

Ingeniería del Software



Autor: Jon Vadillo
www.jonvadillo.com

REQUIREMENTS VS. IMPLEMENTATION

- SIMPLE INTERFACE - ACCOMMODATE ALL USERS
- CUSTOMIZABLE
- SECURE
- LOW MAINTENANCE

REQUIREMENTS



BRAINSTORMING



BUDGET



IMPLEMENTATION



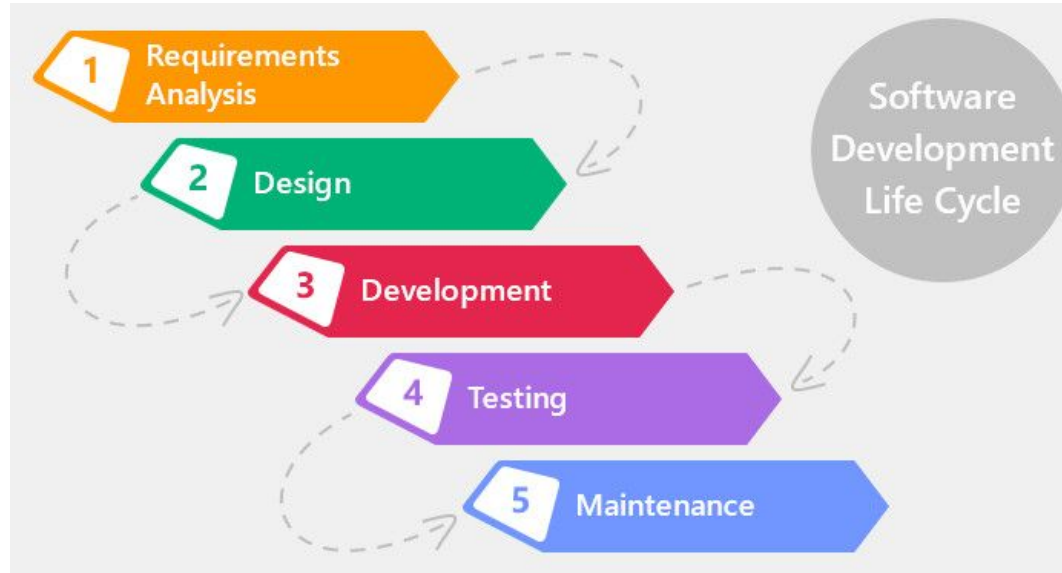
Gestión de Proyectos

- ❑ La gestión de proyectos es la disciplina que agrupa y ordena el **conjunto de tareas o actividades destinadas a obtener unos objetivos.**
- ❑ Esto incluye:
 - ❑ **Planificación, definición, ordenamiento y gestión** de las **actividades que formarán el proyecto software.**
- ❑ Reduce al mínimo las posibilidades de fallo durante el transcurso del proyecto.
 - ❑ La no consecución de los requisitos iniciales del proyecto
 - ❑ La inviabilidad económica del mismo
 - ❑ Un resultado que impida mantenerlo o le impida evolucionar
 - ❑ etc

Ciclo de vida del SW

Fases por las que pasa un proyecto de software desde que es concebido, hasta que está listo para usarse.

Desarrollo en cascada (waterfall)



Source: <https://www.linkedin.com/pulse/what-software-development-life-cycle-sdlc-phases-private-limited/>

Waterfall / Repetitivo



MÉTRICA v3

- ❑ Es una metodología de planificación, desarrollo y mantenimiento de sistemas de información para la **sistematización de actividades del ciclo de vida de los proyectos software**.
- ❑ Procesos principales:
 - ❑ **Planificación** de Sistemas de Información (PSI)
 - ❑ **Desarrollo** de Sistemas de Información (DSI)
 - ❑ **Mantenimiento** de Sistemas de Información (MSI)
- ❑ Cada proceso de los anteriores detalla las actividades y tareas que hay que realizar:
 - ❑ Las técnicas y prácticas a utilizar
 - ❑ Los responsables de realizarla.
 - ❑ Sus productos de entrada y salida.
- ❑ Se apoya en el estándar de modelado UML

Arinbide

| | | |
|----------|----------------------------------------------------------------------|-----------|
| 2 | Requisitos de usuario | 6 |
| 2.1 | Esquema general..... | 6 |
| 2.2 | ASI 1. Definición del sistema | 7 |
| 2.3 | ASI 2.1. Establecimiento de requisitos | 7 |
| 2.4 | ASI 11.1. Presentación y aprobación del catálogo de requisitos | 8 |
| 3 | Análisis del sistema de información | 9 |
| 3.1 | Esquema general..... | 9 |
| 3.2 | ASI 3. Identificación de subsistemas de análisis..... | 11 |
| 3.3 | ASI 2.2. Refinamiento de requisitos y casos de uso | 11 |
| 3.4 | ASI 4. Análisis de casos de uso..... | 11 |
| 3.5 | ASI 5. Análisis de clases..... | 12 |
| 3.6 | ASI 8. Definición de interfaces de usuario..... | 12 |
| 3.7 | ASI 9. Análisis de consistencia | 13 |
| 3.8 | ASI 11.2. Presentación y aprobación del ERS..... | 13 |
| 4 | Diseño del sistema de información | 14 |
| 4.1 | Esquema general..... | 14 |
| 4.2 | DSI 1. Definición de la arquitectura del sistema | 15 |
| 4.3 | DSI 3.3. Diseño de la interfaz de usuario | 16 |
| 4.4 | DSI 4. Diseño de clases | 16 |
| 4.5 | DSI 6. Diseño físico de datos | 17 |
| 4.6 | DSI 8. Generación de especificaciones de construcción..... | 17 |
| 4.7 | DSI 12. Aprobación del diseño del sistema de información | 17 |

Arinbide

| | | |
|----------|-------------------------------------------------------------------|-----------|
| 5 | Construcción del sistema de información | 19 |
| 5.1 | Esquema general..... | 19 |
| 5.2 | CSI 1. Preparación el entorno de desarrollo..... | 20 |
| 5.3 | CSI 2. Generación del código de los componentes | 20 |
| 5.4 | CSI 6. Elaboración del manual de usuario | 20 |
| 5.5 | CSI 9. Aprobación del sistema de información | 20 |
| 6 | Implantación y aceptación del sistema de información | 21 |
| 6.1 | Esquema general..... | 21 |
| 6.2 | IAS 1. Entrega del producto a servicio | 21 |
| 6.3 | IAS 3. Traspaso al entorno de pruebas..... | 22 |
| 6.4 | IAS 9. Presentación y aprobación del sistema..... | 22 |
| 6.5 | IAS 10. Paso a producción | 22 |
| 7 | Entregables | 23 |
| 7.1 | Relación de entregables..... | 23 |
| 7.2 | CRU – Catálogo de requisitos de usuario | 24 |
| 7.3 | ERS – Especificación de requisitos del sistema..... | 25 |
| 7.4 | EDS – Especificación de diseño del sistema | 26 |

Ventajas / Inconvenientes

| Ventajas | Inconvenientes |
|--------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| ✓ Una estructura sencilla gracias a unas fases de proyecto claramente diferenciadas. | ✗ Por norma general, los proyectos más complejos o de varios niveles no permiten su división en fases de proyecto claramente diferenciadas. |
| ✓ Buena documentación del proceso de desarrollo a través de unos hitos bien definidos. | ✗ Poco margen para realizar ajustes a lo largo del proyecto debido a un cambio en las exigencias. |
| ✓ Los costes y la carga de trabajo se pueden estimar al comenzar el proyecto. | ✗ El usuario final no se integra en el proceso de producción hasta que no termina la programación. |
| ✓ Aquellos proyectos que se estructuran en base al modelo en cascada se pueden representar cronológicamente de forma sencilla. | ✗ En ocasiones, los fallos solo se detectan una vez finalizado el proceso de desarrollo. |

¿Cómo solucionar los inconvenientes?

