

La paz, 30 de noviembre de 2023

Metodología de desarrollo XP

Asignatura : Análisis de sistemas I

Docente : Lic. Alfredo Huanca Ticona

Estudiante : Fabio Camacho Encinas

Carrera : Ingeniería de sistemas

Introducción

La industria del desarrollo de software al igual que la mayoría de las industrias, tiene como requisito principal una cadena de producción organizada y eficiente, de manera que la labor intelectual de la creación de software pueda representar utilidades y beneficio económico para los emprendimientos de desarrollo. el mercado desarrollo de software exige a los equipos de desarrollo la aplicación de métodos efectivos para distribuir productos de buena calidad de manera rápida y organizada.

Para este propósito, las metodologías de desarrollo han sido fundamentales desde muy tempranas épocas del inicio del campo de ingeniería de software, sistemas y demás. Siendo la evolución rápida y aparición constante de nuevos enfoques una característica principal en el campo del desarrollo de software, los tipos de metodologías también han ido evolucionando, desde metodologías clásicas como cascada, metodología de prototipos, espiral, a metodologías mas recientes y agiles como XP, Scrum, Kanban y demás.

Este documento se enfoca en describir ciertos aspectos de la metodología XP (Extreme Programming), una metodología ágil de desarrollo que surgio a finales de los años 90 como respuesta a la necesidad de los desarrolladores para ajustarse a los requerimientos cambiantes de los clientes.

Marco teórico

Metodología

La metodología es una serie de métodos y técnicas de rigor científico que se aplican sistemáticamente durante un proceso de investigación para alcanzar un resultado teóricamente válido.

La metodología funciona como el soporte conceptual que rige la manera en que aplicamos los procedimientos en una investigación.

Fuente: https://www.significados.com/metodologia/

Software

El software es un conjunto de reglas o programas que dan instrucciones a un ordenador para que realice tareas específicas. También se conoce como aplicaciones de software, paquetes de software, herramientas de software y programas de software. El software puede utilizarse para gestionar datos, automatizar procesos y crear aplicaciones o productos informáticos.

Fuente: https://www.arimetrics.com/glosario-digital/software

Metodología de software

La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de software comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que cumplimos el objetivo por el cual fue creado.

Fuente: <https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf>

Antecedentes históricos

Las primeras décadas de desarrollo de software

Las metodologías de Desarrollo de Software (DS.) han experimentado un proceso histórico y evolutivo que inicia en los años 40 con la aparición de las primeras computadoras, entonces no se contaban con parámetros ni estándares, el DS. Era prácticamente empírico y artesanal lo que llevó a que una buena parte de los proyectos fallaran en cubrir las exceptivas de los usuarios, así como en entregas extemporáneas y presupuestos excedidos, sobreviniendo la “crisis del Software” la respuesta para superarla fue la adopción de modelos y metodologías clásicas que progresivamente fueron incorporando estándares, controles y formalidades al DS. En un afán que llegó a ser definido como “triángulo de hierro.” La evolución no se detuvo, con la llegada del Internet surgen proyectos caracterizados por requerimientos cambiantes y tiempos de entregas breves para los que las metodologías existentes no se adaptaban idóneamente, surgen las metodologías ágiles, enfocadas en interacción equipo-usuarios, entregas tempranas y adaptación a los cambios; conviven con los esquemas tradicionales y agrupan a comunidades activas. Este esfuerzo documental busca reseñar de manera integral todo ese cambio evolutivo, por cuanto la mayoría de los trabajos en el área se enfocan en divulgar los métodos ágiles dejando subestimada su procedencia.

