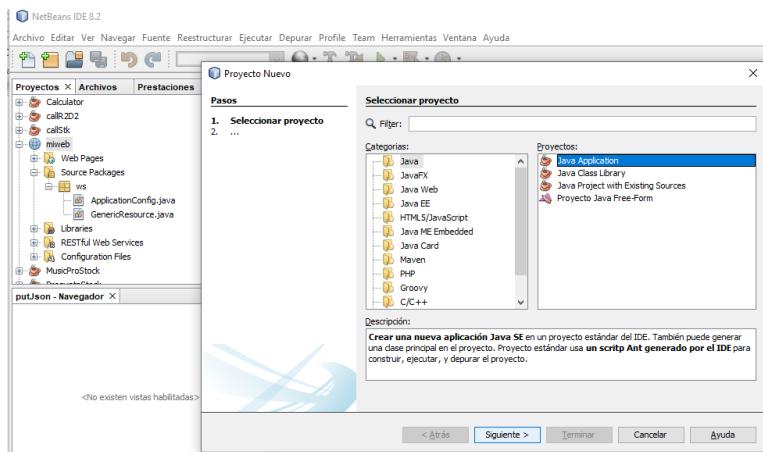
Nota: Al abrir el programa NetBeans podremos ver el menú completo y sus pestañas.

Paso 2: Crear una pequeña aplicación en Java.

Dirígete al menú “Archivo”, “Proyecto nuevo” y selecciona la categoría “Java” y un tipo de proyecto “Java Application”, como en la figura 11. Presiona el botón “Siguiente”.

Imagen 11:

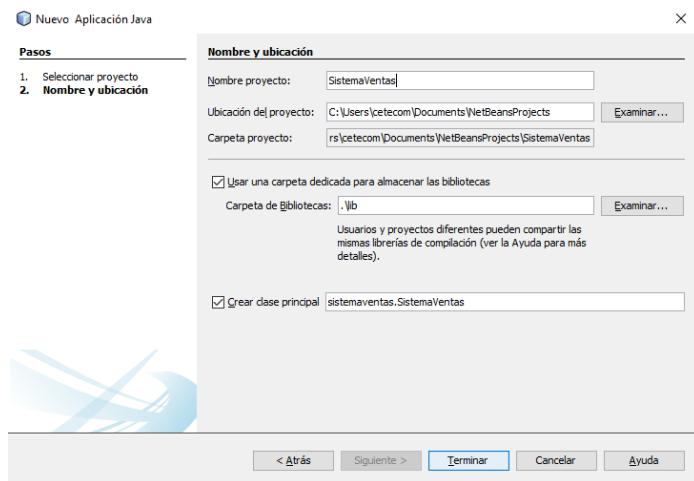
Generar un nuevo proyecto Java Applications.



Nota: Al dirigirse al menú “Archivo”, luego “Proyecto nuevo” y selecciona la categoría “Java” y un tipo de proyecto “Java Application”.

Da un nombre a tu proyecto, en mi caso seleccionaré “SistemaVentas”, presionar “Terminar”,

Figura 12:
Crear el proyecto.

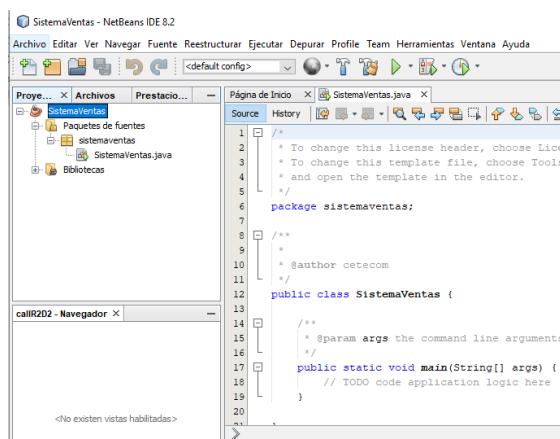


Nota: Da un nombre a tu proyecto, en mi caso seleccionaré “SistemaVentas”, presionar “Terminar”

Netbeans creará el proyecto y ya podemos comenzar a programar, como se ve en la figura 13.

Figura 13:

Proyecto listo para continuar desarrollando.



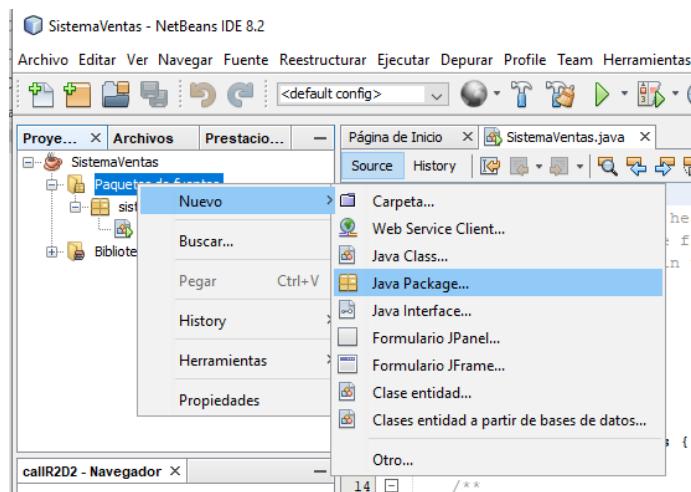
Nota: Ya se ve la pantalla para comenzar a trabajar en el proyecto y programar.

Paso 3: Crear la clase Cliente

Para crear clases en Java usando Netbeans, haz clic derecho en la carpeta “Paquetes de Fuentes”, luego en “Nuevo”, finalmente en “Java Package”, esto creará un paquete, ponle un nombre para continuar, en mi caso usaré “MisClases”, luego presionar “Terminar”. Se podrá apreciar el resultado en la figura 14.

Figura 14:

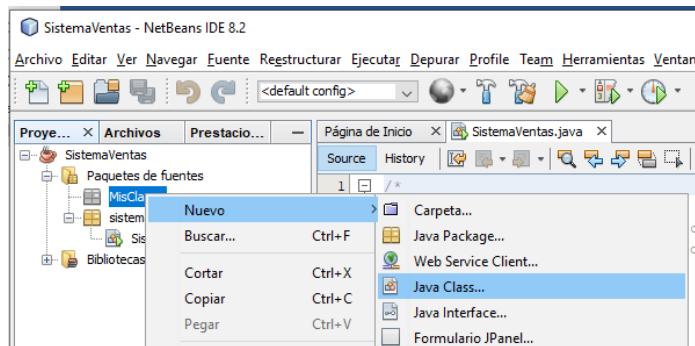
El Package “MisClases” fue creado.



Nota: Haz clic derecho en la carpeta “Paquetes de Fuentes”, luego en “Nuevo”, finalmente en “Java Package”, esto creará un paquete, ponle un nombre para continuar, en mi caso usaré “MisClases”, luego presionar “Terminar”.

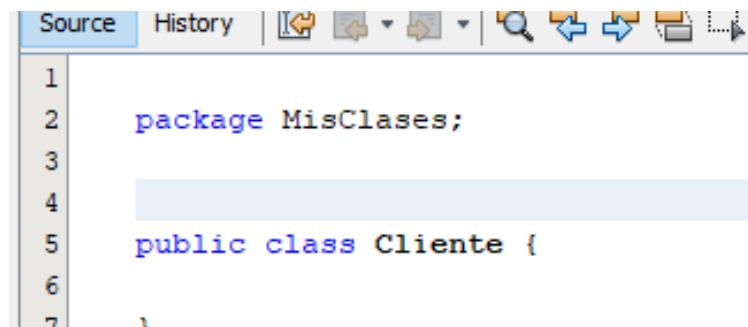
A continuación, vamos a crear una clase, para ello usaremos el nombre Cliente, la cual esta solicitada en el Requerimiento Funcional RF001.

Haz clic derecho sobre el Package “misClases”, luego “Nuevo” y “Java Class”, escribe el nombre cliente y luego el botón “Terminar”.

Figura 15:*Crear la clase Cliente*

Nota: Haz clic derecho sobre el Package “misClases”, luego “Nuevo” y “Java Class”, escribe el nombre cliente y luego el botón “Terminar”.

Esto creará y dejará disponible el editor para terminar la clase.

Figura 16.*La clase cliente creada.*A screenshot of the NetBeans Source editor. The tab bar at the top shows "Source" and "History". The toolbar has icons for file operations like New, Open, Save, Cut, Copy, Paste, Find, and Run. The code editor shows the beginning of a Java class: line 1 has "1", line 2 has "package MisClases;", line 4 has "4", line 5 has "5 public class Cliente {", line 6 has "6", and line 7 has "7". The code is color-coded with blue for keywords like "package", "public", and "class".

Nota: Editor disponible para terminar la clase.

Termina de escribir la clase, indicando los atributos y sus tipos de datos.

Figura 17:

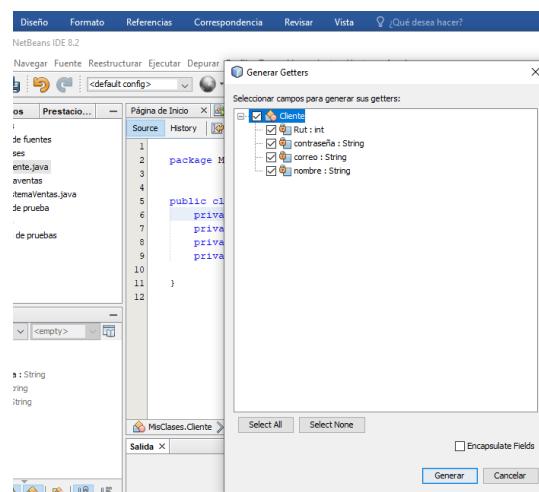
La clase contiene todos los atributos mencionados en el RF001.

Nota: Termina de escribir la clase, indicando los atributos y sus tipos de datos.

Ahora vamos a hacer clic en el menú principal, en la opción “Fuente”, luego “Insertar código” y luego “getters”, haz clic en la clase Cliente para seleccionar todos los atributos, el resultado se aprecia en la figura 18.

Figura 18.

Los getters han sido incorporados.



Nota: Es el resultado de hacer clic en el menú principal, en la opción “Fuente”, luego “Insertar código” y luego “getters”, haz clic en la clase Cliente para seleccionar todos los atributos.

Ahora la clase toma forma con los getters como se aprecia en la figura 19.

Figura 19:

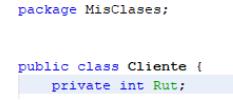
Netbeans ha agregado los getters.

Nota: La clase toma forma con los getters.

Ahora realiza la misma operación anterior, pero esta vez escoge “Setters”, para luego terminar la clase, tal como se aprecia en la figura 20.

Figura 20:

Clase terminada con sus respectivos getters y Setters.



The screenshot shows the Eclipse IDE interface with the Source view selected. The code editor displays the following Java code:

```
1 package MisClases;
2
3
4
5 public class Cliente {
6     private int Rut;
7
8     public void setRut(int Rut) {
9         this.Rut = Rut;
10    }
11
12     public void setNombre(String nombre) {
13         this.nombre = nombre;
14    }
15
16     public void setCorreo(String correo) {
17         this.correo = correo;
```

Nota: Se realiza la misma operación anterior, pero esta vez se escoge “setters”, para luego terminar la clase.

Finalmente recuerda guardar tu proyecto, ahora puedes continuar con el desarrollo de tu aplicación.

Implementando el sistema de ventas a través de la vista lógica

A continuación, realizarán una actividad práctica en la cual, deberán completar el resto de las clases que sus requisitos funcionales requieran, recuerda hacer este caso con tu grupo, así podrán entender bien como colaborativamente un sistema es construido.

Pueden revisar en profundidad el ejercicio, a través de la visualización de estos 2 videos que se podrán encontrar en el material de la semana 4, Creando Clases con starUML y Creando Clases con Java y Netbeans.

Video



En el video “Creando clases con starUML” podrás revisar en profundidad el ejercicio

https://videosduoc.duoc.cl/media/t/1_jvshiokf

Podrás también hacerlo en el video “Creando clases con Java y Netbeans”:

https://videosduoc.duoc.cl/media/t/1_gd1g2g9l

Una vez creado el resto de las clases faltantes, genere los diagramas de comunicación y secuencia, usando para ello lo aprendido en la semana 2, y visualizando el video “Generación de la vista lógica, diagrama de comunicación” y “Generación de la vista lógica, diagrama de secuencia”.

Video



Para generar los diagramas de clases, puedes volver a ver el siguiente video “Vista Lógica – Diagrama de clases”: https://videosduoc.duoc.cl/media/t/1_fimgq6hl

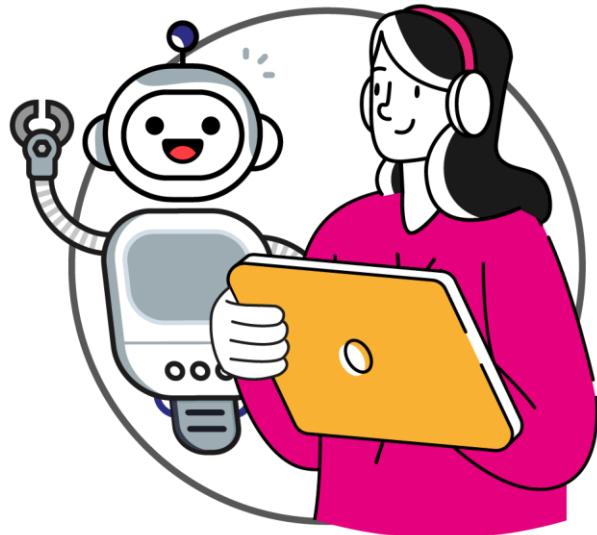
Para generar los diagramas de comunicación, puedes volver a ver el siguiente video “Vista Lógica – Diagrama de Comunicaciones”: https://videosduoc.duoc.cl/media/t/1_shmfifyc

Para generar los diagramas de secuencia, puedes volver a ver el siguiente video “Vista Lógica – Diagrama de secuencia”: https://videosduoc.duoc.cl/media/t/1_u11wmj0i

Cierre de la semana

Durante esta cuarta semana, has aprendido a documentar la vista lógica, en especial los diagramas de clases los cuales nos muestras los componentes de nuestro sistema, clases, datos, métodos, comunicaciones y relaciones, se realizó un paso a paso de como generar clases a través del software starUML y con Netbeans en un proyecto Java.

Durante las siguientes semanas, ganarán un mayor conocimiento en la arquitectura de software, profundizando en temas como la implementación de la vista de comportamiento, diagrama de entidad relación, y también podrás construir el primer diagrama del modelo de datos del sistema, que tengan una excelente semana.



Referencias

Gana J. (2023). Paso a paso de cómo crear una clase usando starUML.

Gana J. (2023). Paso a paso de cómo crear una clase usando Java y Netbeans.

Gana J. (2023). Paso a paso de cómo crear clases.

Gana J. (2023). Paso a paso de cómo crear Diagrama de secuencia.

Gana J. (2023). Paso a paso de cómo crear el diagrama de comunicación.

Apunte



Duoc UC

Facilitador disciplinar: Juan Alberto Gana.

Asesor par: Domingo Sanz.

Reservados todos los derechos Fundación Instituto Profesional Duoc UC. No se permite copiar, reproducir, reeditar, descargar, publicar, emitir, difundir, de forma total o parcial la presente obra, ni su incorporación a un sistema informático, ni su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia,

grabación u otros) sin autorización previa y por escrito de Fundación Instituto Profesional Duoc UC La infracción de dichos derechos puede constituir un delito contra la propiedad intelectual.