- ❑ **Evolucionar** el modelo en cascada
 - ❑ Elaboración de un prototipo
 - ❑ Requerir aprobación por parte del cliente tras la fase de diseño.
 - ❑ Incluir al usuario en las pruebas
- ❑ Optar por un **enfoque completamente diferente**
 - ❑ **Metodologías ágiles**

Scrum

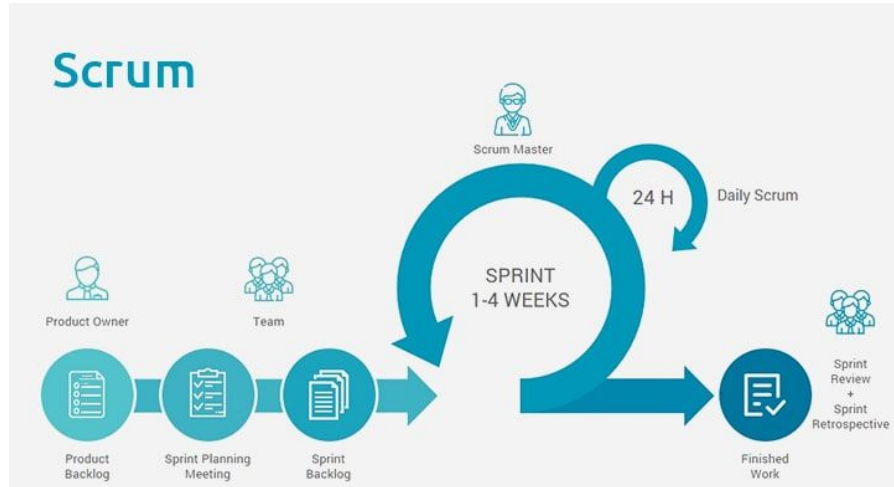


Image source: <https://www.peerbits.com/blog/agile-software-development.html>

Scrum

- ❑ Encaja perfectamente en **entornos complejos, procesos con cambios rápidos o requerimientos emergentes**.
 - ❑ Por ejemplo, si el cliente puede no tiene claro cuál es exactamente el objetivo final en el momento en el que decide comenzar un proyecto.
- ❑ **Proceso**
 - ❑ Se divide el proyecto en etapas breves: cada **iteración (sprint)**, tiene una duración de entre 2 y 4 semanas, obteniendo como resultado una versión del software con nuevas funcionalidades listas para ser usadas.
 - ❑ Al inicio de cada sprint se realiza una reunión para establecer los **objetivos del sprint**.
 - ❑ Se realiza una reunión diaria para que cada persona comente **que hizo el día anterior, que hará hoy y si hay impedimentos**
 - ❑ Al final de cada sprint se hace una reunión llamada retrospectiva del sprint o sprint retrospective para analizar el cumplimiento de objetivos, dificultades, etc. → kaizen

Kanban



Kanban

- ❑ Proviene de la palabra japonesa que significa “tarjetas visuales”, donde Kan es “visual”, y Ban corresponde a “tarjeta”.
- ❑ Uno de sus puntos fuertes es que es una técnica de gestión de las tareas **muy visual**.
- ❑ Se crea **una nueva tarjeta por cada nuevo requisito** del cliente. Se añadirá al flujo de entrada y seguirá su flujo normal hasta la salida.
- ❑ Las tareas se encuentran **ordenadas por prioridad**.
 - ❑ Es un método flexible, se prioriza según las necesidades del momento.
- ❑ Se van arrastrando las nuevas tareas por el panel hasta que lleguen a su estado final.
- ❑ No hay unas fases del ciclo de producción establecidas. Se definen en función de cada caso, o se establecerá un modelo genérico para todos los proyectos de la empresa..

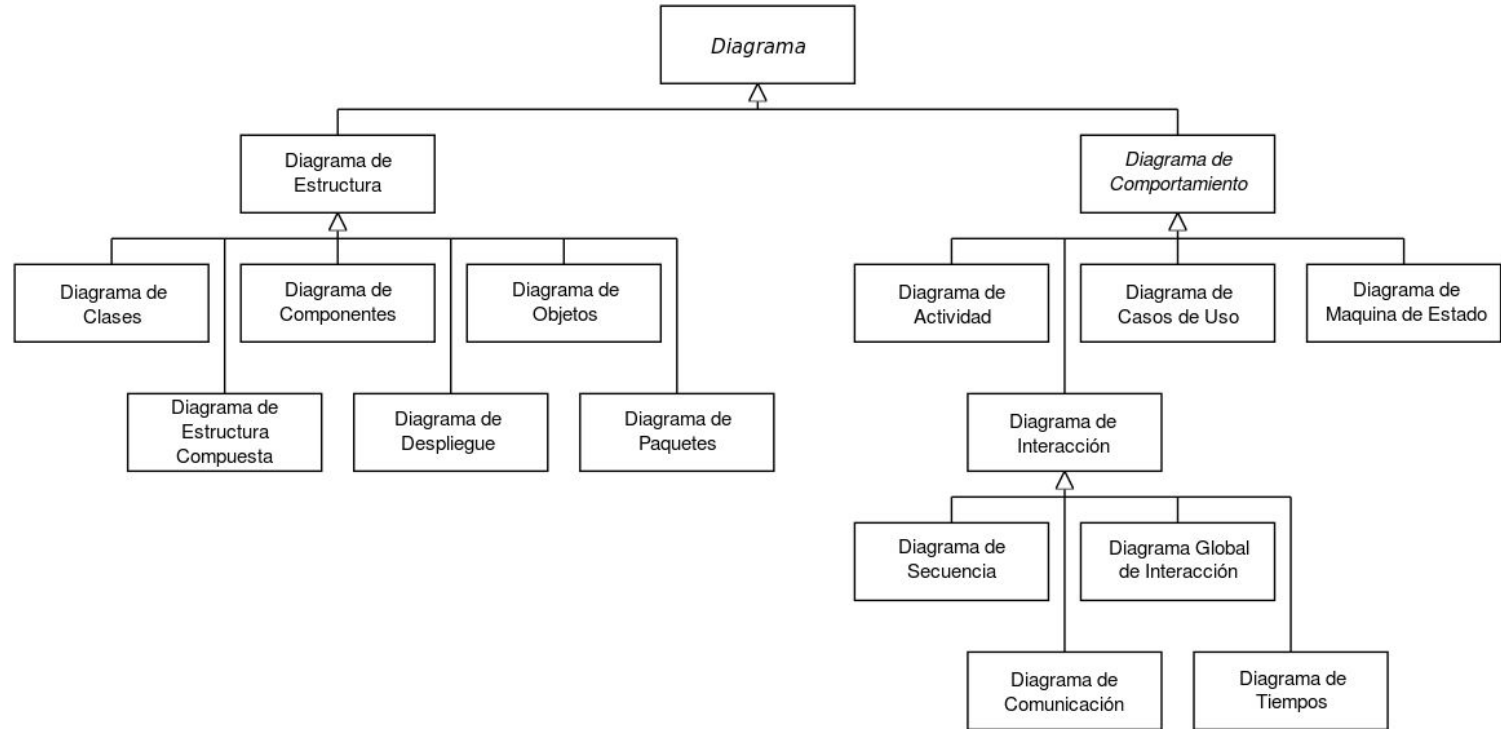
Scrum vs Kanban

- ❑ En Scrum se organiza el trabajo a realizar en iteraciones de un tiempo determinado. En Kanban sin embargo no se planifican iteraciones, es un flujo continuo de trabajo.
- ❑ En Kanban se fija un número máximo de tareas en progreso, mientras en Scrum la capacidad se fija por cada iteración (sprint).
- ❑ En Scrum se estima cuánto tiempo o esfuerzo cuesta completar cada tarea. En Kanban no se hace esta estimación.
- ❑ En Scrum los tableros se “reinician” al final de cada Sprint. En Kanban, al tener un flujo de entrada-salida, las tarjetas van pasando por cada uno de los estados y al llegar al estado final se archivan.
- ❑ Por lo general, Scrum está enfocado para los proyectos de SW y Kanban es más general.