Fuente: <https://revistas.uide.edu.ec/index.php/innova/article/view/651>

Aparición de la metodología de desarrollo XP  
  
El origen de la programación extrema (XP) se remonta a la década de 1990 cuando Kent Black intentaba encontrar una mejor manera de realizar el desarrollo de software mientras trabajaba en un proyecto en DaimlerChrysler (DaimlerChrysler fue una fusión entre dos compañías automotrices prominentes: Daimler-Benz AG, una empresa alemana, y Chrysler Corporation). Su nuevo enfoque, que más tarde se llamaría Metodología de Programación Extrema, demostró ser un método exitoso.

Kent Beck, autor del primer libro sobre este ámbito llamado «Extreme Programming Explained: Embrace Change», publicado en 1999.

Fuente: https://www.umsl.edu/~sauterv/analysis/f06Papers/Hutagalung

Manifesto ágil  
  
Al ser Extreme programming una metodología ágil, se rige por los cuatro principios descritos en el manifestó ágil donde se definen los cuatro valores principales por las que se deberían guiar las

metodologías ágiles.

1. Al individuo y sus interacciones más que al proceso y las herramientas.

2. Desarrollar software que funciona más que obtener una documentación exhaustiva.

3. La colaboración con el cliente más que la negociación de un contrato.

4. Responder a los cambios más que seguir una planificación.

Doce principios de las metodologías agiles

Como consecuencia de estos cuatro valores, el Manifiesto ágil también enuncia los doce

principios que caracterizan un proceso ágil diferenciándolo de otro tradicional donde este

enfoque no se había aplicado lo suficiente, siempre se había dejado implícito pero sin hacer

hincapié en ellos.

a) La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de

software que le aporte un valor.

b) Dar la bienvenida a los cambios incluso al final del desarrollo. Los cambios le darán

una ventaja competitiva a nuestro cliente.

c) Hacer entregas frecuentes de software que funcione, desde un par de semanas a

un par de meses, con el menor intervalo de tiempo posible entre entregas.

d) Las personas del negocio y los desarrolladores deben trabajar juntos diariamente a

lo largo de todo el proyecto.

e) Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo

que necesitan y confiar en ellos.

f) El diálogo cara a cara es el método más eficiente y efectivo para comunicar

información dentro de un equipo de desarrollo.

g) El software que funciona es la principal medida del progreso.

h) Los procesos ágiles promueven un desarrollo sostenido. Los promotores, usuarios

y desarrolladores deben poder mantener un ritmo de trabajo constante de forma

indefinida.

i) La atención continua a la calidad técnica y al buen diseño mejoran la agilidad.

j) La simplicidad es esencial. Se ha de saber maximizar el trabajo que no se debe

realizar.

k) Las mejores arquitecturas, requisitos y diseños surgen de los equipos que se han

organizado ellos mismos.

l) En intervalos regulares, el equipo debe reflexionar con respecto a cómo llegar

a ser más efectivo, y ajustar su comportamiento para conseguirlo.

Características fundamentales

Extreme Programming o XP Programming es un marco de desarrollo de software ágil que tiene como objetivo producir un software de mayor calidad para mejorar la eficiencia del equipo de desarrollo. Se trata de una metodología de desarrollo cuyo objetivo es promover la aplicación de prácticas de ingeniería apropiadas para la creación de software.

