**¿Qué es el extreme programming?**

El extreme programming (XP), programación extrema en castellano, se considera, y de ahí el nombre de extremo, como la **aplicación más radical del desarrollo ágil de software**. Dicho de otra manera: es probable que no haya ningún método, y mucho menos una forma tradicional de programación, más ágil que el XP. En este contexto, el extreme programming se diferencia concretamente del desarrollo en cascada, por ejemplo, que, según los desarrolladores de XP, implica múltiples problemas. A mediados de los años 90, los desarrolladores Kent Beck, Ward Cunningham y Ron Jeffries decidieron darle un giro al método de trabajo tradicional e iniciar un nuevo camino.

Por norma general, el extreme programming está orientado a las necesidades del cliente. Aunque esto parece obvio, el desarrollo de software clásico solo puede atender a los deseos de los clientes de manera limitada, lo que se vuelve particularmente complicado **ciertos requisitos van cambiando continuamente**. Además, XP trata de fomentar la creatividad de los desarrolladores y acepta los errores como una parte natural del trabajo.

Además, XP, al igual que otros métodos ágiles, parte de los **procesos iterativos**. XP rompe con la tradición de completar un proyecto durante meses de principio a fin para que al final el resultado no sea el adecuado. En lugar de esto, se hacen comprobaciones, se habla y se publica constantemente en ciclos cortos. De esta forma se pueden determinar y eliminar los fallos rápidamente.

Para poder cumplir las exigencias, se ha desarrollado un **marco** bastante definido. Se basa en distintos valores, principios y técnicas. Además, se asignan roles definidos para que las tareas se puedan asignar de forma clara.

**Nota**

En función de qué versión del libro de Kent Beck o qué otra fuente se use para el extreme programming, varía la cantidad de valores, principios y técnicas. No obstante, solo se trata de matices que no alteran el proceso en sí de forma significativa.

***Valores***

La programación extrema trata de modificar el enfoque general del trabajo de programación con cinco valores. La idea es que el equipo como conjunto siga una mentalidad determinada para poder colaborar de la mejor manera y entregar un producto de primer nivel.

***Comunicación***

En el XP no solo importa la comunicación entre los miembros del equipo, sino también la comunicación entre los desarrolladores y los clientes. El intercambio continuo está pensado para poder abordar los problemas directamente. Solo **si todos los implicados están en contacto de forma permanente** se pueden evitar y detectar rápidamente los malos entendidos. Además, la comunicación hace que todos trabajen con el mismo nivel de información y se sientan comprometidos con el proyecto. En este contexto, se prioriza el diálogo cara a cara a la comunicación escrita.

***Sencillez***

El XP siempre busca la solución más sencilla. La sencillez implica varias ventajas: si solo te concentras en los factores necesarios, no te distraes con cuestiones secundarias. Este enfoque también implica **desarrollar solo las funciones necesarias en cada momento** y no adelantar trabajo para posibles requisitos futuros. Así, el equipo desarrolla más rápido. Además, un producto simple es mucho más fácil de manejar, ya sea a la hora de seguir desarrollándolo o de realizar el mantenimiento. Un código de programación simple también facilita la comprensión, otra ventaja a favor del valor de la comunicación. Si todo el equipo es capaz de entender el texto fuente, es más fácil intercambiar impresiones.

***Feedback***

Este valor también va estrechamente ligado a la importancia de la comunicación directa. Es importante que el cliente tenga numerosas oportunidades para expresar sus críticas. En el extreme programming también se usan los mensajes del sistema (logs) como feedback. Para poder llevar a cabo un concepto de feedback como lo plantea el XP, es muy importante pensar en **pasos pequeños**. El equipo trabaja en ciclos cortos, comprueba el código cada poco e incluso le presenta el avance al cliente en intervalos cortos. Así, las modificaciones y correcciones de errores se pueden llevar a cabo rápidamente.

***Valentía***

El extreme programming entiende la valentía como la disposición a decir la verdad, incluso cuando es poco agradable. Si hay **errores en el producto**, hay que señalarlos, aunque sean responsabilidad de uno mismo. En un equipo que trabaja según los valores XP tampoco hace falta disculparse. Ningún miembro del equipo debe perder el tiempo en intentar minimizar su responsabilidad en un error, ya que esto no es productivo. Al margen de lo dicho, en este contexto también se entiende la valentía como la disposición a modificar estructuras de organización, a cuestionar los propios métodos de trabajo, a aceptar las críticas y a escribir nuevos códigos desde cero si hace falta.

***Respeto***

Para que el equipo pueda trabajar de manera armónica y pueda ofrecer un rendimiento excelente, se requiere respeto mutuo. El respeto también implica que un desarrollador no realice modificaciones que tengan un impacto negativo en el trabajo de un compañero. Incluso fuera del equipo, la **importancia de los valores** es primordial hasta llegar al cliente. Solo si nos tomamos en serio las demandas de la otra parte, podremos atenderlas de manera correcta. Finalmente, la dirección de la empresa también debe tratar al equipo de desarrolladores con el debido respeto y aportar las competencias y los recursos necesarios a los empleados.

**Principios**

En el extreme programming, los principios se ubican entre los valores y las prácticas, es decir, son el puente entre lo abstracto y lo concreto. Los principios se pueden derivar en mayor o menor medida de los valores citados anteriormente.