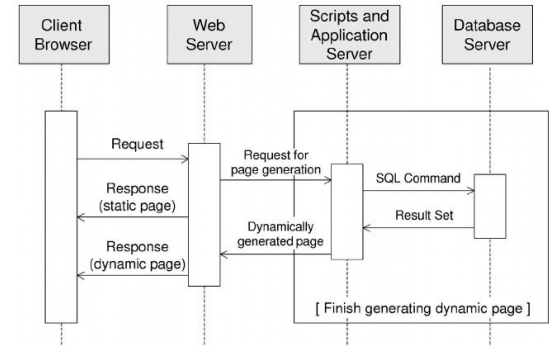
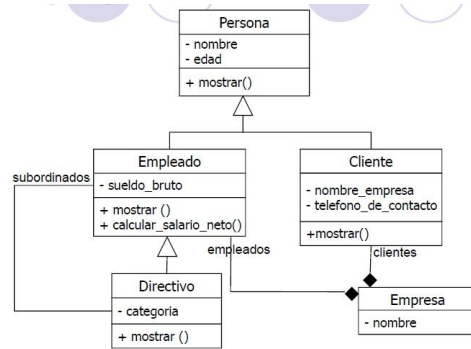
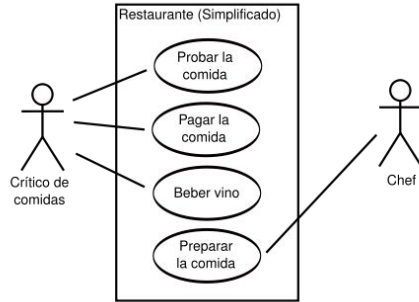
UML

- ❑ El Lenguaje Unificado de Modelado (UML) fue creado para forjar un lenguaje de modelado visual común.
- ❑ Describen los **límites, la estructura y el comportamiento del sistema y los objetos que contiene.**
- ❑ Está basado en el paradigma de programación orientado a objetos.
- ❑ No es un lenguaje de programación. Puede utilizarse para generar código en distintos lenguajes de programación.

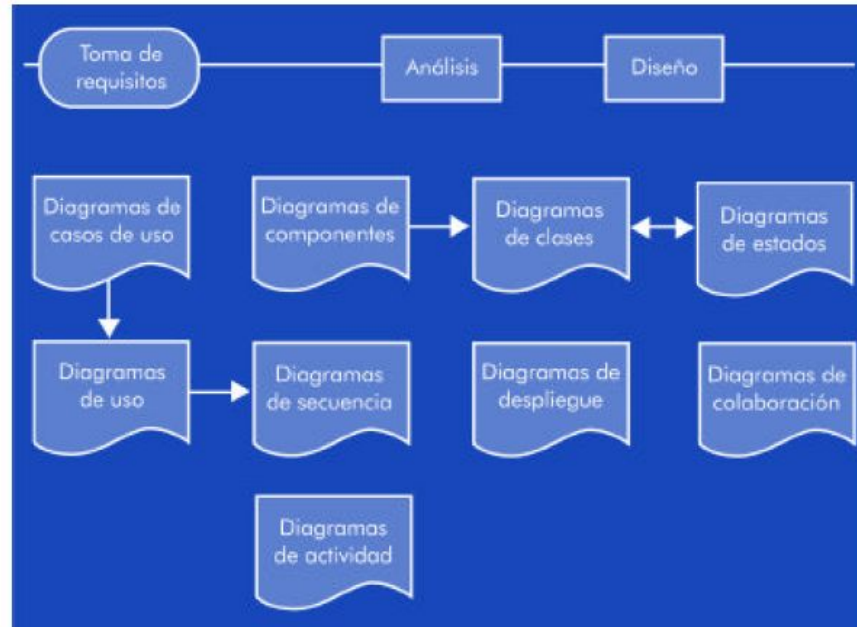
UML



Ejemplos de diagramas UML




Relación UML - Fases de desarrollo



Fases

1. Análisis de requisitos
2. Diseño.
3. Desarrollo.
4. Testing.
5. Implementación o verificación del programa.
6. Mantenimiento.

Fases

1. **Análisis de requisitos** 
2. Diseño.
3. Desarrollo.
4. Testing.
5. Implementación o verificación del programa.
6. Mantenimiento.

Estudio de la **necesidad** tecnológica que
tiene un **cliente** y que **especifica las**
características operacionales que tendrá el
software a desarrollar.

Análisis de requerimientos

- ❑ Se trata de analizar las **necesidades** de los usuarios y determinar los **objetivos** que la aplicación debe cumplir.
- ❑ **Fases:**
 - ❑ **Recogida de requerimientos** (Se realiza a través de técnicas como las entrevistas, reuniones, observación, investigación,...).
 - ❑ **Analizar** los requerimientos, de forma que sean claros y consistentes.
 - ❑ **Documentar** los requerimientos: organizarlos y escribirlos de forma clara y entendible.
 - ❑ Asegurar de que **todo el mundo entiende** los requerimientos.



How the customer explained it



How the project leader understood it



How the engineer designed it



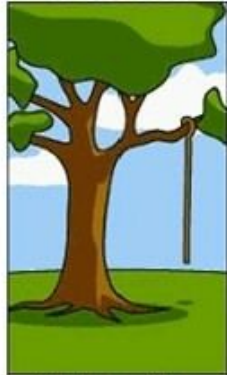
How the programmer wrote it



How the sales executive described it



How the project was documented



What operations installed



How the customer was billed



How the helpdesk supported it



What the customer really needed

Toma de requerimientos

- ❑ El usuario no sabe detallar de forma concreta lo que quiere.
- ❑ El usuario no conoce las posibilidades de una aplicación y de la tecnología.
- ❑ Un error en la toma de requerimientos es el más difícil de corregir a posteriori.

Toma de requerimientos

- ❑ ¿Qué personas toman parte? ¿Quiénes utilizarán la aplicación? → **Identificar roles**
- ❑ ¿Qué hace cada usuario? ¿Cuáles son sus tareas/necesidades principales? → **Identificar tareas**
- ❑ ¿Para qué realizan esas tareas? ¿Cuáles son sus problemas o dificultades?
- ❑ ¿Cómo realizan sus tareas? ¿Cuáles serían los pasos?
- ❑ ¿Qué información necesitan en cada momento para realizar sus tareas?
- ❑ ¿Hay alguna funcionalidad que podría facilitar su tarea?
- ❑ Otras preguntas:
 - ❑ ¿Desde dónde (PC/Tablet/Móvil) accederá cada usuario a la aplicación? ¿En qué entorno accederá a la aplicación?
 - ❑ ¿Es necesario implementar algún mecanismo de seguridad?
 - ❑ ¿Cuánta información se maneja al día?

Toma de requerimientos

| | |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Rol | Administrativo de secretaría en Egibide |
| Tarea 1 | <ul style="list-style-type: none">• <u>Nombre</u>: Alta de estudiantes• <u>Descripción</u>: El estudiante entrega en papel sus datos personales y desde secretaría tienen que realizar el alta.• <u>Necesidad</u>: Introducir los datos de forma Ágil• <u>Datos</u>: introduce DNI, nombre y apellidos, teléfono y dirección. Necesita disponer del listado de ciclos y grupos para asignarle uno al estudiante. |
| Tarea 2 | <ul style="list-style-type: none">• <u>Nombre</u>: Consulta de estudiantes• <u>Descripción</u>: Un profesor acude a secretaría a solicitar datos de un estudiante y el administrativo debe darle todos los detalles.• <u>Necesidad</u>: Acceder a los datos de forma rápida mediante la búsqueda por filtros. Tendrá que poder ver todos los datos en una única pantalla.• <u>Datos</u>: Normalmente busca por DNI o nombre, aunque en ocasiones también por ciclos. |

A top-down view of a wooden workbench covered with an assortment of hand tools. The tools are scattered across the frame, including several pairs of pliers with different handle colors (orange, black, green), screwdrivers with red and yellow handles, adjustable wrenches, and three yellow tape measures. A yellow spirit level is positioned horizontally in the center-left. Other visible items include a metal gear, a small metal ring, a metal rod, and a metal plate with a ruler scale. The word "HERRAMIENTAS" is overlaid in white capital letters in the center of the image.

HERRAMIENTAS

Historias de usuario

- ❑ Una historia de usuario (user story) es **una representación de un requisito** escrito en una o dos frases **utilizando el lenguaje común del usuario**.
- ❑ Son una forma rápida de administrar los requisitos de los usuarios **sin tener que elaborar gran cantidad de documentos formales**.

Como <tipo de usuario> quiero/necesito <objetivo> para poder <beneficio>

As a role, persona needs to perform
some action which allows her to reach
the goal she is trying to accomplish.

User story definition

Front of Card

173

As a student I want to purchase
a parking pass so that I can
drive to school

Priority: ~~High~~ Should
Estimate: 4

Back of Card

Confirmations:

~~The student must pay the correct amount~~
One pass for one month is issued at a time
The student will not receive a pass if the payment
isn't sufficient

The person buying the pass must be a currently
enrolled student.

The student may only buy one pass per month.

Ejemplos

- ❑ Como visitante, quiero ver los productos para poder comprar.
- ❑ Como administrador, quiero poder publicar productos en un orden específico para que los usuarios vean primero lo que me interesa más vender.
- ❑ Como usuario registrado, quiero poder actualizar mi información de perfil para mantener mi información al día.
- ❑ Como administrador, quiero poder visualizar estadísticas de productos vendidos para poder identificar la información más relevante.