Carrera
**Analista Programador
Computacional**

Exp 2 – Semana 5

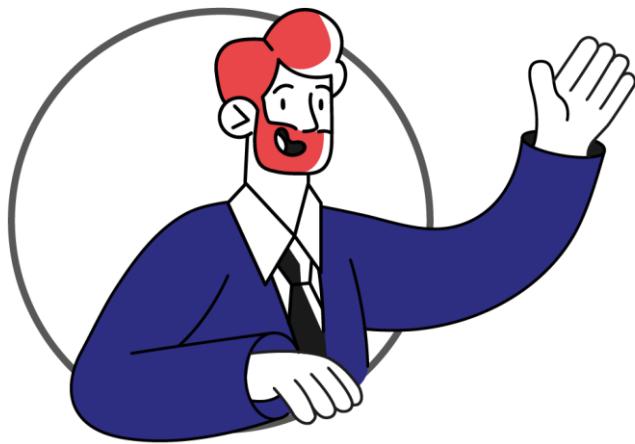
Arquitectura ASY4231

Guía de Aprendizaje

Índice

Introducción a la semana.....	2
Resultado de aprendizaje	3
El estudiante será capaz de:	3
Indicador de logro:.....	3
Conceptos relevantes	3
Preguntas activadoras.....	4
Actividad	4
La vista de comportamiento y procesos	5
Diagrama de Entidad relación	5
Modelando los procesos y las actividades.....	12
Diagrama de actividades	12
Diagrama de procesos de negocio	14
Diagramando las actividades	18
Cierre de la semana	20
Referencias	21
Apunte	22

Introducción a la semana



En esta quinta semana pondremos en práctica lo aprendido sobre cómo documentar la arquitectura de nuestro sistema a través de los distintos diagramas de vistas 4+1, en esta ocasión trabajaremos implementando el modelo de entidad relación y los diagramas de proceso empresarial.

La gran mayoría de los sistemas utiliza un mecanismo de persistencia para almacenar

los datos, estos datos habitualmente son almacenados en los llamados RDBMS o Relational Database Management System o Sistema de gestión de administración de bases de datos, en nuestro caso utilizaremos Oracle.

Como estándar en la gestión de modelado de procesos, se deben crear los diagramas de actividad (parte de la vista de procesos), que nos muestra las acciones del sistema, para su modelado, usaremos la herramienta starUML y para generar el diagrama de proceso usaremos Bizagi modeler.

Durante esta semana también tendrás que revisar el video “Generando el diagrama de Entidad Relación”, el cual explicará cómo crear el modelo de entidad relación, que tengan una excelente semana de aprendizaje.

Resultado de aprendizaje

El estudiante será capaz de:

RA2. Implementa el modelo arquitectónico en el lenguaje de programación seleccionado para soportar la solución sistémica requerida por el cliente.

Indicador de logro:

IL6. Implementa la vista de comportamiento, generando el modelo de entidad relación y los elementos de persistencia.

IL7. Implementa la vista de procesos y diagramas de actividades propuesto en la descripción de arquitectura.

Conceptos relevantes

	4+1	Objeto	Diagrama
	Comunicación	Secuencia	Clase
	Atributo	Operaciones	Relaciones

Preguntas activadoras

- ¿Qué son objetos, y lenguaje orientado a objetos?
- ¿Qué es el diagrama de comunicación y secuencia?
- ¿Qué son las clases?
- ¿Qué son los atributos?
- ¿Qué son las operaciones?

Actividad

En esta quinta semana la actividad será formativa y se llama "Implementando la arquitectura - vista de comportamiento y procesos". Donde deberán implementar la vista de comportamiento generando los componentes de base de datos y persistencia necesarios para soportar el funcionamiento del sistema. También realizarán el modelado de procesos de negocios en base a diagramas de BPMn.

La vista de comportamiento y procesos

Diagrama de Entidad relación

Para iniciar recordemos que un motor de base de datos, en nuestro caso Oracle, administra diversas bases de datos, las cuales a su vez gestionan objetos como tablas, vistas, procedimientos almacenados, usuarios, entre otros muchos objetos.

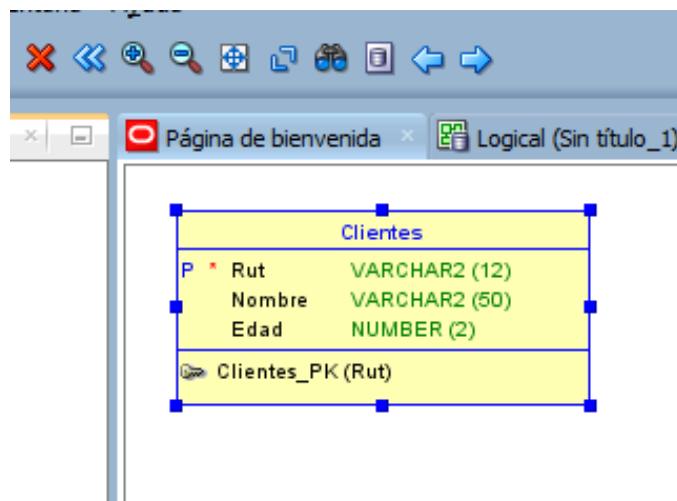
Nos enfocaremos en los objetos que son necesarios para que el sistema pueda almacenar nuestros datos, estos son tablas en general.

Durante el levantamiento de requerimientos de usuario, se inicia el proceso de modelado, para ello se implementan las entidades, las que provienen de los objetos declarados en el diagrama de clases, estas contienen columnas (campos), las cuales tienen atributos y sus respectivas configuraciones.

Por ejemplo como se vé en la figura 1, se muestra una tabla de clientes con sus respectivos atributos:

Figura 1

Tabla de clientes.



Nota: Tabla de clientes con sus respectivos atributos.

Para crear las distintas tablas del sistema podemos utilizar en el caso de Oracle, las herramientas Oracle Datamodeler o bien SQL Developer.

Para crear el modelo de datos, deberán tomar el proyecto que estan desarrollando como caso base y usando el diagrama de clases, crear todas las tablas que sean necesarias, indicando las llaves primarias (PK) y las llaves foraneas (FK).

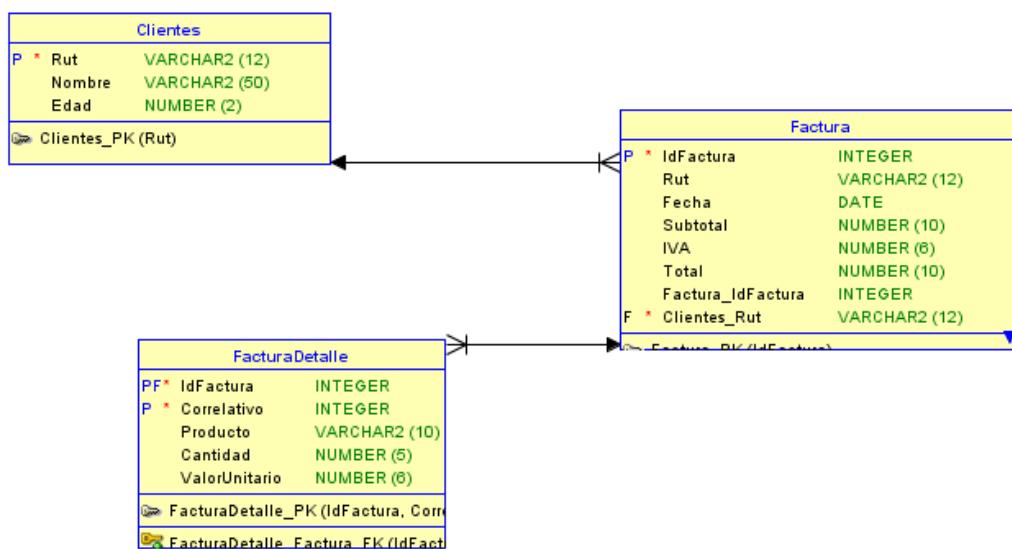
Importante



Recordar que es muy importante indicar las relaciones que existen entre tablas, como se aprecia en la figura 2, un Cliente pueden tener muchas facturas, una factura puede contener muchas líneas de detalle.

Figura 2:

Modelo ejemplo, 3 tablas y sus respectivas cardinalidades.



Nota: Se presentan tres tablas (Clientes Factura y FacturaDetalle) y bajo cada una de ellas sus respectivas cardinalidades.

A continuación, sigue los siguientes pasos para la correcta creación de tus tablas del Sistema de Ventas.

PASO 1: Crear el modelo entidad relación, perteneciente a la vista Lógica.

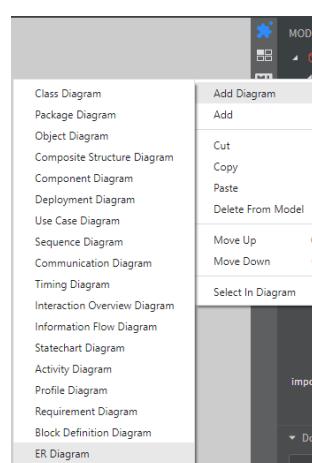
Para crear el modelo entidad relación, usaremos como primer paso, el software starUML para crear el diagrama de entidad relación. Recordar que estamos usando el sistema de ventas, el cual fue provisto como material de este curso.

En el requerimiento RF001, se nos pide que el cliente pueda registrarse, para ello hemos ya creado una clase llamada “Cliente”, la cual fue implementada en starUML y que nos generó código Java para su implementación.

En starUML, cree un nuevo diagrama de entidad relación, como se aprecia en la figura 3.

Figura 3:

Creación de un nuevo diagrama de entidad relación.



Nota: Se crea un nuevo diagrama de entidad-relación al presionar Class Diagram y seleccionar Add Diagram.

Esto creará un nuevo diagrama ER o entidad relación. Posteriormente indique todas las columnas y sus respectivos tipos de datos y largos, tal como se aprecia en la figura 4.

Figura 4:

Diagrama E-R con la entidad Cliente.

Cliente			
PK	Rut	INTEGER(10)	
	Nombre	VARCHAR(50)	
	Correo	VARCHAR(100)	
	Contraseña	VARCHAR(30)	

Nota: Diagrama E-R con la entidad Cliente, sus columnas, tipos de datos y largos.

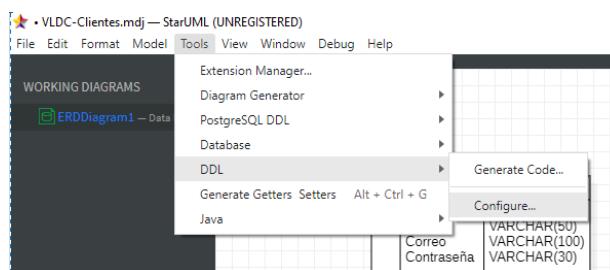
Una vez creado todo el modelo (todas las clases que se convierten en entidades de datos), entidad relación creado con el software starUML, podemos crear el script que puede crear las tablas del sistema.

PASO 2: Configurando starUML para la creación del script de base de datos.

Haga clic en el menú principal, en la opción “Tools”, luego en la opción “DDL”, posteriormente clic en “Configure...”, tal como se aprecia en la figura 5, esto nos llevará a la siguiente pantalla en la cual realizaremos los ajustes necesarios para crear el script de base de datos.

Figura 5:

Configurar starUML para crear el script de base de datos.



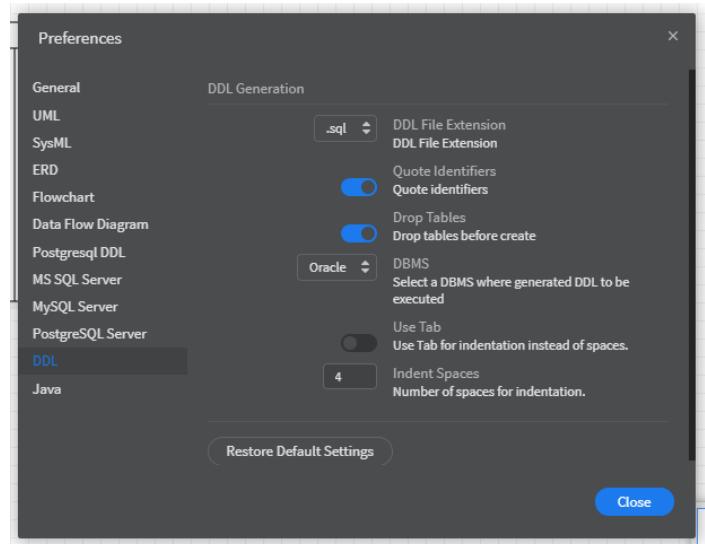
Nota: Hacer clic en el menú principal, en la opción “Tools”, luego en la opción “DDL” para seleccionar “Configure”.

Una vez en la pantalla de configuración revise o cambie las siguientes funcionalidades.

Verificar que este activada en azul la opción “Drop Tables”, y que el campo DBMS indica “Oracle”, presionar “Close”, como se muestra en la figura 6.

Figura 6:

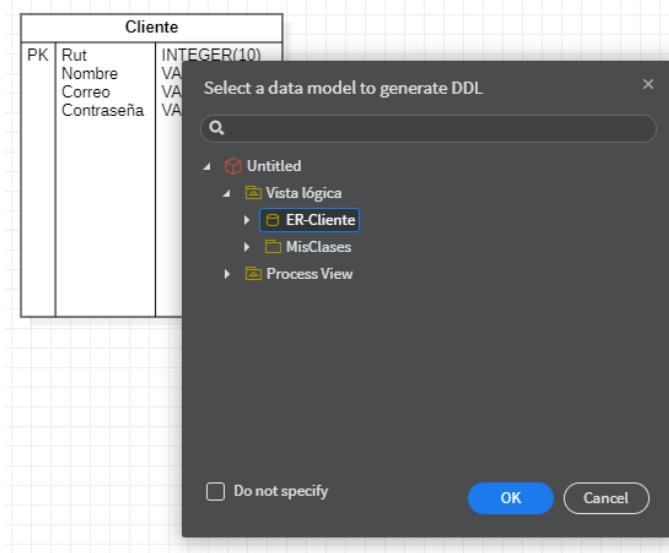
Configuración de starUML



Nota: En la pantalla de configuración deben aparecer o cambiar a las siguientes funcionalidades: “Drop Tables” en azul, y que el campo DBMS debe indicar “Oracle”. Si la funcionalidad es correcta presionar “Close”.

PASO 3: Crear el script de base de datos.

Luego desde el menú principal, selecciona “Tools”, luego “DDL”, finalmente “Generate Code...”, esto abrirá una nueva ventana en la cual debemos seleccionar el modelo de datos creado, en nuestro ejemplo, seleccionaremos el modelo llamado ER-Cliente, tal como se aprecia en la figura 7 y presionamos “OK”, esto hará que starUML nos pregunta donde dejaremos el script, indica la carpeta en la cual quedará almacenado el script de la tabla, en este caso usare la carpeta “Sistema”, de mi estructura del sistema, proporciona un nombre al archivo y aceptar.