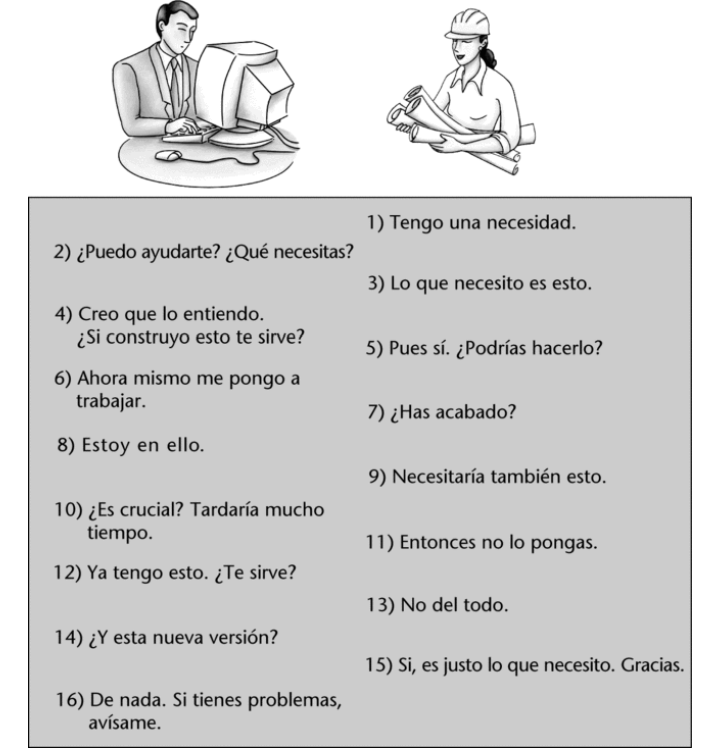
Extreme Programming está diseñada para ofrecer el software que los usuarios necesitan en el momento adecuado. En este sentido, ayuda a los desarrolladores a ajustarse a los requerimientos cambiantes de los clientes.

Este tipo de programación se diferencia de las metodologías tradicionales en que pone más énfasis en la adaptabilidad que en la previsibilidad. El Extreme Programming considera que los cambios de requisitos sobre la marcha son acciones naturales e inevitables en el desarrollo de un proyecto. Creen que ser capaces de adaptarse a los cambios que puedan surgir en cualquier punto del ciclo de vida de un proyecto es una mejor previsión y más realista que intentarlos definir todos en un principio y que no varíen más.

Fuente: https://www.iebschool.com/blog/que-es-el-xp-programming-agile-scrum

Ciclo de vida

El ciclo de vida de XP se organiza como si fuese una conversación cliente- desarrollador.



Éste sería el desarrollo ideal de un proyecto XP. Para acercarnos a esto, se establece un ciclo de

vida dividido en seis fases.

1. Fase de exploración

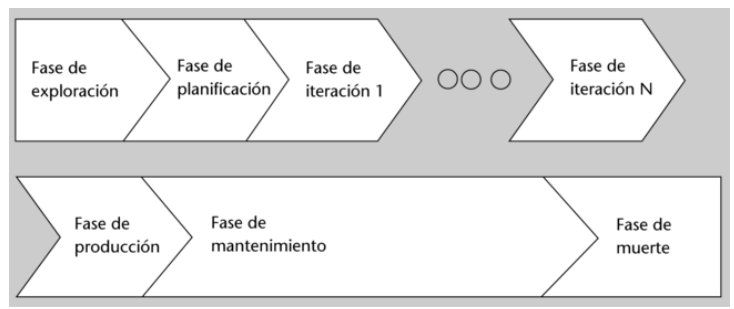
2. Fase de planificación

3. Fase de iteraciones

4. Fase de producción

5. Fase de mantenimiento

6. Fase de muerte del proyecto



1. La fase de exploración: La fase de exploración es la primera fase del ciclo de vida de la

metodología XP. En ella se desarrollan tres procesos:

 Las historias del usuario.

 El spike arquitectónico.

 La metáfora del negocio.

Todo comienza con las “historias del usuario" (users stories). En esta fase los usuarios plantean

a grandes rasgos las funcionalidades y requerimientos que desean obtener del aplicativo. Las

historias de usuario tienen el mismo propósito que los casos de uso, salvo en un punto crucial, las

escriben los usuarios y no el analista. Han de ser descripciones cortas y escritas en el lenguaje del

usuario sin terminología técnica.

Estas historias son las que guiarán la creación de los tests de aceptación que han de garantizar

que dichas historias se han comprendido y se han implementado correctamente.

Las historias de usuario han de ser:

 Escritas por el cliente final, en su lenguaje y sin tecnicismos.