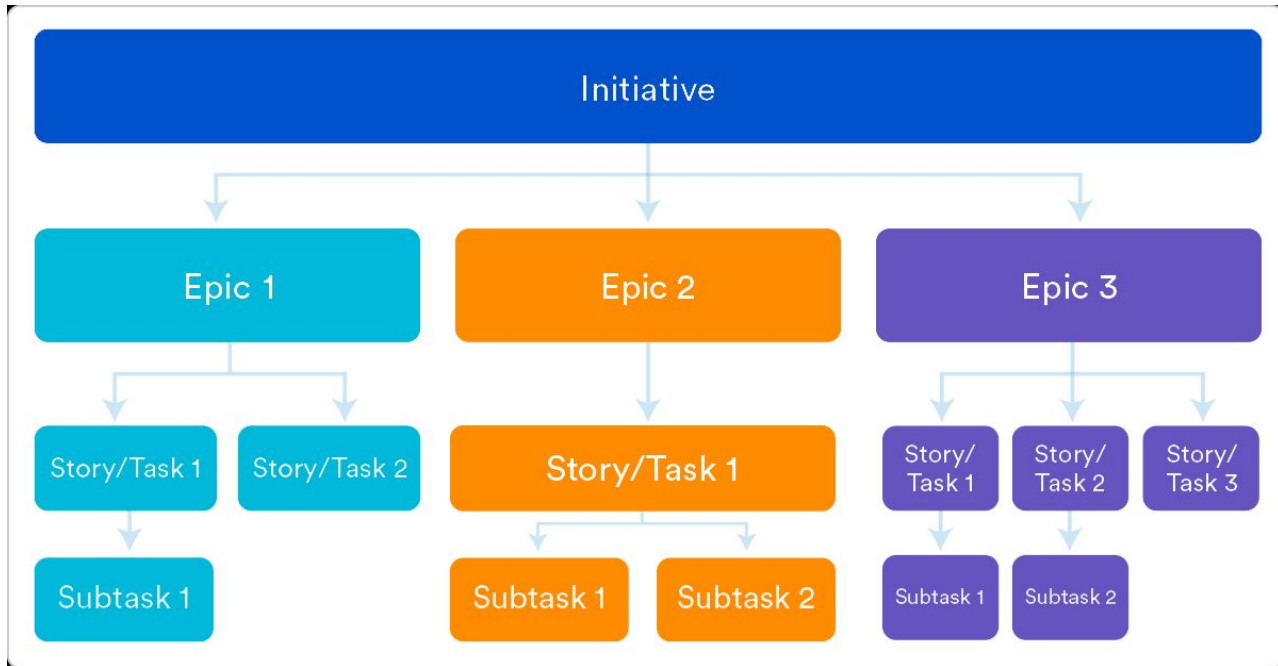
Es importante la última parte de la historia, el “para qué”, ya que nos dará pistas de cómo desarrollar esa funcionalidad.

Características

- ❑ **Independientes:** pueden entregarse en cualquier orden
- ❑ **Negociables:** La historia en sí misma no es lo suficientemente explícita como para considerarse un contrato, la discusión con los usuarios debe permitir esclarecer su alcance y éste debe dejarse explícito bajo la forma de pruebas de validación.
- ❑ **Valoradas por los clientes o usuarios:** deben aportar un valor claro a los usuarios.
- ❑ **Estimables:** Un resultado de la discusión de una historia de usuario es la estimación del tiempo que tomará completarla. Esto permite estimar el tiempo total del proyecto.
- ❑ **Pequeñas:** Las historias muy largas son difíciles de estimar
- ❑ **Verificables:** Las historias de usuario cubren requerimientos funcionales, por lo que generalmente son verificables. Cuando sea posible, la verificación debe automatizarse, de manera que pueda ser verificada en cada entrega del proyecto.

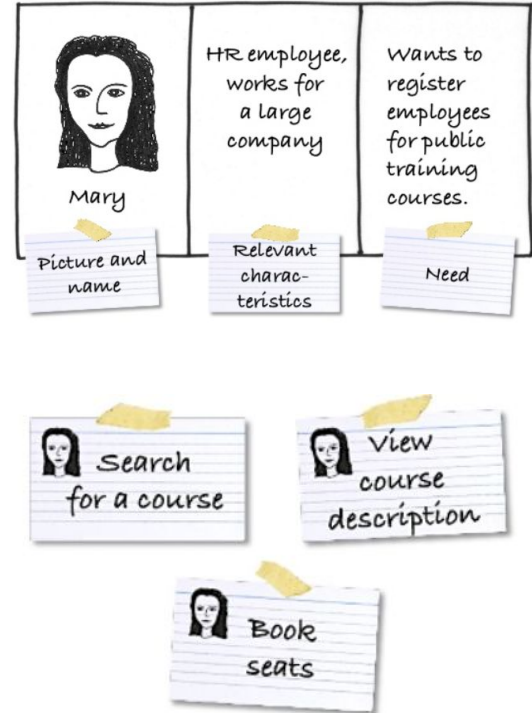
Épica

- ❑ Una épica es **una historia de usuario que abarca una amplia funcionalidad** y que **puede dividirse** a su vez en historias de usuario más concretas.
- ❑ Ayudan a mantener de forma fácil el listado de próximas funcionalidades a añadir.
- ❑ Ejemplo:
 - ❑ **Épica:** como responsable de ventas, quiero revisar los datos históricos de las ventas, para poder identificar las regiones geográficas y productos de mejor desempeño.
 - ❑ **Historia de usuario 1:** como responsable, quiero seleccionar el período de tiempo en el cual realizaré la revisión de las ventas.
 - ❑ **Historia de usuario 2:** como responsable, puedo clasificar la información de ventas por región geográfica y productos.



Proceso

1. Define los **tipos de usuarios** que tendrá la aplicación
2. Haz un listado de **ÉPICAS**
3. Desglosa cada épica en varias **historias de usuario**
 - Añade 3-5 condiciones que cada historia de usuario tiene que cumplir. Esto ayudará a testear que la solución implementada cumple con las funcionalidades descritas.
 - Si necesitas añadir mayor detalle, desglosa las historias en más historias.



Organizar las User stories

| ID | Theme | As a/an | I want to... | so that... | Notes | Priority | Status |
|-----|----------------|---------------|----------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|--------|
| 2 | Game | moderator | create a new game by entering a name and an optional description | I can start inviting estimators | If games cannot be saved and returned to, the description is unnecessary | Required | done |
| 2 | Game | moderator | invite estimators by giving them a url where they can access the game | we can start the game | The url should be formatted so that it's easy to give it by phone. | | done |
| 5 | Game | estimator | join a game by entering my name on the page I received the url for | I can participate | | | done |
| 6 | Game | moderator | start a round by entering an item in a single multi-line text field | we can estimate it | | | done |
| 8 | Game | estimator | see the item we're estimating | I know what I'm giving an estimate for | | | done |
| 40 | Game-- | participant-- | always have the cards in the same order across multiple draws-- | it's easy to compare estimates-- | - | Replaced with A08 because I didn't want the story to talk about "the same order" as that might be a UI implementation detail | todo |
| 35 | Non-functional | user | have the application respond quickly to my actions | I don't get bored | | | done |
| 36 | Non-functional | user | have nice error pages when something goes wrong | I can trust the system and it's developers | | | done |
| A11 | Non-functional | Researcher | results to be stored in a non-identifiable way | I can study the data to see things like whether estimates converged around the first opinion given by "estimator A" for example | No names or story text should be stored but we should store each card of each hand, know who played it, and know the final accepted estimate | | |
| A05 | Game | moderator | edit an item in the list of items to be estimated | so that I can make it better reflect the team's understanding of the item | | | |
| 22 | Archive | moderator | export a transcript of a game as a CSV file | I can further process the stories and estimates | Exported file should be directly importable back into the system. | | done |

Source: <https://www.mountaingoatsoftware.com/blog/a-sample-format-for-a-spreadsheet-based-product-backlog>

Casos de uso

- ❑ Un caso de uso es la **descripción de una acción o actividad**.
- ❑ Un **diagrama de caso de uso** es una descripción de las actividades que deberá realizar alguien o algo para llevar a cabo algún proceso.
- ❑ Un caso de uso debe incluir al menos la siguiente información:
 - ❑ ID
 - ❑ Nombre
 - ❑ Actores
 - ❑ Precondición
 - ❑ Postcondición
 - ❑ Descripción del escenario (pasos a dar por el usuario).
 - ❑ Puede incluir escenarios alternativos o excepcionales.