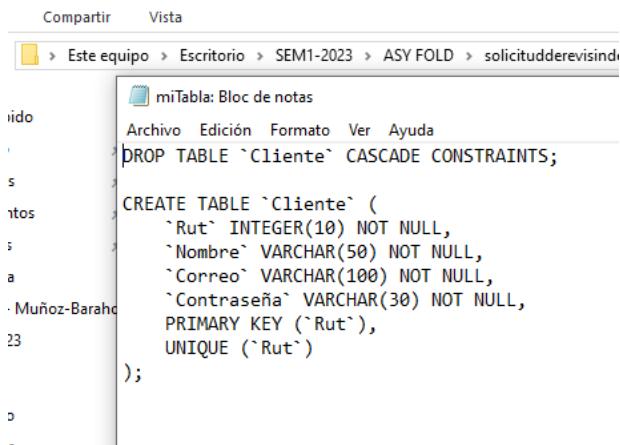
Figura 7:*Creación del script de base de datos*

Nota: Después de presionar “DDL” y seleccionar “Generate Code...”, se abrirá una nueva ventana en la cual debemos seleccionar el modelo de datos creado, en nuestro ejemplo, seleccionaremos el modelo llamado ER-Cliente, y presionamos “OK”.

Para revisar el script, usa un editor de texto, edita el archivo como se ve en la figura 8. Ahora puedes copiar el texto y pegarlo en una sesión de SQL Developer para crear la tabla, el Script primero elimina la tabla en caso de existir y luego procede a crearla en el sistema de base de datos.

Figura 8:

Script de la tabla cliente creado.



The screenshot shows a Microsoft Notepad window titled "miTabla: Bloc de notas". The window contains a MySQL script for creating a table named "Cliente". The script includes a DROP TABLE statement followed by a CREATE TABLE statement. The CREATE TABLE statement defines five columns: "Rut" (INTEGER(10) NOT NULL), "Nombre" (VARCHAR(50) NOT NULL), "Correo" (VARCHAR(100) NOT NULL), "Contraseña" (VARCHAR(30) NOT NULL), and a primary key constraint on "Rut". There is also a unique constraint on "Rut". The file path in the title bar is "Este equipo > Escritorio > SEM1-2023 > ASY FOLD > solicitudderevisi&on.sql".

```
DROP TABLE `Cliente` CASCADE CONSTRAINTS;
CREATE TABLE `Cliente` (
    `Rut` INTEGER(10) NOT NULL,
    `Nombre` VARCHAR(50) NOT NULL,
    `Correo` VARCHAR(100) NOT NULL,
    `Contraseña` VARCHAR(30) NOT NULL,
    PRIMARY KEY (`Rut`),
    UNIQUE (`Rut`)
);
```

Nota: El script de la tabla ya contiene un cliente creado.

Modelando los procesos y las actividades

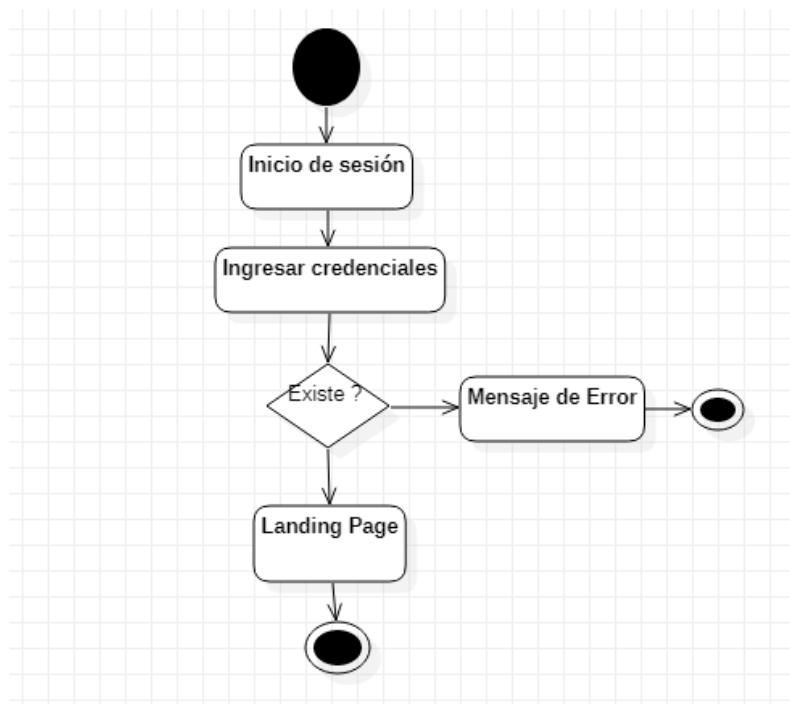
A partir de la vista de proceso, crearemos como siguiente paso, el diagrama de actividades, para ello debemos tener muy claramente cuáles son las actividades que los procesos en estudio indican.

Diagrama de actividades

Una actividad es una acción que un rol responsable ejecuta, como por ejemplo si el proceso corresponde a un inicio de sesión para un usuario, la figura 5 nos muestra su diagrama de actividad.

Figura 5:

Diagrama de actividades del inicio de sesión.

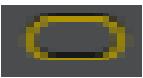


Nota: Diagrama de inicio de sesión, luego Ingresar credenciales llega a la pregunta ¿Existe? Con dos opciones “Mensaje error” o “Landing page”.

Para desarrollar un diagrama de actividades debes conocer los siguientes conceptos:

Tabla 1:

Diagramas de actividades

Actividad: Son las acciones que realiza el sistema.	
Flujo: Indica el flujo que llevan las actividades descritas.	
Inicio: Indica el inicio de las actividades.	
Término: Indica el punto de término de las actividades.	
Nodo de Bifurcación: Permite a partir de una actividad generar 2 ó más tareas en paralelo.	
Nodo de Unión: Permite centralizar varias actividades y concentrarlas en una nueva actividad.	
Decisión: Permite tomar una decisión, la cual puede tomar 2 caminos posibles.	

Nota: Se muestran los diagramas de actividades y su simbología.

Video



Para la creación del diagrama de actividad, refiérase al video “Vista de Proceso - Diagrama de actividad”: https://videosduoc.duoc.cl/media/t1_v979wnn1

Diagrama de procesos de negocio

El diagrama de procesos de negocios nos permite conocer cómo funciona la empresa en términos de procesos.

Los procesos recordemos es un grupo de actividades que se ejecutan en forma sistemática, repetitivamente, tienen un dueño del proceso, tienen entradas y salidas específicas.

Los procesos pueden ser divididos en:

- **Proceso de negocio** (principal): es aquel que representa las actividades principales que dan vida al funcionamiento de la empresa del cliente, ejemplo de ellos son Ventas, Producción, abastecimiento, etc.
- **Proceso de soporte** (secundario): Estos procesos, si bien no generan los ingresos de la empresa, dan soporte a los procesos de negocio, ejemplo de estos son TI, Contabilidad, mantenimiento, etc.

Esta gestión de procesos, en la actualidad da origen al modelado BPMN (Business Process Management notation), algunas de las herramientas para diagramarlos son starUML, Visio y Bizagi Modeler. Durante esta semana se adjunta un manual de buenas prácticas para el modelado de procesos de negocio usando Bizagi Modeler.

Creando un modelo de procesos usando Bizagi Modeler

Como primer paso, debemos entender como funciona la empresa en términos de proceso, para ello te puedes apoyar en un analista de procesos, o bien algún usuario clase en la organización.

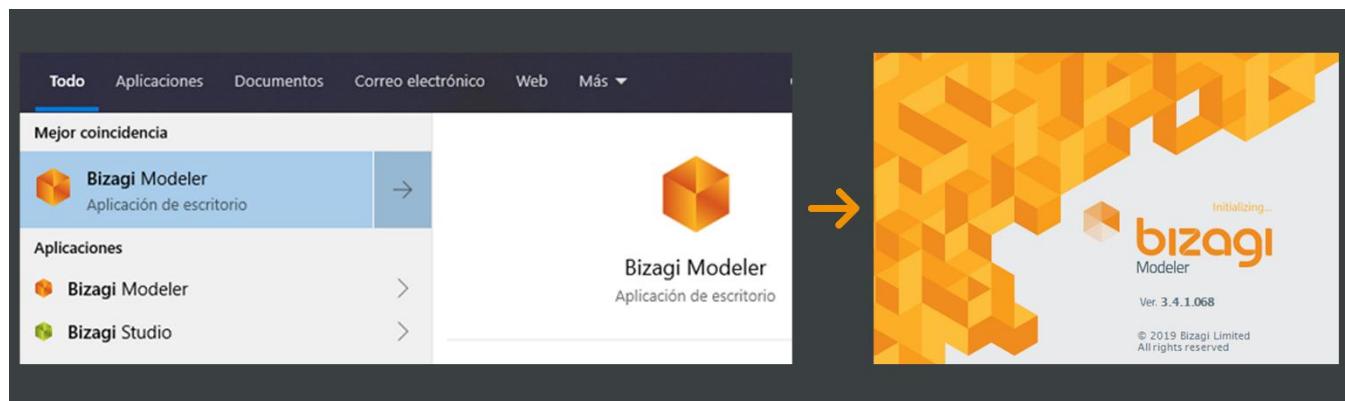
Para diagramar el proceso, primero regístrate en [Bizagi.com](https://www.bizagi.com) como usuario, lo cual es obligatorio para usar Bizagi Modeler en su versión gratuita.

Instala bizagi en tu computador, ábrelo y comencemos a trabajar, el siguiente ejemplo es como debiese ser el proceso principal de ventas propuesto para el desarrollo del sistema de ventas.

PASO 1: Abra Bizagi, desde la línea de comandos o bien desde los programas en Windows, como se ve en la figura 10.

Figura 10.

Abriendo *Bizagi Modeler*

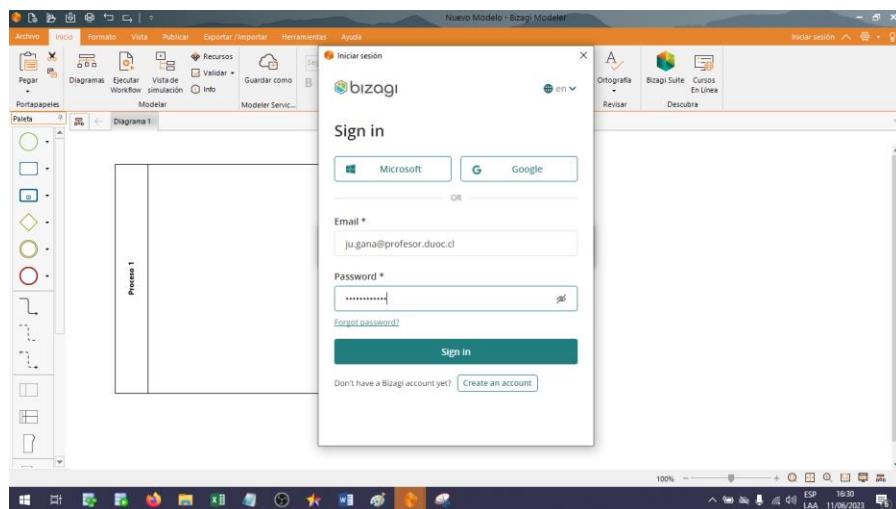


Nota: Desde Windows se puede encontrar la Aplicación “Bizagi Modeler” y seleccionarla.

PASO 2: Ingrese a Bizagi, pedirá su autenticación como usuario, si no posee cuenta, puede crearla desde este mismo lugar, en mi caso usaré mi cuenta existente, luego de eso indicar iniciar sesión.

Figura 11:

Iniciar sesión en Bizagi.



Nota: Al iniciar sesión en Bizagi, solicitará autenticación, puedes elegir entre una cuenta de Microsoft o Google. También tienes la opción de crearla.

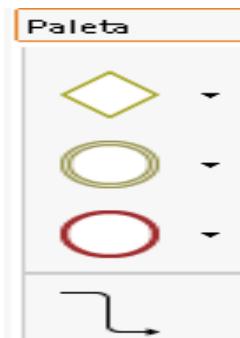
Una vez en Bizagi, configure su proceso para esto por defecto aparece un proceso en blanco, otórgale un nombre al proceso, dando doble clic al nombre del proceso, luego cambia su nombre por Ventas.

PASO 3: Luego debes evaluar cuantos roles participaran del proceso, ejemplo un cliente, un vendedor, un sistema, un proveedor, etc. En nuestro caso usaremos los roles del cliente y vendedor, asumiendo que el proceso tiene una colaboración del vendedor para vender un producto al cliente, en este caso, usaremos 2 Lanes o pistas, las cuales representan al cliente y la otra al vendedor, tal como se muestra en la figura 12.

Figura 12:*Roles*

Nota: El proceso queda definido con 2 roles: Vendedor y Cliente.

Procede a diagramar las actividades del cliente y del vendedor usando las herramientas provistas por Bizagi en su paleta de herramientas al lado izquierdo, algunas de ellas se aprecian en la figura 13.

Figura 13:*Herramientas de diagramado Bizagi.*

Nota: Algunas herramientas de diagramado Bizagi "Paleta"

Diagramando las actividades

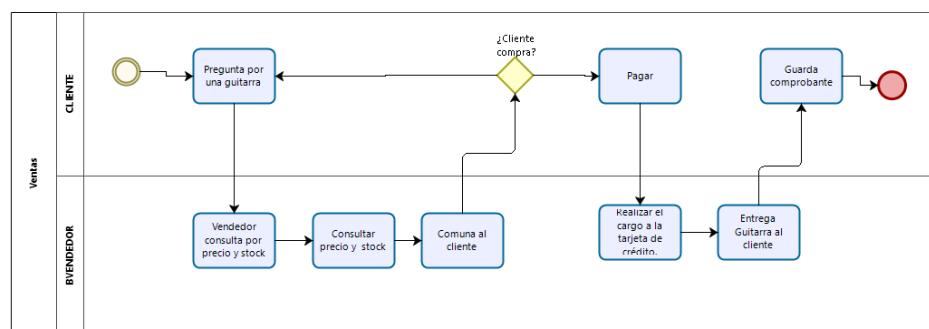
Para realizar el diagrama, veremos que nuestro proceso puede describirse de la siguiente manera, un cliente inicia el proceso cuando entra a la tienda, el vendedor lo saluda, el cliente pregunta por una guitarra, el vendedor consulta precio y stock, entregándole la información al cliente, quien decide comprarla, el vendedor le pide la tarjeta de crédito y procede a realizar el cargo, posterior a ello, el cliente recibe su guitarra y se retira de la tienda.

El breve análisis anterior es obtenido de analizar el proceso de negocio, ya sea provisto por el cliente, observado en terreno o bien desde el analista de proceso.

Con esta información realizaremos el diagrama del proceso de ventas, como se aprecia en la figura 14.

Figura 14:

Proceso diagramado



Nota: Diagrama de proceso de venta.

Se ha completado el diseño del proceso de ventas.

Importante

Recordar que antes de implementar cualquier solución tecnológica, la empresa debiese analizar sus procesos de negocio para generar mejoras, así la solución, estará basada en procesos mejorados.