 Descripciones cortas de la usabilidad y funcionalidad que se espera del sistema.

Paralela y conjuntamente se empieza con el spike arquitectónico, en el que el equipo de

desarrollo empieza a familiarizarse con la metodología, herramientas, lenguaje y

codificaciones que se van a usar en el proyecto.

En el spike arquitectónico el equipo de desarrollo:

 Prueba la tecnología.

 Se familiariza con la metodología.

 Se familiariza con las posibilidades de la arquitectura.

 Realiza un prototipo que demuestre que la arquitectura es válida para el proyecto.

Una vez finalizadas las historias de usuario y el spike arquitectónico, se pasa a desarrollar

conjuntamente la metáfora del negocio.

La metáfora del negocio:

 Es una historia común compartida por el usuario y el equipo de desarrollo.

 Debe servir para que el usuario se sienta a gusto refiriéndose al sistema en los términos

de ella.

 Debe servir a los desarrolladores para implementar las clases y objetos del sistema.

2. La fase de planificación: el resultado ha de ser una planificación, de manera flexible, del

proyecto.

El procedimiento es el siguiente:

 El cliente entrega al equipo de desarrollo las historias de usuario que ha

confeccionado, pero priorizándolas de mayor a menor importancia.

 El equipo de desarrollo las estudia y estima el coste de implementarlas.

 Si el equipo de desarrollo considera que la historia de usuario es demasiado compleja,

entonces el usuario final debe descomponerla en varias historias independientes más

sencillas.

 Si el equipo de desarrollo no ve claro cómo implementar una parte de la historia, el

usuario puede realizar un spike tecnológico para ver cómo se podría implantar y así

poder evaluar el coste.

 Una vez tenemos la lista de historias priorizadas junto con su coste de implementación,

pasamos a convocar la reunión del plan de entregas.

3. La fase de iteraciones: como hemos dividido el proyecto en iteraciones, esta fase se repetirá

tantas veces como iteraciones tengamos. Generalmente, cada iteración suele ser de dos a tres

semanas.

El plan de iteración se trata de la siguiente manera:

 Se recogen las historias de usuario asignadas a esta iteración.

 Se detallan las tareas a realizar por cada historia de usuario.

 Las tareas deben ser de uno o tres días de desarrollo. Si son más grandes, se debería

intentar dividir en varias más sencillas.

 Se estima el coste de cada tarea. Si el total es superior al tiempo de iteración, se deberá

prescindir de alguna historia de usuario que se pasaría a la siguiente iteración. Si son

muchas las historias de usuario desechadas, entonces hay que volver a estimar las

cuatro variables de la metodología y volver a planificar el proyecto.

 Si el tiempo total estimado de las tareas es inferior al tiempo de iteración, se puede

asumir una historia de usuario que correspondiese a la siguiente iteración.

 Se priorizan las tareas que más valor darán al negocio, intentando que se finalicen

historias de usuario lo antes posible.

 Se reparten las primeras tareas al equipo de desarrollo y el resto se deja en una cola

de tareas sin asignar de dónde se irán tomando a medida que se vayan finalizando las

anteriores.

Se convocan reuniones de seguimiento diarias para ver si nos vamos retrasando en las

estimaciones o nos vamos adelantando a ellas y así poder desechar o incorporar historias de

usuario.

Lo más importante es que en cada momento de cada iteración estemos realizando la tarea

que más valor posible da al negocio de entre las que tenemos pendientes, de manera que, si

tenemos que reducir el alcance del proyecto, sólo afecte a las funcionalidades secundarias de

nuestro aplicativo.

4. La fase de producción: llegamos a esta fase al alcanzar la primera versión que el usuario final

decida que puede ponerse en producción.

Pasaremos el aplicativo a producción cuando alcance las funcionalidades mínimas que

aporten un valor real al negocio y una operativa arquitectónica estable.