| entificación del caso de uso | |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ID | CU-009 |
| Nombre | Consultar Total Unidades Vendidas |
| Actores | Podrán acceder al sistema todos los actores autorizados que participan en la conexión del sistema. Gerente Jefe Supervisor Vendedor TBCONTROL Cubo Comercial Disco (Medio donde se guardara archivo exportado) Impresora (Medio donde se imprimirá) |
| Descripción breve | Este caso de uso describe como los actores pueden utilizar la opción Total Unidades Vendidas, obteniendo un despliegue en pantalla del resultado en un formato definido. |
| Precondiciones | El actor debe estar conectado a la aplicación y haber seleccionado la opción Total Unidades Vendidas. |
| Poscondiciones | Resultado de la consulta es mostrado en la pantalla. Se registra en tabla TBCONTROL la consulta realizada por el actor. |
| Flujo de eventos | Una vez seleccionada la opción Total Unidades Vendidas se leerá el cubo comercial. |
| Flujo básico | Actor selecciona opción Total Unidades Vendidas desde menú. Se lee cubo comercial. Se despliega resultado de la consulta en pantalla. El resultado de la consulta será mostrado en pantalla. Una vez mostrado el resultado en la pantalla el actor puede: -Cambiar formato del resultado a formato rejilla (DataGridView) -Cambiar formato del resultado a Grafico -Guardar resultado a algún medio de almacenamiento -Imprimir resultado |
| Frecuencia de uso | Caso de uso de frecuencia diaria. |
| Incluye (Include) | Caso de uso CU-015 Registrar en TBCONTROL. |
| Extiende (Extend) | Casos de uso CU-013 Exportar a Excel y CU-014 Imprimir |

Diagramas de casos de uso

- ❑ **Actores:** representan los roles que juegan los usuarios u otros sistemas en el sistema del problema. Identificar a los actores de un caso de uso pasa por averiguar quién está involucrado en cada requisito concreto.
- ❑ **Caso de uso:** son las acciones que pueden tener lugar en el sistema que queremos modelar. Para identificarlas, puede ser útil preguntarse cuáles son las tareas y responsabilidades de cada actor, si habrá actores que recibirán información del sistema, etc.
- ❑ **Relaciones:** indican actividad o flujo de información

Diagrama de casos de uso

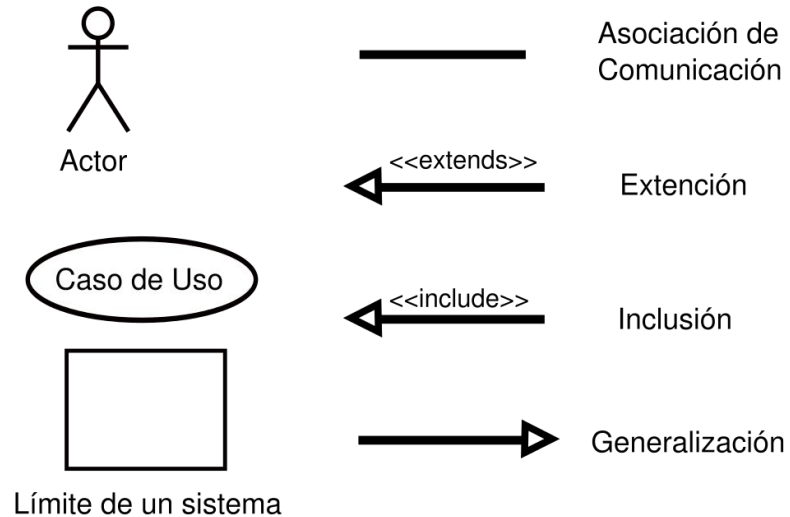
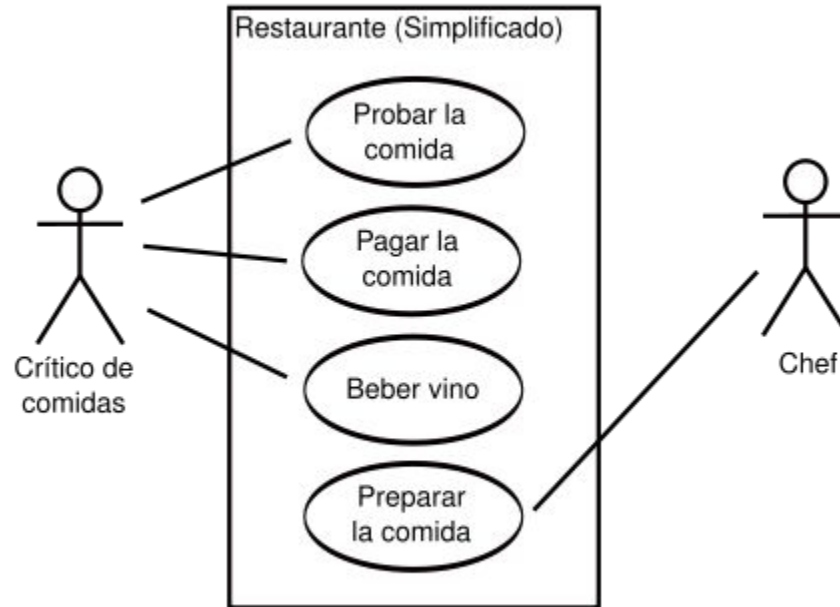
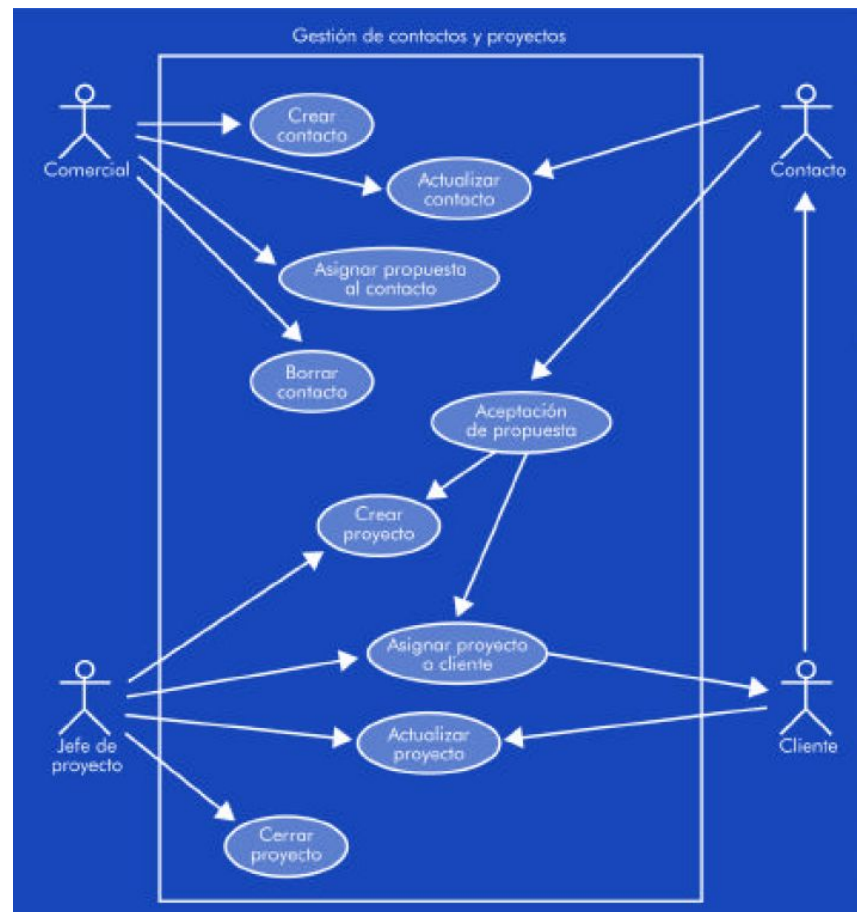
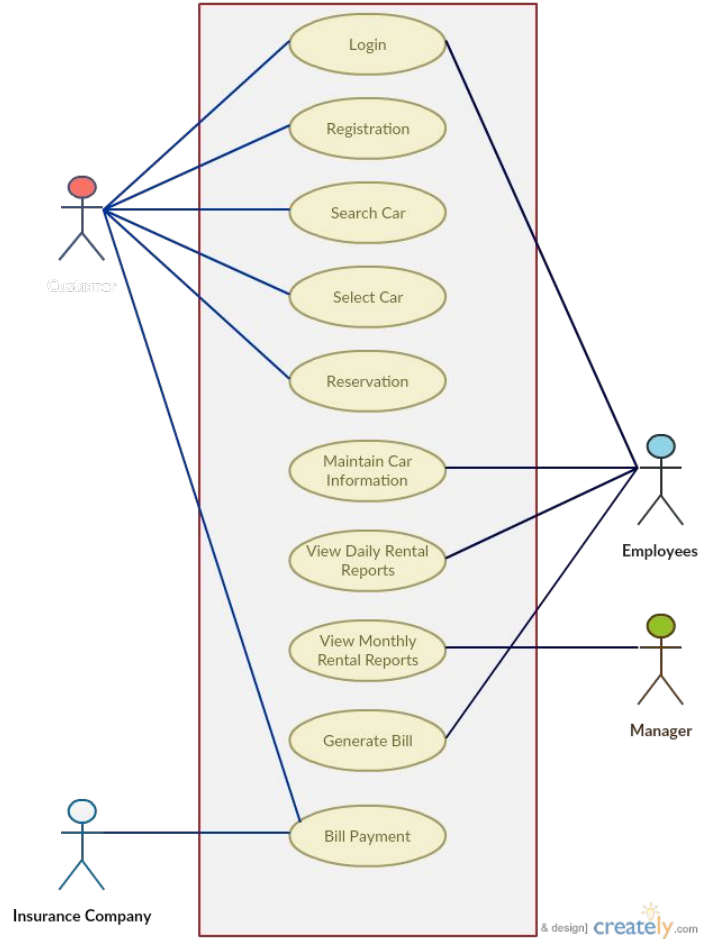