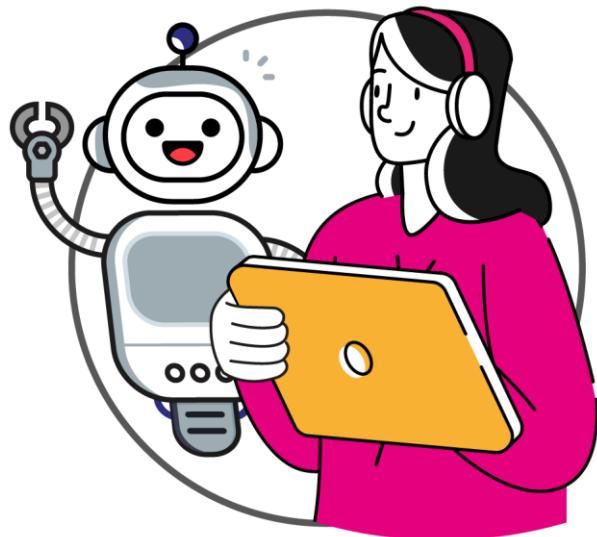


Para continuar con nuestro proceso de aprendizaje, en su grupo de trabajo, desarrolle los diagramas de actividad y procesos que se pueden apreciar en el proyecto base que han estado trabajando, luego el docente dará una retroalimentación formativa

Cierre de la semana

Durante esta quinta semana, has aprendido a documentar la vista lógica, en especial el diagrama de entidad relación (MER) y los diagramas de actividad y proceso pertenecientes a la vista de proceso, por otro lado, aprendiste a utilizar las herramientas starUML para generar los diagramas y luego generar los scripts de creación de las tablas y sus relaciones, recuerda utilizar la herramienta SQL Developer, el cual permite la creación de tablas y los distintos objetos de la base de datos.

Durante las siguientes semanas, tendrán la misión de validar la arquitectura creada, a través de la metodología ATAM, que tengan una excelente semana.



Referencias

Gana, J (2019). Manual de buenas prácticas modelando procesos con Bizagi



Apunte



Duoc UC[®]

ONLINE

Duoc UC

Facilitador disciplinar: Juan Alberto Gana.

Asesor par: Domingo Sanz.

Reservados todos los derechos Fundación Instituto Profesional Duoc UC. No se permite copiar, reproducir, reeditar, descargar, publicar, emitir, difundir, de forma total o parcial la presente obra, ni su incorporación a un sistema informático, ni su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia,

grabación u otros) sin autorización previa y por escrito de Fundación Instituto Profesional Duoc UC La infracción de dichos derechos puede constituir un delito contra la propiedad intelectual.



Carrera **Analista Programador Computacional**

Exp 2 – Semana 6

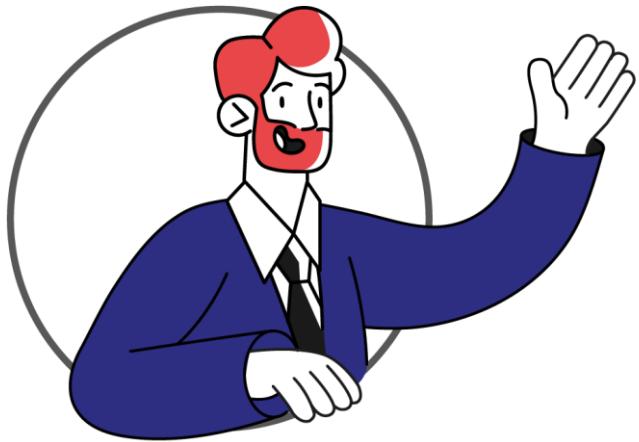
Arquitectura – ASY4231

Guía de aprendizaje

Índice

Introducción a la semana.....	9
Resultado de aprendizaje	10
El estudiante será capaz de:	10
Indicador de logro:.....	10
Conceptos relevantes	10
Preguntas activadoras.....	11
Actividad	11
DAS (Documento de arquitectura de Sistemas)	12
Elaboración del DAS	14
Introducción	14
Metas y restricciones de la arquitectura.	18
Vista de casos de uso y escenarios de calidad	19
Vista lógica	20
Vista de procesos	23
Vista de desarrollo.....	24
Vista física	26
Vista de escenarios	27
Decisiones de Diseño y Selección de alternativas	28
Análisis de Reutilización.....	29
Cierre de la semana	30
Referencias	31
Lecturas de la semana	31
Apunte	32

Introducción a la semana



entregado al cliente para su entendimiento.

Posterior al análisis del DAS, tendrás la misión junto a tu grupo de terminar el DAS incorporando todos los elementos que han venido desarrollando en las semanas 4 y 5, que tengan una excelente semana de aprendizaje.

En esta sexta semana seguiremos aplicando el conocimiento generado, esta vez debemos formalizar los distintos aspectos de la arquitectura, lo cual haremos con un nuevo documento llamado DAS o bien Documento de Arquitectura de sistemas.

El DAS nos permite ordenar y conceptualizar los aspectos de la arquitectura, generando un documento fácil de usar y que puede ser

Resultado de aprendizaje

El estudiante será capaz de:

RA2. Implementa el modelo arquitectónico en el lenguaje de programación seleccionado para soportar la solución sistémica requerida por el cliente.

Indicador de logro:

IL5. Implementa la vista lógica del proyecto, en base a los requerimientos funcionales del proyecto.

IL6. Implementa la vista de comportamiento, generando el modelo de entidad relación y los elementos de persistencia.

IL7. Implementa la vista de procesos y diagramas de actividades propuesto en la descripción de arquitectura.

Conceptos relevantes

	4+1	Objeto	Diagrama
	Comunicación	Secuencia	Clase
	Atributo	Operaciones	Relaciones

Preguntas activadoras

- ¿Qué son objetos, y lenguaje orientado a objetos?
- ¿Qué es el diagrama de comunicación y secuencia?
- ¿Qué son las clases?
- ¿Qué son los atributos?
- ¿Qué son las operaciones?

Actividad

En esta sexta semana realizarán la actividad sumativa de la Experiencia 2 llamada "Implementación de la arquitectura", esta actividad está dividida en 3 partes.

Para la primera parte a través de un encargo grupal, deberán entregar la documentación realizada con los aportes solicitados. La documentación requerida será el D.A.S, diagrama de clases, comunicación y secuencia, diagrama de entidad relación, diagrama de BPMN de los procesos de la empresa. Para la Parte II deberán hacer una presentación en resumiendo el trabajo realizado.

Para la Parte III, deberán generar un espacio de reflexión sobre los aprendizajes adquiridos de manera individual.

DAS (Documento de arquitectura de Sistemas)

El DAS o Documento de arquitectura de software es un documento técnico que le permite al arquitecto documentar el sistema en términos de su arquitectura, esto es, metas, restricciones, supuestos, atributos de calidad, los diferentes puntos de vistas, los cuales se documentan con las vistas 4+1 y las decisiones arquitectónicas.

Recordemos que la arquitectura del software debe ser compartida con miembros del equipo desarrollador, usuarios y stakeholders en general, desde aquí se desprende que las vistas 4+1 están orientadas a distinto público.

Para refrescar el conocimiento, la Vista de escenarios corresponde en su mayoría a los casos de uso, los cuales están orientados a los usuarios finales, la vista de proceso está mayoritariamente dirigida a los integradores de sistemas, la vista de desarrollo a los programadores y desarrolladores, la vista lógica a los usuarios y finalmente la vista física a los ingenieros de sistemas.

Por tal motivo el DAS debe ser especificado en un lenguaje más bien neutro que pueda ser entendido por todas las personas involucradas en el desarrollo del software.

El DAS no tiene una estructura siempre igual a otros proyectos, ya que depende del contexto de nuestro sistema, recordemos que la arquitectura no solo se usa para documentar el desarrollo de un software, también puede ser utilizado para documentar la creación de servicios WEB por ejemplo, o bien la implementación de algún sistema de clase mundial, el cual será ejecutado en una infraestructura específica.

A continuación, explicaremos cómo llenar este documento, basándonos en el sistema de ventas que hemos venido construyendo a través de las semanas anteriores.

Para resumir, nuestro cliente nos ha solicitado la construcción de un sistema que permita asistirlo en el proceso de ventas de sus clientes, además de herramientas de gestión de productos. Del mismo modo se espera que los clientes puedan ingresar productos al carrito de compras, y los administradores puedan visualizar los pedidos a través del módulo de gestión de pedidos.

En términos de atributos de calidad, el cliente nos ha mencionado que desea tener un sistema seguro, por lo que los usuarios deberán autenticarse con usuario y clave, respecto del rendimiento, el cliente nos ha indicado que el sistema debe manejar una gran carga de transacciones, por lo que el sistema debe tener un buen control de las sesiones concurrentes. Este sistema es muy importante para el cliente, luego le preocupa mucho la disponibilidad del mismo, es por ello que hace énfasis en que el sistema debe estar disponible para cuando sus clientes lo necesiten, es decir su disponibilidad no debe ser inferior a un 99,7% del tiempo.

Finalmente el cliente desea que el sistema sea intuitivo y fácil de usar por sus clientes y usuarios, ya que esto puede permitir que los usuarios finales en general aumenten su uso y no requieran sesiones de entrenamiento en el uso del sistema.

Durante el transcurso de tiempo en el llenado de este documento DAS, iremos dando indicaciones de qué información colocar en cada ítem, para luego un ejemplo, el cual saldrá de la información del proyecto que hemos venido entregando semana a semana.

Elaboración del DAS

Introducción

Contexto del problema: Indicar brevemente cual es la problemática o necesidad del cliente, esta información puede ser recuperada del E.R.S.

Ejemplo:

El cliente nos ha encargado el desarrollo de un sistema que permita ayudar a gestionar la venta de productos para sus clientes, a través de una plataforma que le permita en forma fácil poder seleccionar productos, llevarlos a un carro de compras, pagarlos y programar su despacho.

Del mismo modo requiere de una plataforma para sus administradores, lo que les permitirá gestionar los pedidos de los clientes.

Propósito: Indicar en breves palabras que se quiere conseguir con la solución y para qué se está creando el sistema, en general debe explicar como el sistema en construcción aportará valor agregado a la gestión del cliente.

Ejemplo:

El sistema permitirá un aumento de las ventas en un 10% diario, dado que los clientes, no tendrán que viajar hasta las distintas sucursales para realizar sus pedidos, ya que podrán hacerlo a través de internet, lo que les permitirá ahorrar en el desplazamiento para sus compras.

También optimizará la gestión del inventario, ya que podremos controlar en tiempo real la disminución del inventario dado las compras en línea, también mejorar el desempeño de los administradores, quienes podrán gestionar los pedidos en forma electrónica.

Ámbito: Explicar en pocas palabras el alcance de una decisión de diseño o de una característica arquitectónica en el sistema siendo construido, que parte está siendo afectada.

Ejemplo:

El sistema proveerá de funcionalidades para los clientes, los que podrán revisar productos, seleccionarlos, subirlos al carro de compras, pagar y seleccionar la forma de entrega. Los administradores podrán gestionar los pedidos.

El sistema deberá usar la infraestructura existente y se deben verificar los puntos débiles de la seguridad y rendimiento del mismo. En esta etapa solo se podrá pagar con medios de pago electrónico PayPal.

Definiciones, Acrónimos, Abreviaciones: Indicar y explicar cualquier palabra de orden técnico que utilices, recuerda que los stakeholders o usuarios leerán el documento, por lo que deben conocer la terminología usada.

Ejemplo:

DAS = Documento de arquitectura de Software.

TCP/IP = Protocolo de comunicaciones.

ERS = Especificación de requerimientos de usuario.

CU = Caso de uso.

Referencia: Referencia cualquier material o documento utilizado.

Ejemplo:

Para mayor detalle en los requerimientos del usuario, refiérase al ERS.

Resumen ejecutivo: Explicar en breves palabras, porque la necesidad de construir el sistema incluye solicitudes del cliente, definición del problema y solución proyectada.

Ejemplo:

Para una mejor gestión de la empresa, se ha decidido desarrollar un nuevo software que permita agilizar las compras de los clientes a través de un sistema de ventas WEB, mejorar el desempeño de los administradores de pedidos a través de un gestor de pedidos.

Representación: Hablar sobre cómo se enfoca la arquitectura del sistema, la cual en nuestro ejemplo está basada en el estándar de las vistas 4+1.

Ejemplo:

Para el desarrollo del software, el departamento de tecnología, a cargo del arquitecto, ha decidido utilizar la metodología de vistas 4+1 de Kruschten, la cual se diseñará para que los distintos stakeholders y miembros del equipo, puedan entender las distintas miradas del sistema.

Metas y restricciones de la arquitectura.

Metas de la Arquitectura:

Documentar acá que se requiere alcanzar con la arquitectura, asegurar calidad, rendimiento, usabilidad, etc.

Ejemplo:

Se espera que el sistema en desarrollo pueda lograr las necesidades del cliente en términos de la calidad y características solicitadas, esto es mejorar el desempeño de las transacciones, implementar ciberseguridad para que nuestros clientes puedan compra en un ambiente seguro, que el sistema esté disponible al menos un 99,7% y que posea mecanismos de tolerancia a fallos.

Restricciones de la arquitectura:

Habla de que puntos pueden poner restricciones, como por ejemplo la infraestructura a utilizar, presupuesto, tiempo, etc.

Ejemplo:

Para el desarrollo del sistema, este deberá ajustarse a la actual infraestructura de la empresa, es decir usar la actual base de datos Oracle, los servidores de aplicaciones existentes. Respecto del tiempo, se cuenta con 5 meses para la construcción del sistema y un presupuesto de \$25.000.000.- de pesos.

Vista de casos de uso y escenarios de calidad

Definir los requisitos funcionales, explica los principales casos de uso y escenarios de calidad, para ello puedes incorporar los requisitos documentados en la planilla de requisitos simplificado y los escenarios de calidad creados en semanas anteriores.

Ejemplo:

A continuación, se indican los principales requerimientos funcionales (CU).

- a) RF001: El sistema debe garantizar la seguridad de la información confidencial, como datos personales y detalles de pago, utilizando técnicas de cifrado y almacenamiento seguro.
- b) RF002: Se deben implementar mecanismos de autenticación y autorización para garantizar que solo los usuarios autorizados puedan acceder a ciertas funciones y datos.
- c) RF003: La interfaz de usuario debe ser intuitiva y fácil de usar, tanto para los clientes como para los administradores.

Vista lógica

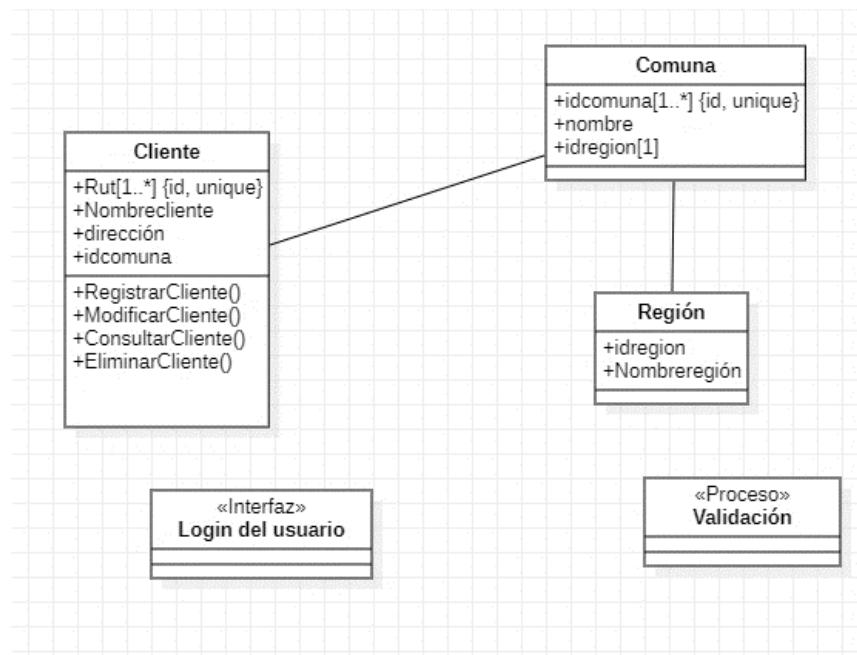
Explicar e insertar acá tus diagramas de la vista lógica, diagrama de clases, comunicación y secuencia.