Es decir, no esperamos a tener todas las funcionalidades implementadas, sino que en

cuanto tenemos algo que los usuarios pueden utilizar y que ayuda al negocio, pasamos la

primera versión a producción.

Paralelamente, se sigue con las iteraciones finales de proyecto. De esta manera, antes de

que finalice el proyecto, ya estamos dando valor a la organización, el ROI (retorno de la

inversión) del proyecto empieza a generarse antes de que éste finalice su versión final.

En la etapa de producción se realizan también iteraciones como en la anterior etapa, pero

el ritmo de éstas ya no es de dos a tres semanas, sino mensuales.

Esta fase se mantiene hasta que realizamos la última entrega, con la que finalizamos el

ámbito del aplicativo y pasamos al mantenimiento del mismo.

Durante la fase de producción, el ritmo de desarrollo decae debido a que el equipo debe

solventar las incidencias de los usuarios. Es por esto por lo que a veces es necesario incorporar

nuevo personal al equipo.

5. La fase de mantenimiento: una vez el alcance del proyecto se ha conseguido, y tenemos todas

las funcionalidades en producción, se revisan con el usuario aquellas nuevas historias de

usuario que se han producido tras la puesta en producción del proyecto.

Estas nuevas funcionalidades se van incorporando según su valor de negocio y el

presupuesto adicional del que se disponga.

El equipo de desarrollo se reduce a la mínima expresión, dejando algunos miembros para

el mantenimiento.

6. La fase de muerte del proyecto: Cuando no existen más historias de usuario para introducir

en nuestro sistema o cuando se reduce progresivamente valor de las historias de usuario

implementadas en él, el proyecto entra en la fase de muerte.

Se irá desinvirtiendo en él hasta abandonarlo totalmente cuando no aporte valor al

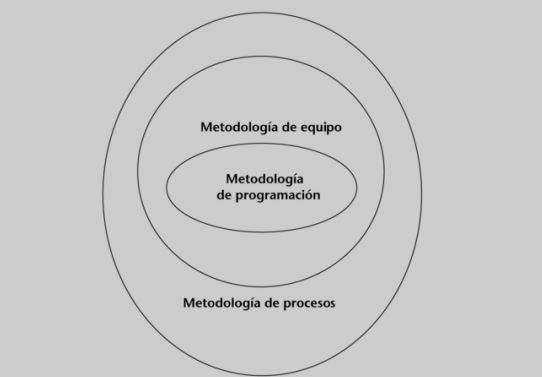
negocio o cuando sus historias de usuario hayan sido absorbidas por otro sistema de

información.

Estructura de la metodología

Para alcanzar el objetivo de software como solución ágil, la metodología XP se estructura en tres

capas:



1. Metodología de programación: diseño sencillo, testing, refactorización y codificación con

estándares.

2. Metodología de equipo: propiedad colectiva del código, programación en parejas, integración

continua, entregas semanales e integridad con el cliente.

3. Metodología de procesos: cliente in situ, entregas frecuentes y planificación.

Introducir la vertiente de las relaciones sociales dentro de una metodología es lo que hace de XP

algo más que una guía de buenas maneras. Convierte a la programación en algo mucho más

humanizado, en algo que permite a las personas relacionarse y comunicarse para encontrar soluciones,

sin jerarquías ni enfrentamientos. Los analistas y programadores trabajan en equipo con el cliente final,

todos están comprometidos con el mismo objetivo, que la aplicación solvente o mitigue los problemas

que tiene el cliente. La vertiente social es fundamental en otras áreas del conocimiento. XP también

humaniza a los desarrolladores. Un entorno agradable para el trabajo, que facilite la comunicación y

los descansos adecuados, forma parte de esta metodología.