Diagrama de casos de uso






Car Rental System



Fases

1. Análisis de requisitos
2. **Diseño** 
3. Desarrollo
4. Testing
5. Implementación o verificación del programa
6. Mantenimiento

Diseño de la solución

- ❑ Diseño a alto nivel de la solución propuesta que **cubrirá las necesidades detectadas y los requerimientos establecidos.**
- ❑ Niveles:
 - ❑ Diseño de las interfaces (Prototipado)
 - ❑ Diagrama de flujo de navegación
 - ❑ Diseño de la estructura de datos
 - ❑ Diseño de la arquitectura (tecnologías, frameworks, patrones)
 - ❑ ...

Diagramas de actividad

- ❑ Representan los flujos de trabajo paso a paso entre los usuarios y el sistema.
- ❑ Utilizan símbolos con significados definidos que representan los pasos del algoritmo, y representan el flujo de ejecución.

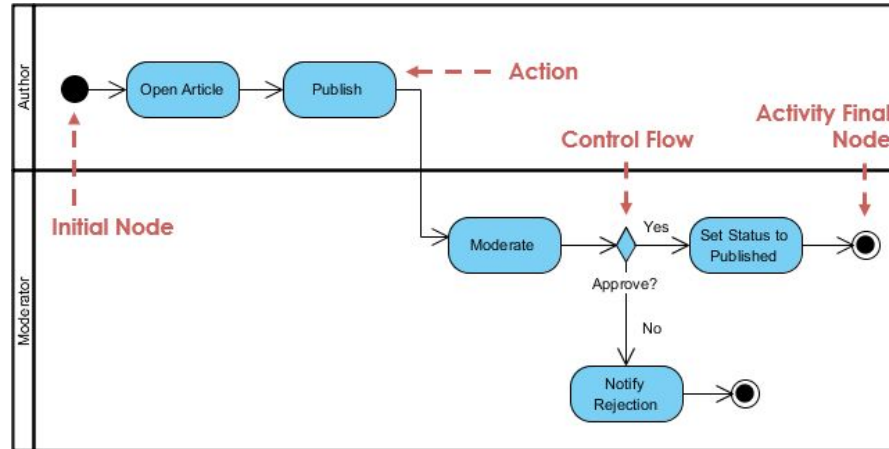
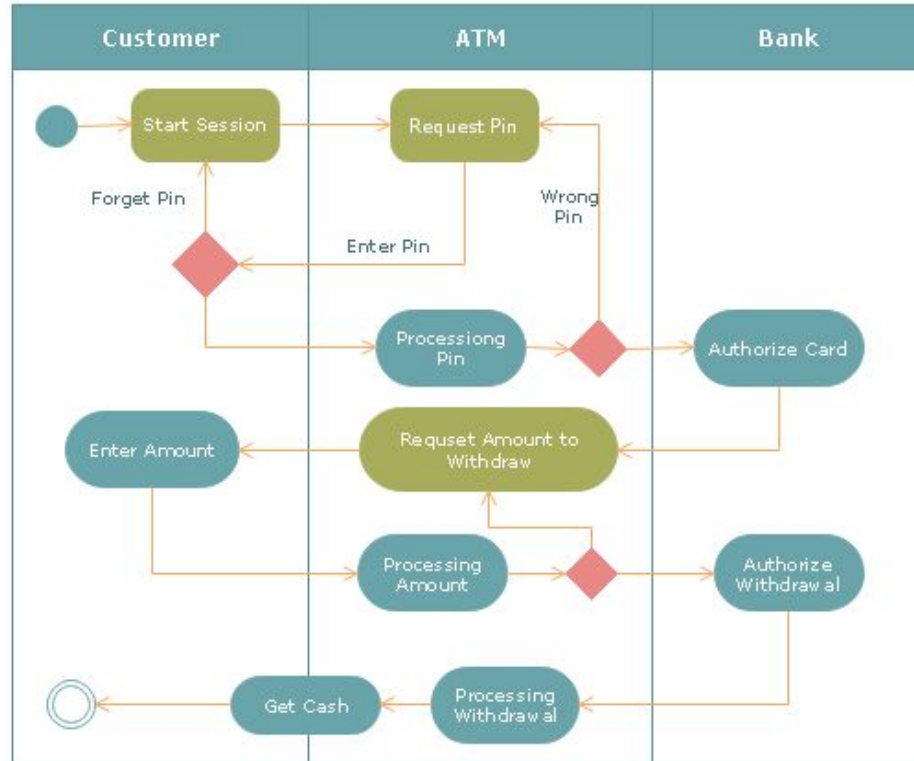