Ejemplo:

A continuación, se adjuntan los diagramas de la vista lógica, la cual está orientada a los usuarios.

Figura 1:

Diagrama de Clase.



Nota: Diagrama de clases.

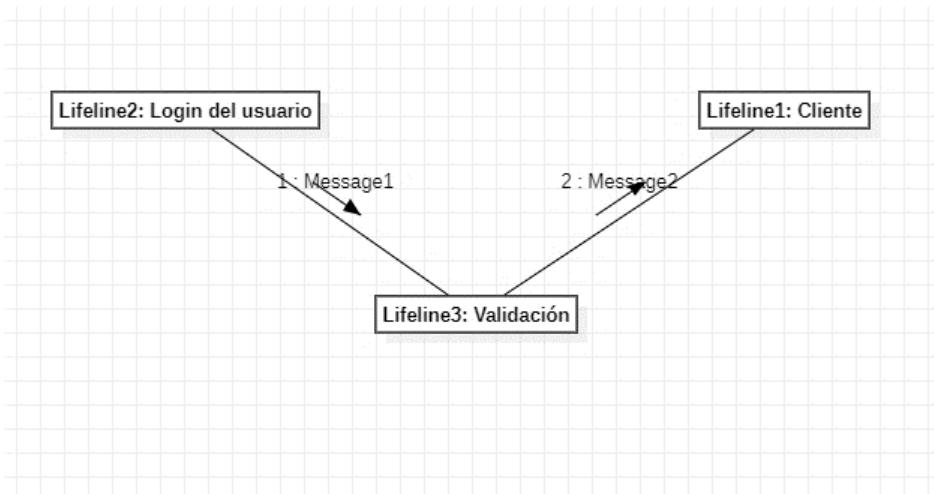
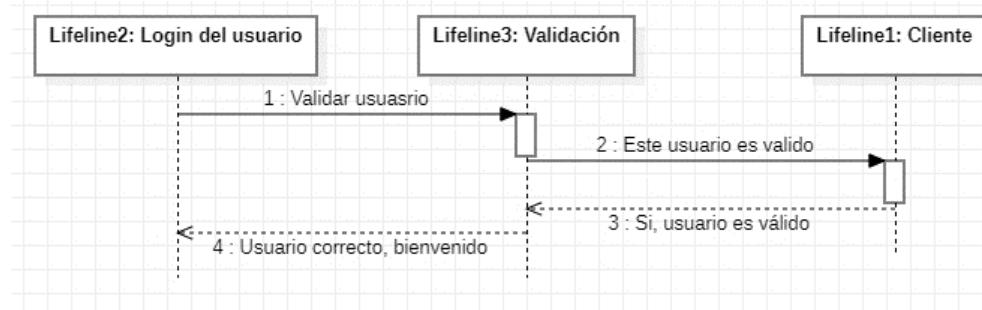
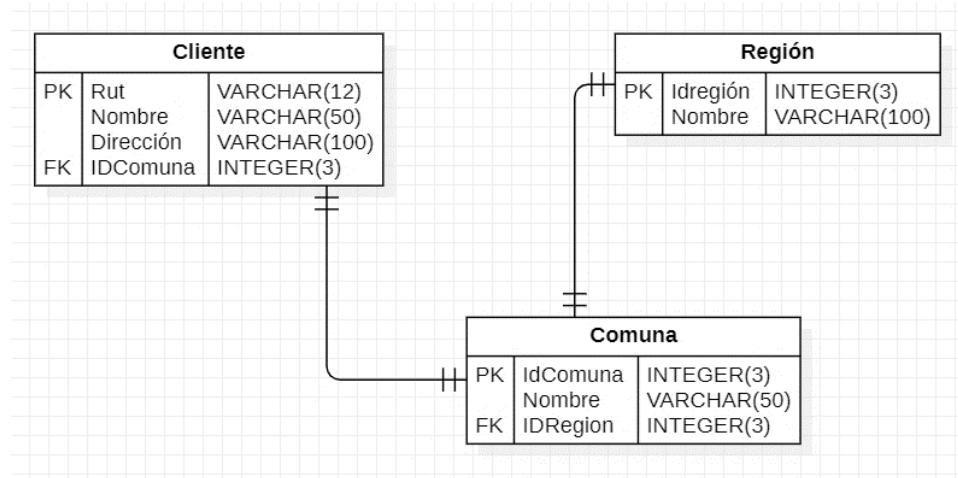
Figura 2:*Diagrama de Comunicaciones.**Nota: Diagrama de comunicaciones.***Figura 3:***Diagrama de secuencia**Nota: Diagrama de secuencia.*

Figura 4:*Diagrama de Entidad relación.**Nota: Diagrama entidad-relación*

Vista de procesos

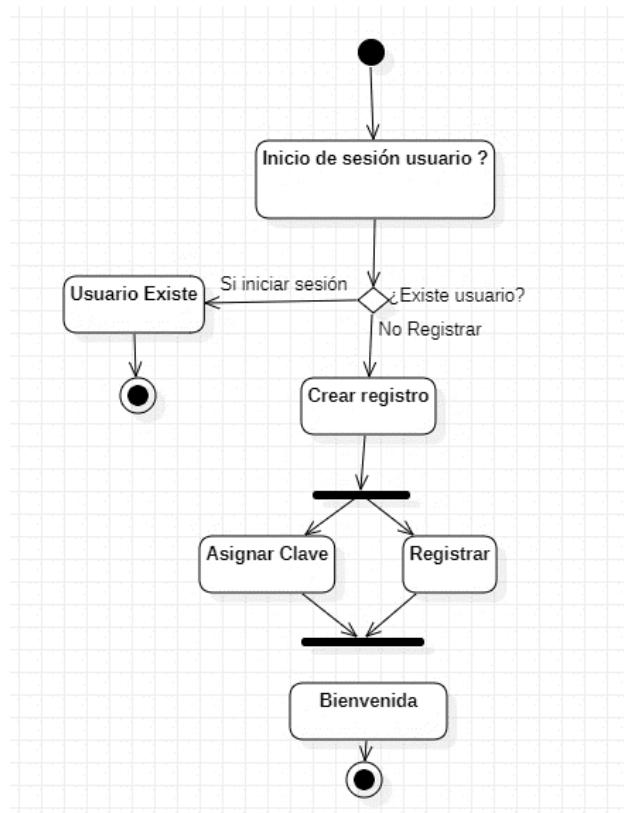
Explica e inserta los diagramas de proceso y actividad.

Ejemplo:

A continuación, se adjuntan los diagramas de la vista de proceso, la cual está orientada a los integradores de sistemas y analistas de procesos.

Figura 5:

Diagrama de Actividades.



Nota: Diagrama de actividades.

Vista de desarrollo

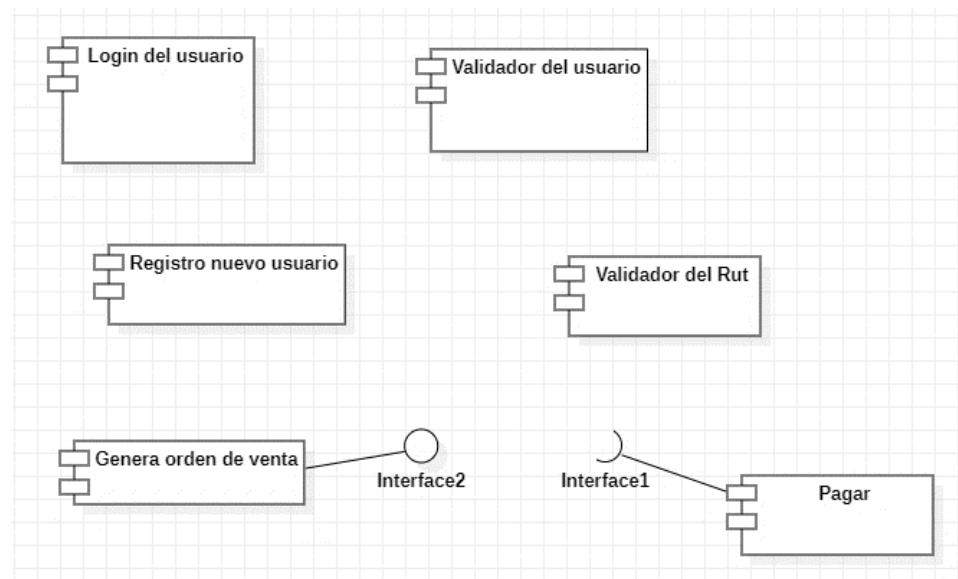
Explica e inserta los diagramas de componentes y paquetes.

Ejemplo:

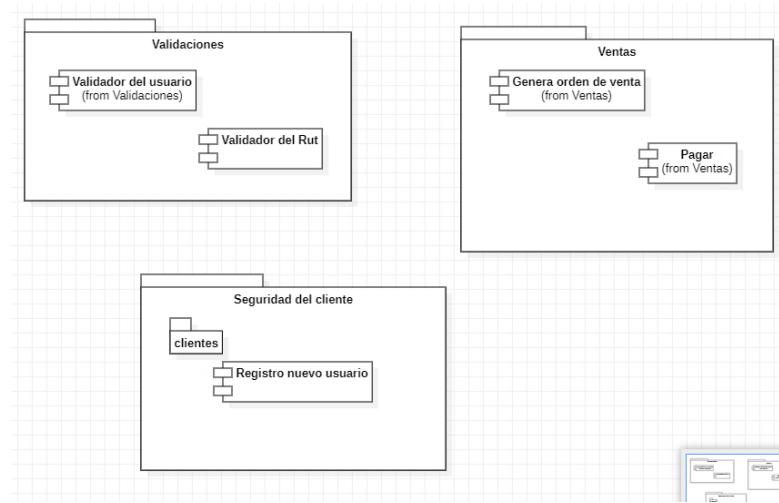
A continuación, se adjuntan los diagramas de la vista de desarrollo, la cual estará orientada a nuestro equipo de desarrollo.

Figura 6:

Diagrama de Componentes.



Nota: Diagrama de componentes

Figura 7:*Diagrama de paquetes.*

Nota: Diagrama de componentes.

Vista física

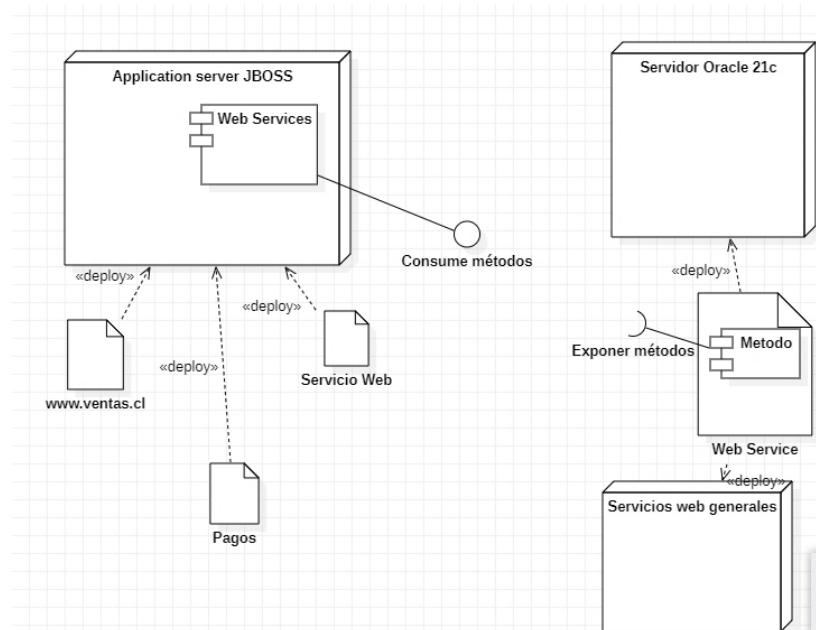
Explica e inserta los diagramas de topología.

Ejemplo:

Se adjuntan los diagramas de la vista Física, la cual estará orientada a nuestro equipo de ingeniería de sistemas.

Figura 8:

Diagrama de Topología.



Nota: Diagrama de topología.

Vista de escenarios

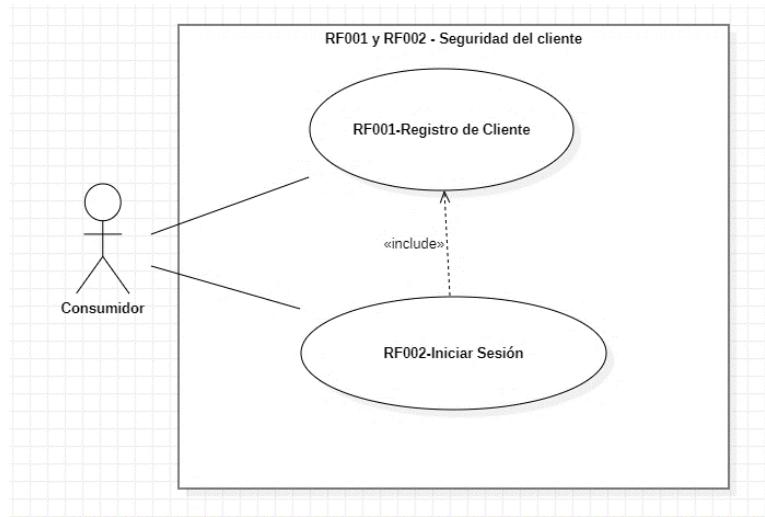
Explica e inserta los diagramas de casos de uso.

Ejemplo:

A continuación, se adjuntan los diagramas de la vista de escenarios, la cual está orientada a nuestros usuarios y stakeholders.

Figura 9:

Diagrama de Casos de Uso.



Nota: Diagrama de casos de uso.

Decisiones de Diseño y Selección de alternativas

Explica acá tus principales decisiones arquitectónicas, como por ejemplo lenguajes a utilizar, bases de datos, patrones, infraestructura, etc.

Ejemplo:

Para el desarrollo del sistema de ventas se han tomado las siguientes decisiones arquitectónicas y alternativas.

- a) Se deberá ampliar el ancho de banda de Internet contratado por la empresa, este no deberá ser inferior a 800Mb, del mismo modo contratar un nuevo enlace con las mismas características, con otro proveedor.
- b) Se deberá incurrir en la compra de un balanceador de carga para duplicar el acceso al sitio web y prevenir un corte de algunos de los enlaces de internet.
- c) usaremos la base de datos Oracle existente, pero se deberá subir su actual versión a la 21c.
- d) usaremos lenguaje Python con Django para la generación del sitio y sus funcionalidades.
- e) Se mantendrán los actuales servidores de aplicaciones y base de datos.

Análisis de Reutilización

Comenta si se van a reutilizar algunos componentes de otros sistemas, reutilizar código, modelos de entidad relación etc.

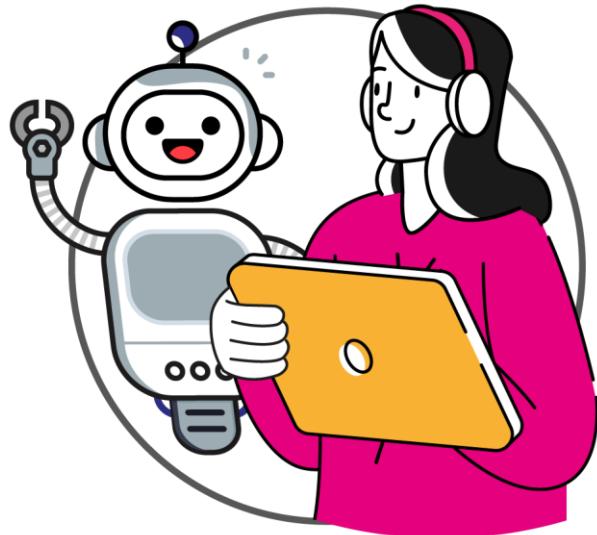
Ejemplo:

Para agilizar el desarrollo, se utilizarán las rutinas de seguridad desarrolladas para el sistema de remuneraciones, con esto se pretende ahorrar una semana de trabajo en los puntos de seguridad del sistema.