Diagramas UML recurrentes

En Extreme Programming (XP), la metodología se centra en prácticas ágiles y en la comunicación directa entre los miembros del equipo de desarrollo. A diferencia de enfoques más tradicionales, XP no tiene una prescripción estricta de diagramas UML (Unified Modeling Language). Sin embargo, algunos equipos que siguen XP pueden utilizar ciertos diagramas UML de manera informal para facilitar la comunicación y la comprensión del diseño. Aquí hay algunos ejemplos:

Diagrama de Clases: Puede ser utilizado para representar la estructura de las clases en el sistema y las relaciones entre ellas.

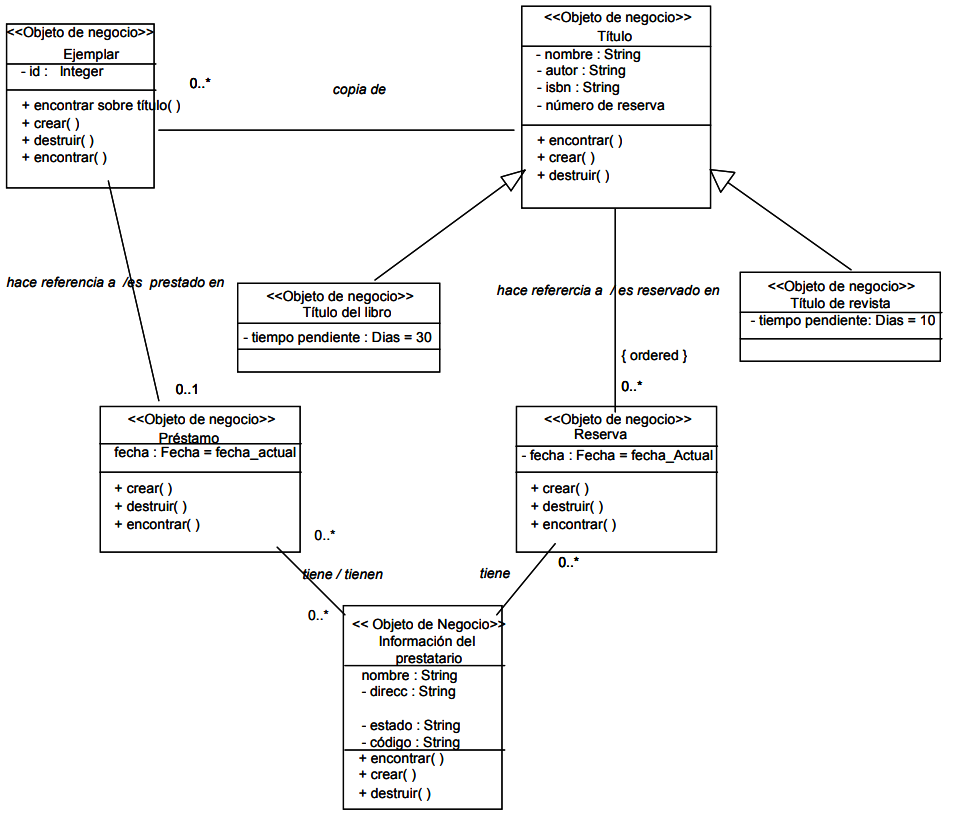


Diagrama de Secuencia: Puede ayudar a visualizar y comprender la interacción entre los objetos y las clases a lo largo del tiempo, lo que es útil para modelar el flujo de trabajo en XP.