Image source: <https://www.visual-paradigm.com/>

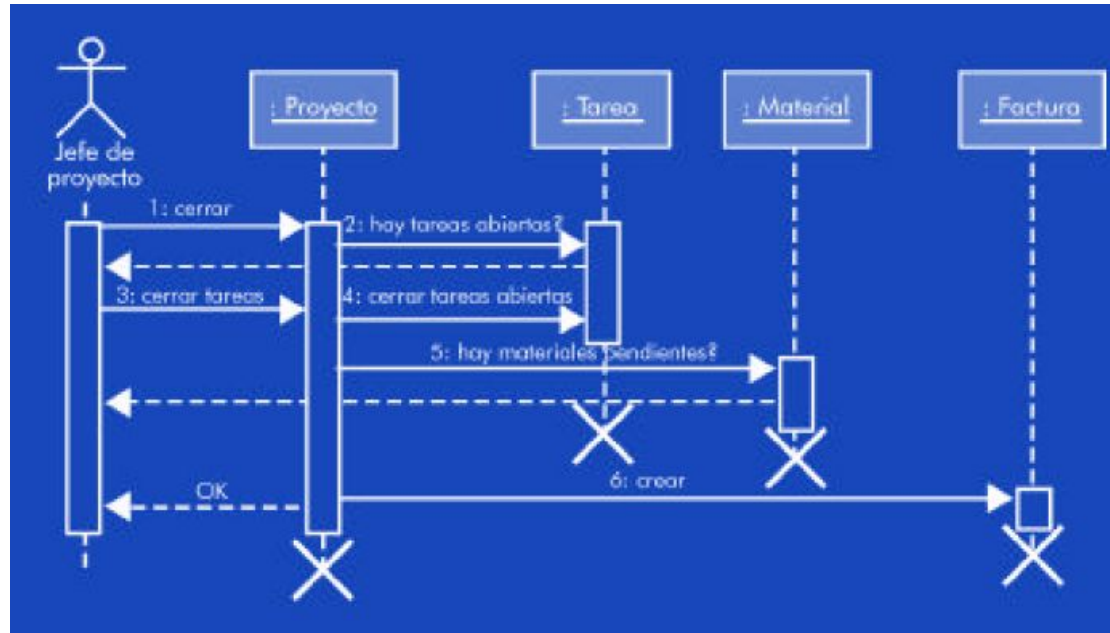
ATM Withdrawal Activity Diagram



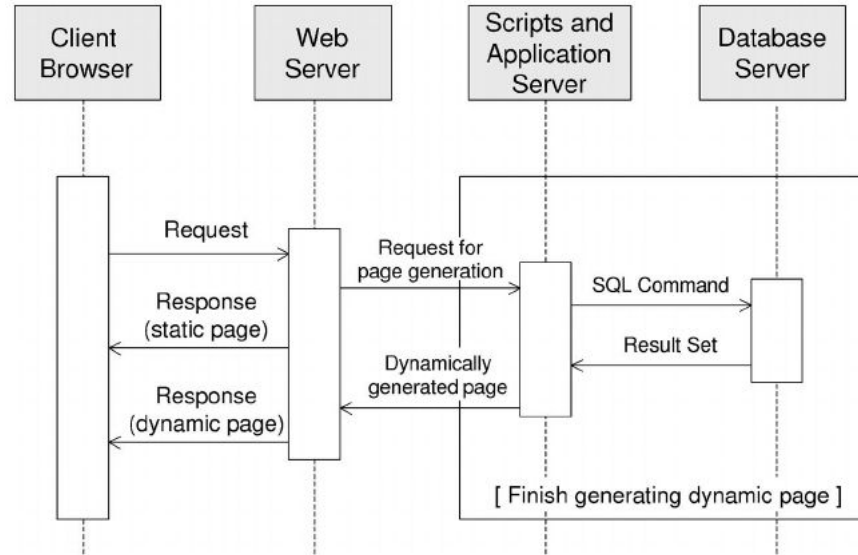
Diagramas de secuencia

- ❑ Modelan el flujo de la lógica dentro del sistema de forma visual.
- ❑ Pueden usarse tanto en análisis como en diseño, proporcionando una buena base para identificar el comportamiento del sistema.
- ❑ La secuencia puede expresar tanto un caso de uso completo como quizá un caso concreto.
- ❑ Componentes
 - ❑ Objeto
 - ❑ Actor
 - ❑ Vida del objeto
 - ❑ Activación
 - ❑ Mensaje

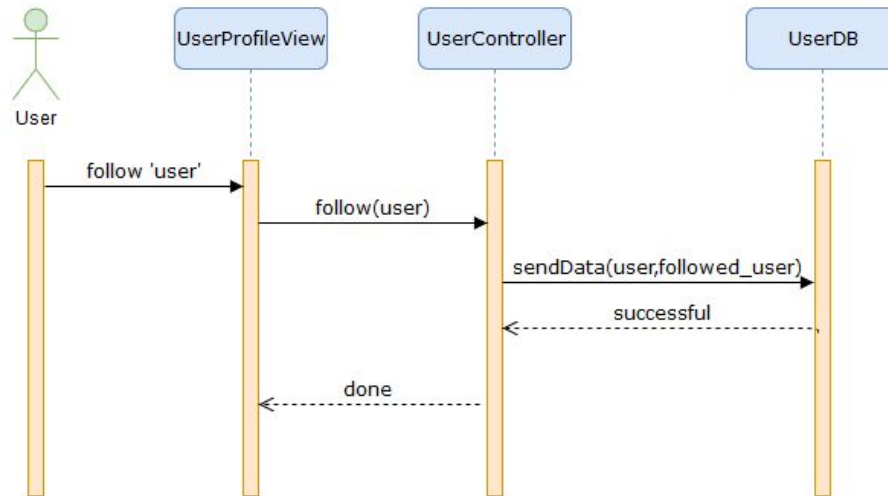
Diagramas de secuencia



Diagramas de secuencia



Diagramas de secuencia



Diagramas de clases

- ❑ Son los más utilizados en el modelado de sistemas orientados a objetos.
- ❑ Describe la estructura de un sistema mostrando:
 - ❑ Las clases del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos
- ❑ Permiten construir sistemas ejecutables, aplicando ingeniería directa e inversa.

| Persona |
|--------------------------------------------------------------------------------------------------------------------------|
| nombre:string edad:integer |
| edad ():integer cambieNombre(nuevoNombre:string) camineHacia (punto:Coordenada) print () print (x,y:integer) |

| Reglas de visibilidad |
|------------------------------------------------------------------------------------|
| + Atributo público : int # Atributo protegido : int - Atributo privado : int |
| + "Operación pública" # "Operación protegida" - "Operación privada" |

Diagramas de clases

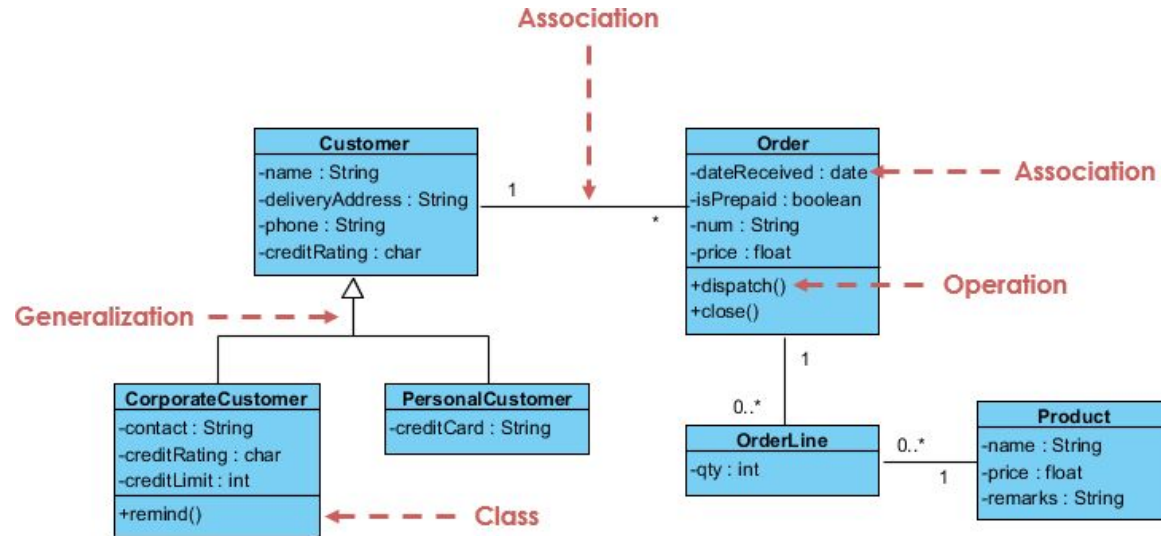
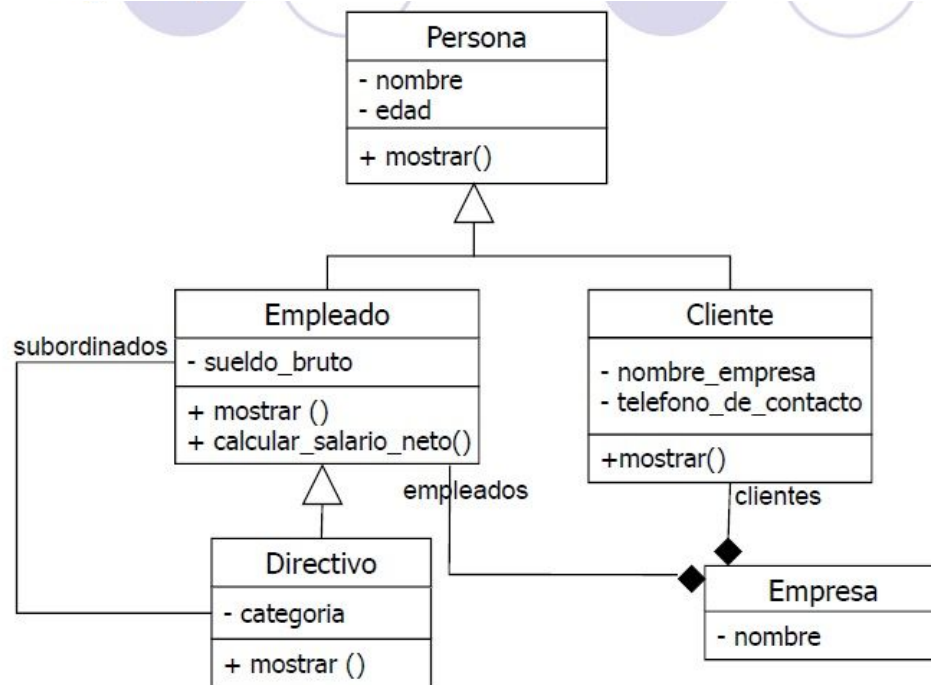