Cierre de la semana

Durante esta sexta semana, has aprendido a documentar el documento de arquitectura de sistemas DAS, el cual es un compendio de todo el trabajo realizado hasta la fecha.

Durante las siguientes semanas, tendrán la misión de validar la arquitectura creada, a través de la metodología ATAM, que tengan una excelente semana.



Referencias

Pressman, R. S. (2021). *Ingeniería del Software: Un enfoque práctico* (9th ed.). McGraw Hill.

Lecturas de la semana

Diseño Arquitectónico: Un enfoque recomendado

Pressman, R. S. (2021). *Ingeniería del Software: Un enfoque práctico* (9th ed.). McGraw Hill.

Páginas 181 - 205

Apunte



Duoc UC

Facilitador disciplinar: Juan Alberto Gana.

Asesor par: Domingo Sanz.

Reservados todos los derechos Fundación Instituto Profesional Duoc UC. No se permite copiar, reproducir, reeditar, descargar, publicar, emitir, difundir, de forma total o parcial la presente obra, ni su incorporación a un sistema informático, ni su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia, grabación u otros) sin autorización previa y por escrito de Fundación Instituto Profesional Duoc UC La infracción de dichos derechos puede constituir un delito contra la propiedad intelectual.



Carrera **Analista Programador Computacional**

Exp 3 – Semana 7

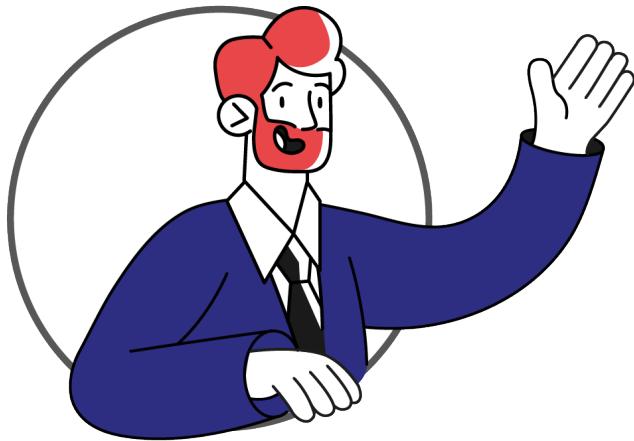
Arquitectura – ASY4231

Guía de aprendizaje

Índice

Introducción a la semana.....	3
Resultado de aprendizaje	4
El estudiante será capaz de:	4
Indicador de logro:.....	4
Conceptos relevantes.....	4
Preguntas activadoras.....	5
Actividad	5
Evaluación de las arquitecturas.....	6
Evaluación tardía.....	7
Evaluación temprana.....	7
Mecanismos para evaluar arquitecturas.....	8
ATAM (Architecture Tradeoff Analisys Method).....	9
Evaluemos nuestra Arquitectura.....	11
Paso 1. Presentar ATAM.....	11
Paso 2. Definamos los escenarios	11
Paso 3. formalicemos los atributos de calidad a utilizar.....	13
Paso 4. Identificar los riesgos.....	13
Paso 5. Medidas de mitigación a los riesgos arquitectónicos.....	14
Paso 6. Evaluación de las consecuencias de mitigación.....	15
Paso 7. Análisis de los trade-off.....	15
Paso 8. Decisiones.....	16
Paso 9. Presentaciones.....	17
Cierre de la semana	18
Referencias	19
Apunte	20

Introducción a la semana



ceremonia que todos los proyectos debiesen realizar por el bien del sistema siendo construido o ya desarrollado.

Aprenderemos sobre evaluación tardía y evaluación temprana, y las fases necesarias para realizar el proceso evaluativo, que tengan una excelente semana.

En esta séptima semana, ya nos acercamos al final del proceso arquitectónico, en este aprenderemos sobre como evaluar la arquitectura desarrollada, realizar los procesos de ajuste necesarios, presentar nuestro trabajo como arquitecto a todos los stakeholders del proyecto y estar abiertos a críticas constructivas y/o correcciones de última hora, lo cual es habitual y debe ser una

Resultado de aprendizaje

El estudiante será capaz de:

RA3. Evaluar el prototipo funcional del modelo arquitectónico propuesto, según los atributos de calidad requeridos.

Indicador de logro:

IL8. Utiliza la metodología ATAM para Validar los escenarios de calidad propuestos en la descripción de arquitectura, sobre el prototipo construido.

IL9. Propone mejoras para la arquitectura propuesta en términos de atributos de calidad y mitigación de riesgos.

IL10. Desarrolla un informe sobre recomendaciones, propuestas de mejoras de calidad futuras y mitigaciones de riesgos observados de acuerdo al prototipo propuesto.

Conceptos relevantes

	Arquitectura	StakeHolder	Árbol de Utilidad
Validación	ATAM	Trade Off	
Consecuencias	Escenarios		

Preguntas activadoras

- ¿Qué es ATAM?
- ¿Qué es validar la arquitectura?
- ¿Qué diferencia hay entre validar la arquitectura en forma temprana o tardía?

Actividad

En la séptima semana, la actividad será formativa y se llama "Validando nuestra arquitectura", donde evaluaremos el resultado de un proceso arquitectónico de ejemplo que hemos visto a lo largo de la asignatura, a través de la metodología ATAM, en la que el arquitecto explica y defiende el modelo arquitectónico desarrollado a los miembros del equipo validador (ustedes como equipo serán el equipo validador).

Este proceso considera validar los escenarios creados para los atributos de calidad, crear nuevas propuestas, mejoras de calidad futuras y mitigaciones de riesgos observados de acuerdo al prototipo propuesto, por lo tanto, de acuerdo a lo aprendido y basado en su experiencia, preparen un breve informe con recomendaciones de mejora para el ejemplo del proceso arquitectónico entregado en la guía de la semana 7.

Evaluación de las arquitecturas.

Antes de introducirnos en los conceptos de validación de las arquitecturas, recordemos un poco, Arquitectura de software es un área de investigación que permite conectar a los usuarios y stakeholders del proyecto con las decisiones que deben ser tomadas para desarrollar el sistema, esto permite mejorar las comunicaciones entre los equipos de trabajo.

La arquitectura, nos permite establecer metas y restricciones para la construcción del sistema, de allí la importancia de esta ya que en forma anticipada podemos detectar mejoras, posibles problemas a futuro y asegurar que el sistema a construir tendrá el balance perfecto entre calidad y funcionalidad que el cliente espera.

Finalmente indicaremos que las arquitecturas son abstracciones de los sistemas, que son fáciles de implementar, además para sistemas con requerimientos similares puede ser reutilizada.



Importante:

La arquitectura de software es la estructura fundamental de un sistema de software, que comprende los componentes del software, las relaciones entre ellos y con el entorno, y los principios y directrices para su diseño y evolución. [Bass, 98]

El siguiente paso es confirmar que todo lo realizado en términos de la arquitectura está bien construido, que se han tomado las decisiones correctas y que estas han sido bien comunicadas al equipo.

Para ello debemos entender que existen mecanismos para validar nuestra arquitectura, estos mecanismos son métodos que pueden ser tempranos o tardíos.

Evaluación tardía

Los mecanismos de evaluación tardía evalúan las arquitecturas existentes, que ya han sido implementadas, buscan principalmente evaluar si lo desarrollado cumple con los objetivos requeridos. Algunos mecanismos que se pueden utilizar son:

- Análisis de cumplimientos de requerimientos, buscan comparar lo desarrollado con los requisitos solicitados y encontrar brechas entre ellos.
- Análisis de rendimiento, buscan establecer el resultado de lo desarrollado en término de tiempo de respuesta, velocidad de transacciones, etc.
- Análisis de seguridad, busca posibles brechas de seguridad, en la cual nuestros sistemas podrían ser intervenidos maliciosamente por agentes externos a la organización.

En general estos mecanismos y otros existentes buscan posibles mejoras en y no cumplimientos con lo establecido por los clientes, lo cual no asegura que este del todo correcto, por esto se recomienda realizar la evaluación en forma temprana, cuando el diseño está comenzando.

Evaluación temprana

Dado que la arquitectura tiene un profundo impacto en el proyecto, y que además es uno de los primeros entregables que el proyecto desarrolla, es de suma importancia realizar una evaluación lo más temprano posible el proyecto, ya que los defectos que puede tener el producto, es mejor que sean detectados durante la etapa de diseño y no cuando esté terminado, ya que el costo de corrección es mucho más elevado una vez que ya esto todo construido.

Dado que la fase de validación de la arquitectura es antes que la implementación (pudiendo ser realizada por fases, sprint o cualquier iteración del proyecto), pude ayudarnos a dilucidar algunas interrogantes como, por ejemplo, ¿las decisiones tomadas son las más adecuadas?, o ¿existen otros enfoques que podemos evaluar para obtener una mejor arquitectura?, el proceso de evaluación compuesto por el equipo evaluador y los stakeholders, son los encargados de responder a estas interrogantes.

Mecanismos para evaluar arquitecturas

Los mecanismos o métodos para evaluar arquitecturas persiguen revisar que el diseño se apegue a las necesidades del producto, en términos de atributos de calidad, por ejemplo, la seguridad, el rendimiento, la tolerancia fallas y la adecuación funcional, algunos de ellos son:

- **SAAM (Software Architecture Analisys Method)**, este método busca la validación de la arquitectura basado en escenarios, es iterativa y congrega a distintos equipos para avalar las decisiones arquitectónicas, busca validar como la arquitectura se comporta con los distintos escenarios planteados.
- **CBAM (Cost benefit Analisys method)**, es un mecanismo que evalúa la arquitectura en función de los costos incurridos y los beneficios entregados, alinea a los distintos equipos implicados en la arquitectura, se revisan atributos de calidad y consideraciones económicas.
- **ATAM (Architecture Tradeoff Analisys Method)**, es un método sistemático y estructurado, que permite analizar los distintos atributos de calidad clave definidos, analiza el cumplimiento de los requerimientos del cliente, y las decisiones arquitectónicas, así como los distintos tradeoff, los cuales buscan el equilibrio entre los distintos aspectos, en el cual algunas consideraciones se sacrifican en función de la mejora en otros. Este último método será el que aplicaremos para nuestro proceso de evaluación.

ATAM (Architecture Tradeoff Analisys Method)

El principal objetivo de **ATAM** proporcionar al grupo evaluador, al team de desarrollo y al arquitecto de software, obtener la certeza que las decisiones arquitectónicas tomadas son las mejores para el desarrollo del producto.

ATAM tiene una estructura básica, la cual se puede acomodar a las necesidades del proyecto, un ejemplo es:

- Preparación, Se definen objetivos, contexto y los requerimientos del sistema siendo construido. Se conforma el equipo evaluador, formado por los distintos stakeholders.
- Presentación, El equipo de arquitectura presenta los objetivos, la arquitectura propuesta, atributos de calidad al grupo evaluador, se crean los distintos escenarios que evaluarán a la arquitectura.
- Análisis de riesgos, se identifican los principales riesgos asociados a la arquitectura propuesta, se analizan los distintos tradeoff entre atributos de calidad y se revisan los compromisos y limitaciones.
- Evaluación de la arquitectura, se realizan simulaciones, para ver el comportamiento de los atributos de calidad propuestos contra los distintos escenarios creados.
- Presentación de resultados, se informa al grupo evaluador, los distintos hallazgos, problemas y resultado de los escenarios, se realizan mejoras y se establece un plan de acción para enfrentar los problemas encontrados y así tomar decisiones de mejor calidad.

Análisis de sensibilidad

El análisis de sensibilidad es muy relevante en el proceso ATAM, ya que en este punto se ven expuestos los atributos de calidad y las distintas decisiones.

Un punto de sensibilidad (sensitivity point) es una decisión arquitectónica a evaluar que afecta a uno o más componentes de la arquitectura, los cuales son críticos para lograr la respuesta de los atributos de calidad, la medida de respuesta de un atributo es sensible a cambios de las decisiones. Ejemplos de esto son:

- Si evaluamos el atributo de calidad seguridad, donde esperamos que el sistema y su entorno este seguro, es sensible a las decisiones de cómo se implementan las capas de autenticación, perfiles de seguridad y valuación de datos.
- Si evaluamos el rendimiento, velocidad en el tiempo de respuesta, es sensible por ejemplo a la decisión de optar por algún tipo de hardware o bien la ubicación del servidor de la base de datos on-premise o cloud.
- Si la Fiabilidad está siendo evaluada como punto de sensibilidad, esta puede estar en riesgo ante la decisión por ejemplo de la ubicación de un servidor puesto en una bodega sin seguridad.

Trade-off

Los trade off son las decisiones arquitectónicas que pueden afectar a más de un atributo de calidad, aquí son necesarias los análisis y negociaciones, ya que por ejemplo, en el caso anterior, donde se indicaba el atributo de calidad de Fiabilidad y se hablaba de colocar el servidor en la bodega sin seguridad, aquí el trade off es , mover el servidor al datacenter principal, pero a costa de tener que remover otro servidor o bien invertir dinero en agregar mecanismos de protección para un nuevo servidor, en otras palabras, se deben sacrificar algunos puntos por el beneficio que conlleva una mejora a uno de los atributos de calidad.

Evaluemos nuestra Arquitectura

Basado en el caso que estamos siguiendo estas últimas semanas, realizaremos un ejemplo de cómo se debiese hacer la evaluación de la arquitectura, algunas consideraciones:

- a) Usaremos un mecanismo de evaluación temprana, es decir, aún no se construye el sistema.
- b) Nuestros escenarios estarán orientados a nuestro sistema de ventas on-line.
- c) Los atributos de calidad en los que pondremos énfasis son usabilidad, fidelidad y seguridad.

Paso 1. Presentar ATAM.

La información sobre ATAM, tipo de evaluación, si será temprana o tardía, el proceso en general debe ser explicado al grupo evaluador, para ello presente los conceptos lo más simple que pueda, recuerden que los stakeholders no necesariamente son personas del ámbito técnico, por lo cual es necesario controlar el lenguaje técnico.

En esta presentación deben explicarse las metas y restricciones de la arquitectura, como cualquier otro punto de vista que se pueda usar para avalar la arquitectura realizada.

Paso 2. Definamos los escenarios.

Es necesario identificar los posibles escenarios donde nuestro sistema se va a desempeñar a futuro, es clave que imaginemos todas las posibilidades que se podrían enfrentar en un momento determinado, esto permitirá reforzar la arquitectura con nuevos puntos de vista los cuales podrían no haber sido considerados en el diseño inicial. A continuación, un ejemplo de nuestro proyecto.

Escenario 1. el sistema debe estar funcionando el 99.7% del tiempo, nuestros clientes finales podrán utilizar el sistema a la hora que estimen conveniente, por lo que se espera que el servidor esté funcionando 24x7. Se dejará un pequeño margen de 0.3% del tiempo, por temas imprevistos, esto significa que al día está permitido que el sistema pueda estar con algún incidente solo 30 minutos.