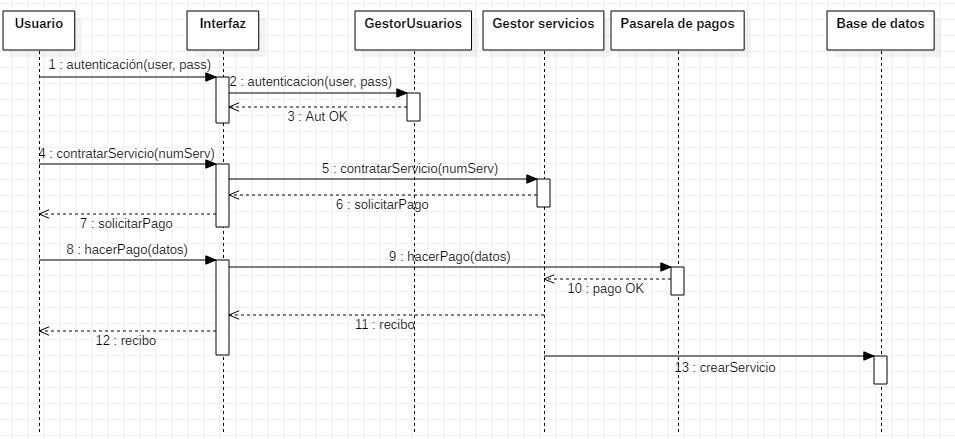
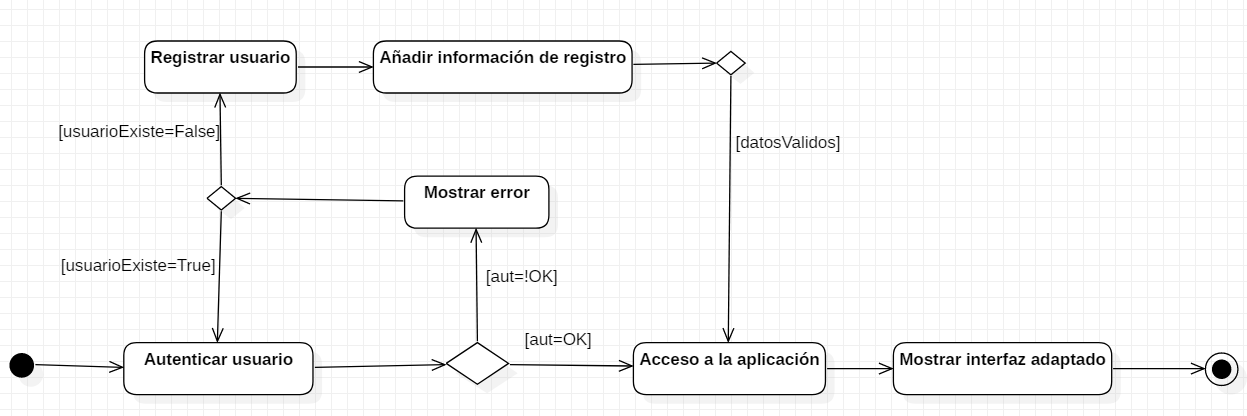
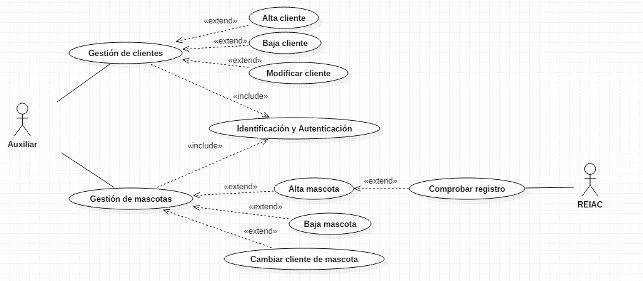


Diagrama de Actividad: Puede ser útil para modelar los flujos de trabajo y las actividades dentro del sistema, especialmente en situaciones donde se busca destacar los procesos y las interacciones.



Diagramas de caso de uso: Los diagramas de casos de uso pueden ayudar a clarificar y visualizar los requisitos del sistema desde la perspectiva del usuario. Esto facilita la comprensión de las funcionalidades clave que deben ser implementadas.



Es importante destacar que en XP, la documentación y los diagramas tienden a ser ligeros y se utilizan de manera pragmática según las necesidades del equipo. La comunicación cara a cara y la simplicidad son valores clave en XP, por lo que la elección de utilizar diagramas UML dependerá de la preferencia y necesidades específicas del equipo de desarrollo.