Image source: <https://www.visual-paradigm.com/>

Diagramas de clases



Prototipado

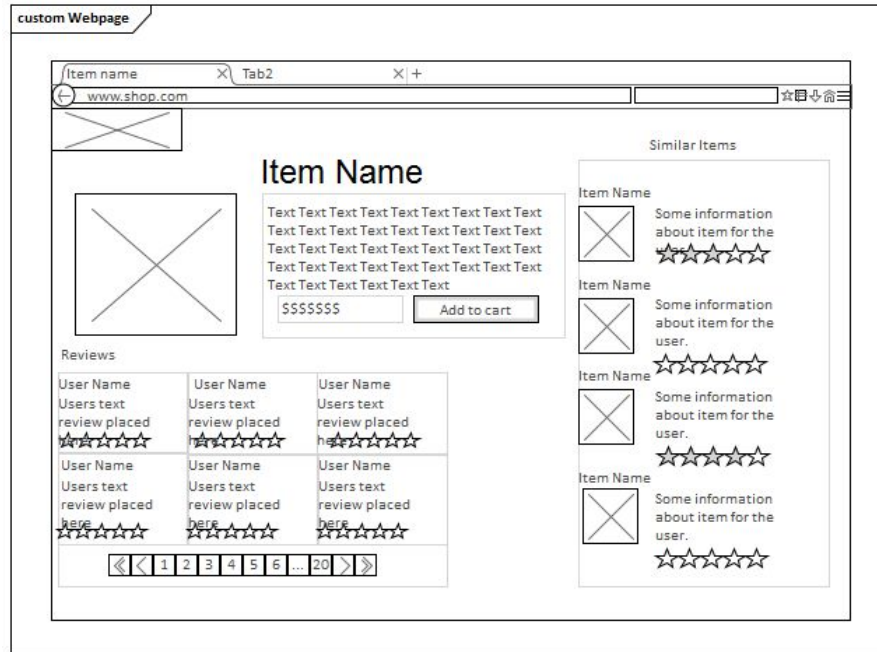
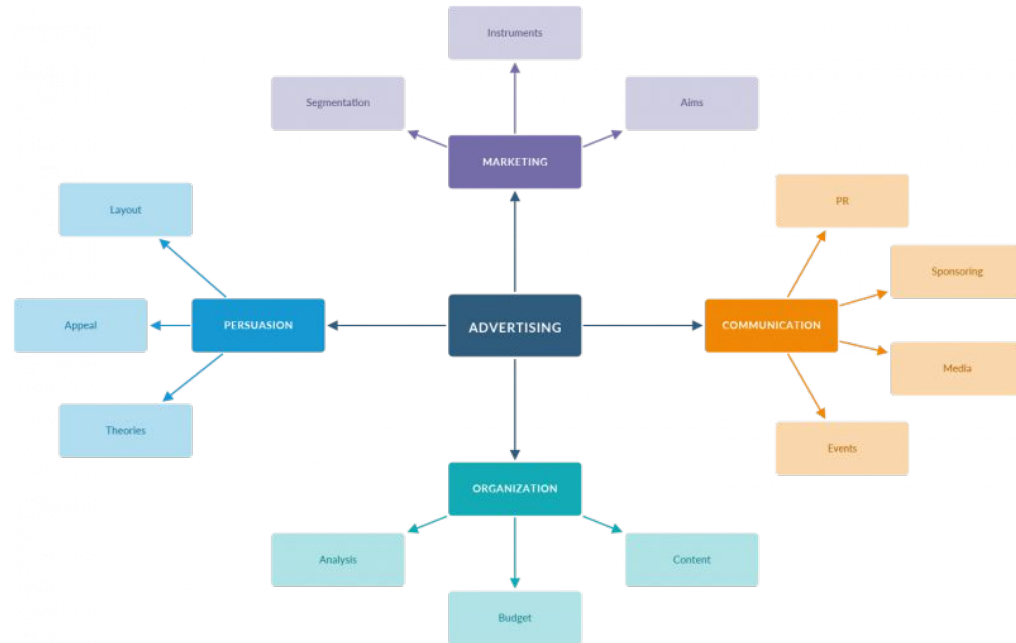
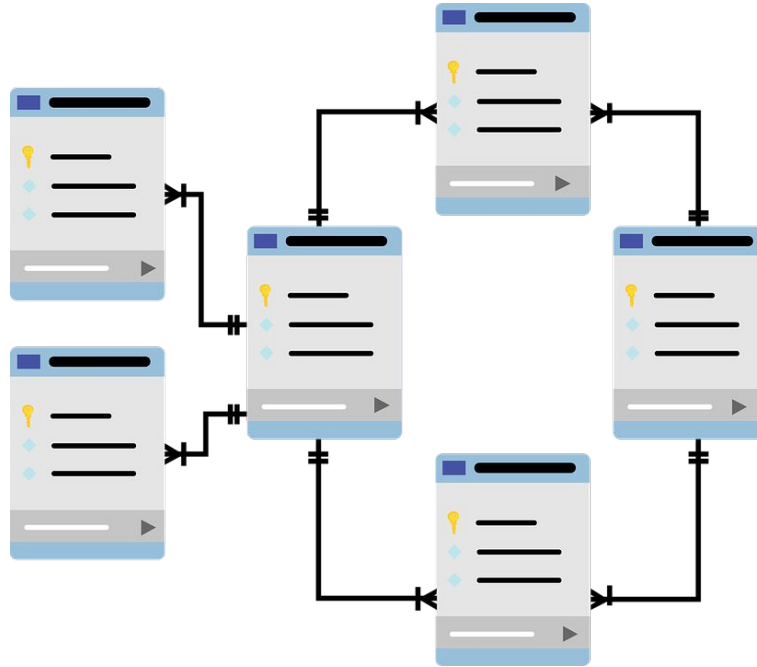


Diagrama de navegación

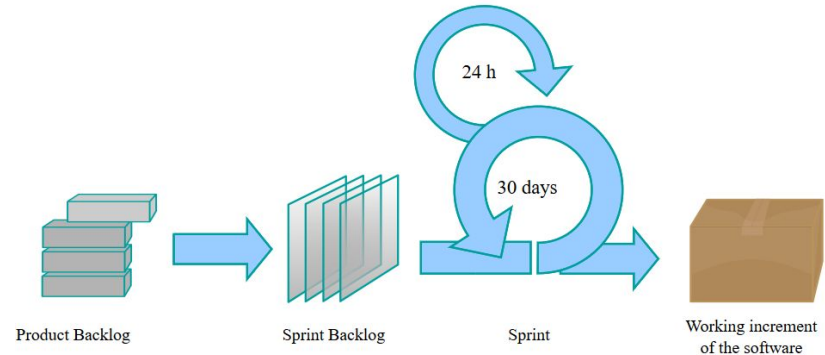
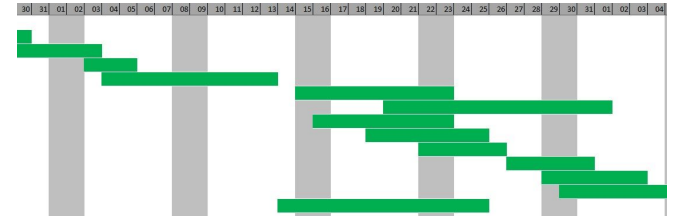


Arquitectura de datos



Bonus: herramientas de planificación

- ❑ Gestión de proyectos integral:
 - ❑ Clickup: <https://clickup.com/>
 - ❑ Bitrix24: <https://www.bitrix24.com/>
 - ❑ Zenkit: <https://zenkit.com/en/features/gantt-view/>
- ❑ Diagramas Gantt
 - ❑ GanttProject: <https://www.ganttproject.biz/>
 - ❑ TeamGantt: <https://www.teamgantt.com>
 - ❑ TeamWeek: <https://teamweek.com/>
- ❑ SCRUM
 - ❑ Trello: <https://trello.com/>
 - ❑ Kanban Tool: <https://kanbantool.com/es/>



Sources

- ❏ <https://www.atlassian.com>
- ❏ <https://www.mountaingoatsoftware.com/>
- ❏ <https://www.visual-paradigm.com/>

“Software quality begins with the quality of the requirements.”

Pearl Zhu.