Escenario 2. Ante la falla de algún componente de hardware, como falla en el servidor, el equipo de TI tiene la capacidad y los recursos para de inmediato dejar un servidor secundario funcionando en un corto período de tiempo.

Escenario 3. los clientes para proceder a comprar deberán autenticarse en el sistema, para ello los clientes nuevos, deberán primeramente registrarse en el sistema, se espera que los usuarios no tengan complicaciones y que creen su usuario al primer intento.

Escenario 4. Hackers podrían intentar robar la información de nuestros clientes, para ello los clientes usarán un doble factor de identidad, así prevenir que, ante un hurto de contraseñas, estas no puedan ser usadas ya que faltaría el segundo nivel de seguridad.

Una vez preparados los escenarios, se deben crear los árboles de utilidad, los cuales mostrarán en forma de resumen como la arquitectura se apega a los requisitos del cliente.

Esto generará una lista priorizada de los atributos de calidad evaluados.

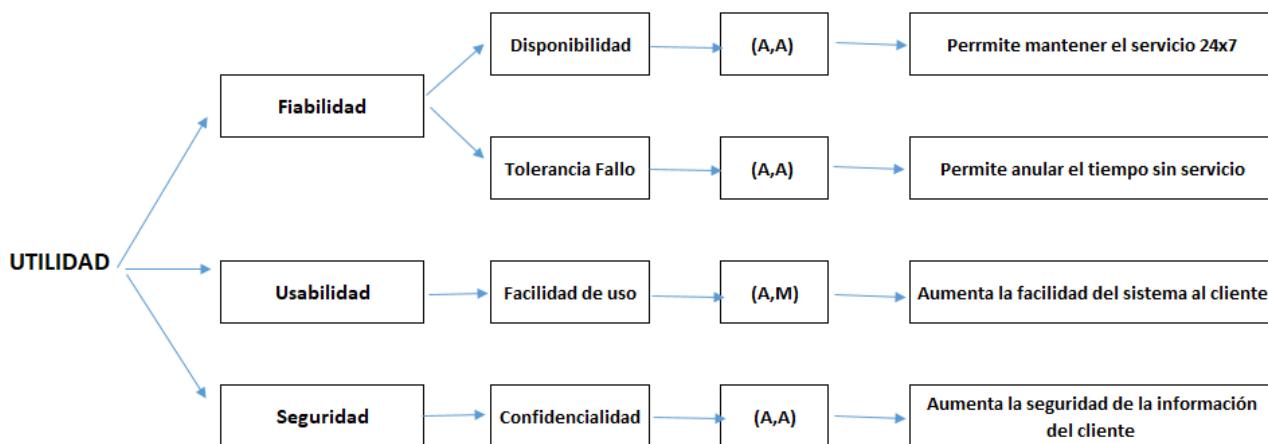
Árboles de decisión

La forma de evaluar es plantear los atributos de calidad desde 2 dimensiones, 1) que tan importante es para el cliente y 2) que tan costoso es su implementación. Esto nos dará una lista priorizada.

La escala está dada por 2 dimensiones (**Importancia para cliente, Complejidad de implementar**), y se ajusta según los siguientes parámetros, los cuales sirven para ambas dimensiones:

- a) Alto
- b) Medio
- c) Bajo

Un ejemplo árbol de utilidad es como se muestra en la figura 1. Es este caso, la Disponibilidad es “A”, de alta importancia para el cliente, pero su complejidad es también “A”, alto. En el caso de Facilidad de uso, la importancia para el cliente es “A”, alto, pero la complejidad es “M”, o sea no es compleja de implementar del todo.

Figura 1:*Árbol de utilidad.*

Nota: Árbol de utilidad.

Paso 3. formalicemos los atributos de calidad a utilizar.

Si bien es cierto existe un gran número de atributos de calidad, aquí es importante incorporar los más relevantes, los que podrían poner en riesgo la estabilidad del sistema una vez que esté en ambiente productivo. A continuación, un ejemplo de nuestro proyecto.

- a) Atributo Fiabilidad, característica Disponibilidad (escenario 1)
- b) Atributo Fiabilidad, característica Tolerancia a fallos (escenario 2)
- c) Atributo Usabilidad, característica Facilidad de uso (escenario 3)
- d) Atributo Seguridad, característica Confidencialidad (escenario 4)

Paso 4. Identificar los riesgos.

Tal como mencionamos semanas atrás, el peor riesgo es no reconocer la existencia de estos. Por lo que en este punto debemos evaluar los riesgos inherentes al proceso, con énfasis en los escenarios y atributos de calidad seleccionados. A continuación, un ejemplo de nuestro proyecto.

Riesgo 1, que haya períodos de falla, no considerados y que no exista personal de turno, durante el período donde se produce la falla.

Riesgo 2, El servidor secundario, no tenga la misma versión de sistema operativo o nivel de parchado del servidor primario, por lo que el software podría no ser ejecutado correctamente.

Riesgo 3, Que el sistema no sea lo suficientemente fácil de operar, sobre todo en el doble factor de identidad.

Riesgo 4, Que el usuario no cuente a la mano con los dispositivos solicitados para el doble chequeo de identidad, o la persona no esté familiarizado con la tecnología.

Paso 5. Medidas de mitigación a los riesgos arquitectónicos.

Si hemos detectado posibles riesgos, debemos indicar cuáles serán las medidas de mitigación, estas son actividades que desarrollaremos para mejorar las prestaciones del sistema cuando este en plena ejecución en también productivo. A continuación, un ejemplo de nuestro proyecto.

Medida 1. contratar un servicio externo de monitoreo 24x7, este servicio nos proveerá la tranquilidad que nuestros sistemas están funcionando 24x7, y en caso de algún incidente, el servicio contactará a nuestro personal autorizado para preocuparse del incidente y su corrección. Mitiga al riesgo 1.

Medida 2. al momento de desplegar en la infraestructura el nuevo servidor de respaldo o secundario, realizar un proceso de check-list para comprobar que es exactamente igual en términos de sistema operativo, parches y versiones en general de los componentes. Mitiga el riesgo 2

Medida 3. Deberá existir un grupo de soporte el cual podrá orientar al usuario vía telefónica o en línea con Whatsapp, así el cliente puede tener una respuesta rápida al problema que este presentando al momento de su operación. Mitiga al riesgo 3.

Medida 4. mantener un equipo de soporte que pueda orientar al usuario en como operar el dispositivo móvil que tendrá que usar para responder al doble factor de identidad.

Paso 6. Evaluación de las consecuencias de mitigación.

Aquí es necesario enfrentar las distintas realidades para cada medida de mitigación, ya sea en recursos monetarios como recursos de infraestructura y tiempo. A continuación, un ejemplo de nuestro proyecto.

Tabla 1: Evaluación de las consecuencias de la mitigación

Mitigación	Descripción	Costo / Recurso
Medida 1, contratar un servicio externo de monitoreo 24x7, este servicio nos proveerá la tranquilidad que nuestros sistemas están funcionando 24x7, y en caso de algún incidente, el servicio contactará a nuestro personal autorizado para preocuparse del incidente y su corrección. Mitiga al riesgo 1.	Habilitar un servicio externo para el monitoreo implicará desarrollar un presupuesto mensual para el pago de los servicios de monitoreo externos, en estos casos se recomienda que el servicio sea externo para mantener la transparencia del servicio.	CLP 1,500,000 mensuales.
Medida 2, al momento de desplegar en la infraestructura el nuevo servidor de respaldo o secundario, realizar un proceso de check-list para comprobar que es exactamente igual en términos de sistema operativo, parches y versiones en general de los componentes. Mitiga el riesgo 2	Esta actividad puede estar enmarcada en los actuales servicios de TI nos entrega. Se trata de un checklist del servidor, el cual arrojará posibles diferencias en la infraestructura. Si existen, estas deberán ser ejecutadas antes que la plataforma este en ambiente productivo.	Sin costo de operación, eventualmente costo de licencias si es que aplica instalarlas.
Medida 3, Deberá existir un grupo de soporte el cual podrá orientar al usuario vía telefónica o en línea con WhatsApp, así el cliente puede tener una respuesta rápida al problema que este presentando al momento de su operación. Mitiga al riesgo 3	Probablemente este punto de mitigación va a requerir un presupuesto adicional, dado que serán personas que estarán de cara al servicio al cliente.	Costo de una persona en call center, aproximadamente CLP 700,000 p/p.
Medida 4, mantener un equipo de soporte que pueda orientar al usuario en como operar el dispositivo móvil que tendrá que usar para responder al doble factor de identidad. Mitiga al riesgo 4	Esta es la misma consideración para el punto de mitigación 3, eventualmente se puede hacer una sinergia entre ambos procesos.	Costo de una persona en call center, aproximadamente CLP 700,000 p/p.

Paso 7. Análisis de los trade-off.

Tal como fue definido anteriormente, los trade-off serán actividades en las cuales deberemos sacrificar en algunas situaciones o atributos de calidad por el bien de otras.

Los atributos de calidad definidos fueron los siguientes:

- a) Atributo Fiabilidad, característica Disponibilidad (escenario 1)

- b) Atributo Fiabilidad, característica Tolerancia a fallos (escenario 2)
- c) Atributo Usabilidad, característica Facilidad de uso (escenario 3)
- d) Atributo Seguridad, característica Confidencialidad (escenario 4)

Todos ellos requieren recursos para su implementación, pero si como arquitectos tuviésemos que tomar una decisión en términos de importancia,

Trade Off 1. La importancia de la seguridad de la información es una gran preocupación de la empresa, por lo que revisado los casos, aplicar doble factor de seguridad nos dará la tranquilidad que nuestros clientes podrán contar con un sitio seguro donde su información esté segura y libre de accesos no deseados, hemos decidido, no aplicar en los accesos y pantallas de autenticación la facilidad de uso esperada, es decir, las pantallas deberán contener la información necesaria para el doble factor de identidad, se deberán usar dispositivos móviles para receptionar la segunda clave. Sabiendo que esto complicará el uso rápido e intuitivo de nuestro sistema, nuestros clientes se verán beneficiado al final con un sitio seguro, luego nuestro trade off es bajar la usabilidad e incorporar mejoras a través de un doble factor de identidad.

Trade Off 2. Dato que se pretende instalar el servidor que está en la bodega en el datacenter principal, esto se logra solo si se remueve otro servidor desde el datacenter ya que el espacio disponible no alcanza para todos. En este caso, vamos a bajar el nivel de tolerancia a fallos de las máquinas de dominio, aumentando la disponibilidad del servidor de las aplicaciones de ventas web.

Paso 8. Decisiones.

Una vez realizado el análisis de los atributos de calidad, riesgos y trade-off, será necesario tomar las decisiones correspondientes.

En nuestro caso las decisiones serán avalar los atributos de calidad y el resultado de los escenarios propuestos, junto a los riesgos detectados, se ha tomado la decisión de:

1.- Se aplicarán todas las medidas necesarias para cumplir con la Seguridad, aplicando doble factor de seguridad para mantener la confidencialidad de los datos del cliente.

2.- Se procede a trasladar el servidor desde las bodegas, hacia el datacenter principal, lo que aumentará la disponibilidad del servicio de ventas web. Sin embargo, esta decisión activará el proceso de remover desde el datacenter, el servidor de dominio, dado que no existe el espacio suficiente, esto podría provocar situaciones riesgosas durante la autenticación de los usuarios internos de la empresa. Se dejará nota de la situación para que sea levantada en un futuro proyecto de expansión de la infraestructura.

3.- El proceso de desarrollo, deberá incorporar doble factor de identidad, en este caso se espera que el cliente al momento de autenticarse, le sea enviado un código a través de internet el cual llegará a su dispositivo móvil. En ese caso, el cliente deberá usar el código de activación en un tiempo no superior a 1 minuto, luego de lo cual el token enviado quedará anulado y el cliente deberá realizar el proceso nuevamente. Sabemos que esta decisión puede impactar en la facilidad de uso del sistema, pero incrementa sustancialmente la confidencialidad, al intensificar la autenticación del usuario.

Importante



Estas decisiones podrían afectar presupuestariamente al proyecto, ya que deberá incurrir en gastos no planificados.

Paso 9. Presentaciones.

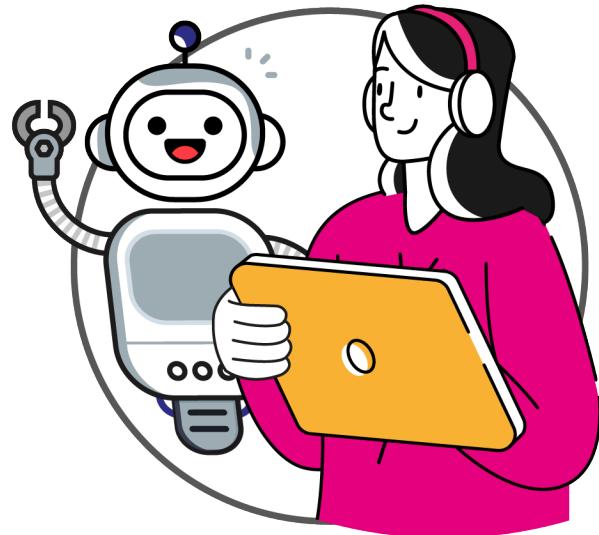
En la etapa de presentaciones de la información, el equipo debe resumir todos los puntos trabajados, preparar algún material audiovisual para presentar los resultados a los diferentes **stakeholder** el proyecto.

En la etapa de presentaciones podrían efectuarse algunos **brainstorming** en los cuales los comentarios del equipo podrían eventualmente, cambiar decisiones arquitectónicas, se le puede pedir al arquitecto tomar nuevos cursos, para enfrentar los escenarios y así robustecer el proceso, como también se podrían solicitar nuevos escenarios no considerados durante el proceso, esto debe provocar que se haga una nueva iteración de los pasos 1 al 8.

Cierre de la semana

Durante esta séptima semana, has aprendido acerca de cómo validar la arquitectura de software, hemos utilizado para ello la metodología ATAM, la cual es un estándar para evaluación de arquitecturas de software, hemos aprendido que esta evaluación, no solo impacta al equipo de desarrollo, también a todos los stakeholders, por lo cual es muy importante hacerlos partícipe de esta mecánica de evaluación.

Esperamos haya sido una gran semana de aprendizaje, los espero en la siguiente semana, donde seguiremos profundizando en los diferentes temas referidos a la arquitectura de software, tengan una buena semana.



Referencias

- ✓ Morales, Pavel, "Vista lógica".

<https://www.youtube.com/watch?v=Qlh989ihCmQ>

- ✓ Morales, Pavel, "Vista de Escenarios".

<https://www.youtube.com/watch?v=kZYcgl96cJ4>

- ✓ Morales, Pavel, "Vista de Procesos".

https://www.youtube.com/watch?v=yPp_jGFnnMY&list=PLjK8W0kxXIDaUwKGNKTjvj0RzqJOR9bwE&index=3

- ✓ Morales, Pavel, "Vista de Despliegue".

<https://www.youtube.com/watch?v=xkp4CLtlpb0>

- ✓ Morales, Pavel, "Vista Física".

<https://www.youtube.com/watch?v=f0lG-QY7ciQ&list=PLjK8W0kxXIDaUwKGNKTjvj0RzqJOR9bwE&index=4>

Apunte

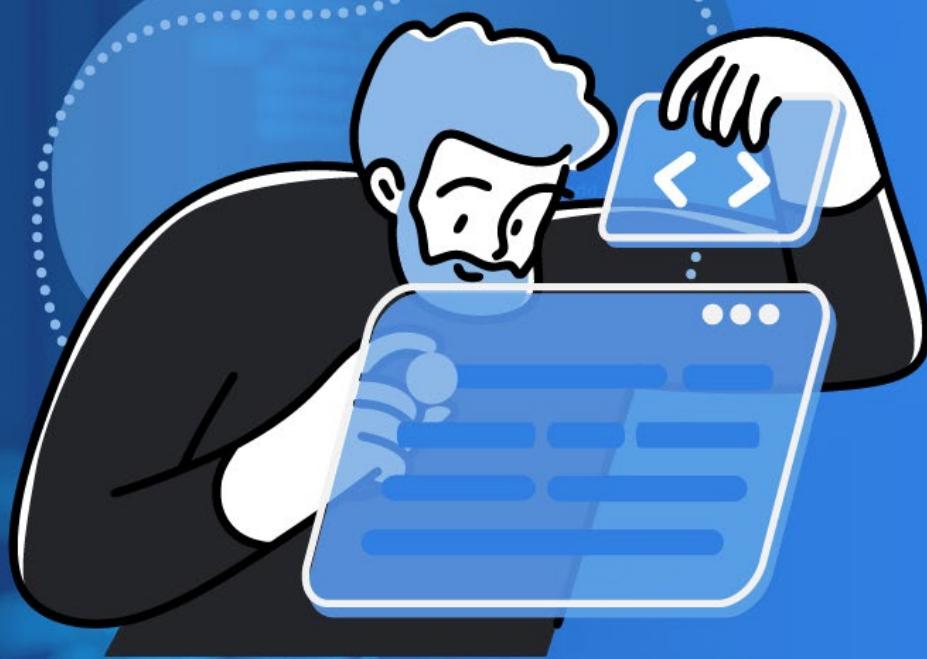


Duoc UC

Facilitador disciplinar: Juan Alberto Gana.

Asesor par: Domingo Sanz.

Reservados todos los derechos Fundación Instituto Profesional Duoc UC. No se permite copiar, reproducir, reeditar, descargar, publicar, emitir, difundir, de forma total o parcial la presente obra, ni su incorporación a un sistema informático, ni su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia, grabación u otros) sin autorización previa y por escrito de Fundación Instituto Profesional Duoc UC La infracción de dichos derechos puede constituir un delito contra la propiedad intelectual.



Carrera
**Analista Programador
Computacional**

Exp 3 – Semana 8

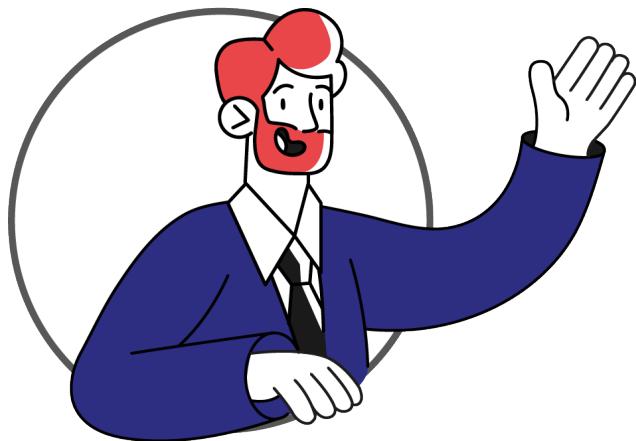
Arquitectura – ASY4231

Guía de aprendizaje

Índice

Introducción a la semana.....	3
Resultado de aprendizaje	4
El estudiante será capaz de:	4
Indicador de logro:.....	4
Conceptos relevantes.....	4
Preguntas activadoras.....	5
Actividad	5
Cierre de la semana	11
Referencias	12
Apunte	13

Introducción a la semana



En esta octava semana, pondremos en práctica lo aprendido en la última semana y realizaremos una evaluación de nuestra arquitectura, desde el punto de vista del grupo evaluador, es decir stakeholders, equipo de desarrollo, proveedores, etc.

En la semana anterior, se explicó sobre tipos de evaluación de arquitectura, ya sean tempranas o tardías, además se les presentó

un ejemplo de cómo implementar un proceso de validación ATAM sobre el ejemplo que hemos venido trabajando durante las últimas semanas.

La actividad formativa de esta semana será simular en proceso de presentación del proceso evaluativo de la arquitectura, donde tú y tus compañeros deberán usar el ejemplo entregado la semana pasada, basándose en lo aprendido y en su experiencia laboral, deberán preparar un informe libre en el cual destaque las fortalezas y debilidades del ejemplo entregado, es decir, deberán juzgar en trabajo realizado y proponer mejoras, al proyecto entregado en términos de arquitectura.

Resultado de aprendizaje

El estudiante será capaz de:

RA3. Evaluar el prototipo funcional del modelo arquitectónico propuesto, según los atributos de calidad requeridos.

Indicador de logro:

IL8. Utiliza la metodología ATAM para Validar los escenarios de calidad propuestos en la descripción de arquitectura, sobre el prototipo construido.

IL9. Propone mejoras para la arquitectura propuesta en términos de atributos de calidad y mitigación de riesgos.

IL10. Desarrolla un informe sobre recomendaciones, propuestas de mejoras de calidad futuras y mitigaciones de riesgos observados de acuerdo al prototipo propuesto.

Conceptos relevantes

	Arquitectura	StakeHolder	Árbol de Utilidad
	Validación	ATAM	Trade Off
	Consecuencias	Escenarios	Riesgo

Preguntas activadoras

- ¿Qué es ATAM?
- ¿Qué es validar la arquitectura?
- ¿Qué diferencia hay entre validar la arquitectura en forma temprana o tardía?

Actividad

En esta octava semana tendrán la actividad sumativa de la Experiencia 3, donde se entregará el documento ATAM. Para esto la actividad se dividirá en 3 partes.

Parte I. la entrega de la documentación, en equipo deberán generar y entregar el documento ATAM.

Para la Parte II, deberán presentar los resultados y argumentar cada uno de ellos.

Para la Parte III, deberán generar un espacio de reflexión sobre los aprendizajes adquiridos de manera individual

Evaluación de las arquitecturas.

Como ya es de su conocimiento, las distintas etapas de una validación de arquitectura, basada en metodología ATAM son las siguientes:

Paso 1. Presentar ATAM.

El equipo de arquitectura, liderado por el arquitecto de software, le presentará al grupo evaluador, que significa una metodología de evaluación de arquitectura, cuáles son sus alcances y etapas que se verán informadas en el proceso. Es una presentación en algún medio visual que permita a la audiencia entender el proceso en el que serán vinculados al revisar la arquitectura del software.

Paso 2. Definamos los escenarios.

La definición de escenarios por parte del equipo de arquitectura, puede ser un riesgo, ya que tal vez es un grupo que está lejano de la operación como tal del nuevo sistema, por lo cual podrían omitir situaciones que el negocio (empresa/cliente) ve en forma habitual durante su operación normal, por ejemplo si se está trabajando para desarrollar un sistema eCommerce el cual tendrá afluencia de público, esta cantidad de potenciales consumidores podría tener un alto tráfico en alguna semana donde existan ofertas, como lo es, los días Black Friday o Cyber week. En este caso su papel como miembro del equipo evaluador, es revisar los escenarios que se van a proponer y así proponer nuevos escenarios en el caso que usted así lo determine.

Paso 3. Formalicemos los atributos de calidad a utilizar.

El equipo de arquitectura se basará en los requisitos no funcionales para determinar los atributos de calidad, las personas que habitualmente desarrollan software tienen una vista más orientada a la funcionalidad, por lo cual los atributos de calidad, podrían no ser pensados desde el inicio del sistema, como preocuparse de la usabilidad, seguridad, mantenibilidad, portabilidad, etc.

Su papel en este punto es afirmar o bien complementar las decisiones de arquitectura, ayudando al equipo a pensar en los distintos atributos que podrían ser necesarios para mejorar el desempeño del sistema, sus cargas mínimas y máximas en términos de consumir recursos de la infraestructura y/o mejorar la seguridad del ambiente productivo. Ver la siguiente imagen para mayor claridad en los atributos de calidad.

Gráfico 1:*Calidad de producto de software*

Nota: Calidad del producto de software

Paso 4. Identificar los riesgos.

En todo proyecto, es imprescindible descubrir y gestionar los riesgos. Un riesgo es una condición que nos podría afectar en el proyecto en cualquier momento, por lo que debemos abrir nuestras mentes y pensar en las cosas más insólitas que podrían ocurrir en un proyecto, por ejemplo , desarrollando un proyecto y enfrentar una reunión muy importante con el cliente, pareciera no ser algo riesgoso, pero que pasa si hay un corte de tráfico, que pasa si no tenemos cargada nuestra tarjeta de pago de locomoción, o no tenemos dinero en nuestra cuenta para pagar en línea, tal vez vamos en nuestro propio vehículo, debemos cargar

combustible, hay tráfico, no hay donde estacionar, etc., todas estas situaciones , son riesgos para nuestra reunión con el cliente, lo que provocará un retraso en ella y hasta una posible cancelación.

Lo importante es que cuando hemos detectado algún riesgo, debemos inmediatamente registrarla, y realizar seguimiento, junto con esto, debemos asignar un responsable, eventualmente un presupuesto y hacerle seguimiento, ya que como han aprendido en otros ramos, un riesgo no se elimina, solo lo podemos mover de lugar, compartirlo o bien mitigarlo.

Este último punto es de suma importancia, ya que nuestra arquitectura está abierta a contener riesgos, requisitos mal entendidos, diagramas mal hechos, tiempos de respuesta lentos, etc.

En resumen, su papel en este punto es revisar que ustedes como grupo validen que los riesgos y las medidas de mitigación propuestas en la arquitectura, sean consistentes con el proyecto y verificar que no existan otros riesgos que no han sido documentados. En la eventualidad que vea riesgos no detectados, es su responsabilidad como equipo, hacerlos presente y trabajar en conjunto para evaluar y diseñar alguna medida de mitigación.

Paso 5. Medidas de mitigación a los riesgos arquitectónicos.

Tal como se comentó en el punto anterior (Paso 4), es necesario que cada riesgo identificado, cuente con un plan de mitigación, en este caso el proyecto de ejemplo, informó de algunas medidas de mitigación, su rol, debe validar que estas son suficientes para mitigar el impacto de que esos riesgos se materialicen, en otras palabras , verifique que las medidas de mitigación son las necesarias y que no sean necesarias nuevas medidas, si no está de acuerdo o bien cree que existen otras mitigaciones, haga las indicaciones al grupo durante la sesión de discusión de la validación de la arquitectura.

Paso 6. Evaluación de las consecuencias de mitigación.

Si bien las medidas de mitigación harán que el proyecto tenga un mejor pronóstico en caso que los riesgos sean materializados, las medidas propuestas, podrían causar que otros riesgos sean descubiertos o bien que nuestro proyecto se debilite producto de estas medidas. En el caso que un riesgo nos obligue a entregar presupuesto para su control, esta medida, si bien nos ayuda a mitigar el riesgo, podría provocar que el proyecto quede sin presupuesto anticipadamente, o bien si riesgo era debido a obsolescencia de alguna infraestructura, la medida de mitigación, podría ser reemplazar o comprar un nuevo hardware, con la posibilidad que esta medida traiga nuevos riesgos como por ejemplo, que el personal requiera de entrenamiento ya que el hardware es desconocido, etc. Su papel es prevenir y discutir sobre el alcance las medidas de mitigación.

Paso 7. Análisis de los trade-off.

Tal como habíamos indicado, un trade-off en este caso es un intercambio, ya que debemos evaluar que, para cumplir un cierto propósito, eventualmente debemos dejar otro de lado. En este caso su misión es determinar que los puntos de intercambio sean razonables y que el proyecto no quede débil al tomar una decisión que implique reemplazar alguna pieza de la arquitectura por otra, o bien eliminar una actividad por temas de presupuesto, etc.

Paso 8. Decisiones.

Finalmente, en este punto, el equipo de arquitectura mostrará las decisiones a las que han llegado, es el punto donde usted con su grupo pueden debatir estas, haciendo los comentarios e indicaciones necesarias para poder mejorar el proceso, un nuevo punto de vista hace que el proyecto pueda ser desarrollado de mejor manera y así fortalecer a los equipos de trabajo.

Apoye las decisiones, o bien, ayude a que se generen nuevas decisiones producto de la revisión del proceso.

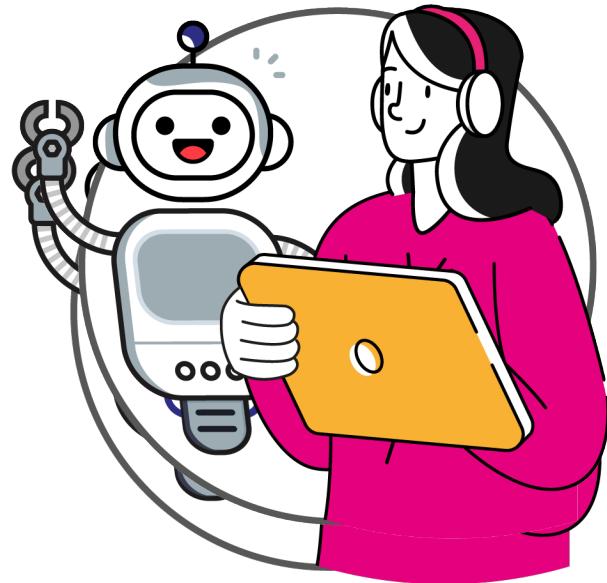
Paso 9. Presentaciones.

Finalmente, en este punto puede evaluar que el material gráfico, documentación, presentaciones en algún medio audiovisual, estén acordes para poder mostrar los resultados a los stakeholders y miembros del equipo evaluador.

Cierre de la semana

Durante esta semana, has aprendido el valor de poder evaluar los pasos de ATAM, revisando el material preparado como ejemplo, el cual tendrá muchos aspectos que se reflejan en la vida real de un proyecto, ya que la arquitectura es crucial para su desarrollo.

Esperamos haya sido una gran semana de aprendizaje, los espero en la siguiente semana para finalizar nuestra asignatura con el proyecto final.



Referencias

- ✓ Morales, Pavel, "Vista lógica".

<https://www.youtube.com/watch?v=Qlh989ihCmQ>

- ✓ Morales, Pavel, "Vista de Escenarios".

<https://www.youtube.com/watch?v=kZYcgl96cJ4>

- ✓ Morales, Pavel, "Vista de Procesos".

https://www.youtube.com/watch?v=yPp_jGFnnMY&list=PLjK8W0kxXIDaUwKGNKTjvj0RzqJOR9bwE&index=3

- ✓ Morales, Pavel, "Vista de Despliegue".

<https://www.youtube.com/watch?v=xkp4CLtlpb0>

- ✓ Morales, Pavel, "Vista Física".

<https://www.youtube.com/watch?v=f0lG-QY7ciQ&list=PLjK8W0kxXIDaUwKGNKTjvj0RzqJOR9bwE&index=4>

Apunte



Duoc UC

Facilitador disciplinar: Juan Alberto Gana.

Asesor par: Domingo Sanz.

Reservados todos los derechos Fundación Instituto Profesional Duoc UC. No se permite copiar, reproducir, reeditar, descargar, publicar, emitir, difundir, de forma total o parcial la presente obra, ni su incorporación a un sistema informático, ni su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia, grabación u otros) sin autorización previa y por escrito de Fundación Instituto Profesional Duoc UC La infracción de dichos derechos puede constituir un delito contra la propiedad intelectual.