**Feedback inmediato**

El feedback debe solicitarse y aplicarse lo antes posible. En este contexto, la idea es que el feedback del propio sistema (al comprobar el código) se aplique en **cuestión de segundos**, en lugar de recopilar primero todas las impresiones. El feedback de los clientes, en cambio, debe solicitarse y aplicarse en intervalos de días o semanas.

**Tendencia a lo sencillo**

El principio de la sencillez se corresponde con la base del valor con el mismo nombre, pero incluye instrucciones concretas de aplicación. En este sentido se usan dos métodos:

* **You ain’t gonna need it** (YAGNI): mientras una función no se solicite expresamente, no se debe implementar para no realizar trabajo en vano.
* **Don’t repeat yourself** (DRY): debe evitarse repetir tareas y diseñar el código de manera que los cambios no tengan que aplicarse en varios puntos, sino una sola vez, en la medida de lo posible.

**Modificaciones incrementales**

En el extreme programming, las modificaciones siempre se realizan en pasos pequeños. En lugar de implementar grandes actualizaciones para contrarrestar varias fuentes de errores a la vez, solo se trata un problema cada vez. Así, el equipo **reacciona más rápido** y es más fácil entender la causa de las modificaciones. No obstante, el principio no solo hace referencia al código del programa. Los cambios en el diseño o en la estructura del equipo también se deben llevar a cabo en pasos pequeños e incrementales.

**Aceptación de modificaciones**

Como la programación extrema gira alrededor del cliente, sus peticiones gozan de la máxima importancia. Por ello, todo el equipo debe aceptarlas como algo positivo y no oponerse a su realización. La consigna es incluso **animar a los clientes a que manifiesten sus peticiones de cambios** en lugar de intentar convencerlos de que no hacen falta.

**Trabajo de alta calidad**

Suena banal, pero es imprescindible para el funcionamiento del extreme programming: el equipo debe realizar un trabajo de alta calidad. El listón lo pone el cliente. No obstante, para poder realizar un trabajo de calidad, hace falta contar con la dirección. Si los factores son los adecuados y el equipo puede estar contento con el trabajo realizado, dicho equipo va a gozar de un excelente **estado de ánimo**.

**Técnicas**

Las prácticas del XP son instrucciones de actuación y métodos de trabajo muy concretos. Mientras que los valores y principios presentados también se usan en otros métodos de trabajo ágiles, las técnicas concretas del extreme programming son elementos claramente distintivos. Estas técnicas también cambiaron ligeramente y varían en función de la fuente empleada. Por norma general, las técnicas se dividen en cuatro ámbitos distintos.

**Feedback fino**

En la programación extrema, los programadores trabajan con ciclos extremadamente cortos. De esta forma se puede comprobar el código escrito una y otra vez. El denominado **test-driven development** (desarrollo guiado por pruebas) está tan basado en la comprobación que los desarrolladores escriben primero un entorno de prueba antes de comenzar a crear el código fuente en sí. En el momento en el que un código no supere esta prueba, no se puede seguir desarrollando. En este punto, el feedback nace del propio sistema.

El denominado **planning game** (partido de planificación) es una reunión que tiene lugar al inicio de cada fase de desarrollo. El equipo y el cliente se reúnen para valorar el trabajo realizado hasta el momento, aportar opiniones y debatir futuras funciones. A continuación, se asignan las tareas.

La idea de un **on-site customer** (cliente *in situ*) también genera **feedback** continuo. En el mejor de los casos, se busca que al menos un representante del cliente prácticamente forme parte del equipo para poder resolver dudas, aportar ideas o transmitir prioridades más rápidamente.

Finalmente, el [pair programming](https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/pair-programming/" \o "Pair Programming) (programación en pareja) garantiza el principio de cuatro ojos: siempre hay dos personas trabajando simultáneamente en el código. Mientras un desarrollador escribe el código, otro lo comprueba, transmite consejos de mejora y señala errores. Este método es muy caro porque implica el uso de dos trabajadores para un solo trabajo, pero el código resultante es notablemente mejor y exige menos trabajo de repaso.

**Proceso continuo**

Los equipos de XP están continuamente rehaciendo su código. La idea es que el **refactoring** mejore el texto fuente y elimine repeticiones y componentes innecesarios. Un código optimizado de esta forma es más fácil de entender, incluso para lectores externos, y es menos susceptible de contener errores.

A través de la **[continuous integration](https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/integracion-continua/)** (integración continua), los equipos de extreme programming y otros métodos de trabajo ágiles garantizan una integración continua del nuevo código en el proyecto completo. Un desarrollador escribe su trabajo varias veces al día en el proyecto. De esta forma se comprueban continuamente todas las aportaciones y los implicados siempre trabajan con toda la información.

En el XP los programas y las actualizaciones que funcionan se publican lo antes posible. Estos **small releases** (entregas cortas) aumentan la frecuencia de los feedbacks. Así, los errores se pueden detectar más fácilmente y ya se corrigen en la siguiente actualización. El cliente tiene la oportunidad continua de probar directamente los últimos avances del desarrollo e implementar críticas y sugerencias.

**Comprensión conjunta**

Si el diseño es simple (**simple design**) todo el mundo puede entender el código. Por lo tanto, todo lo que complique el texto fuente de manera innecesaria debe eliminarse. Cualquier desarrollador que trabaje según el extreme programming querrá evitar cualquier duplicación. Además, debe quedar claro cuál es el objetivo del programador a partir del propio texto fuente.

Para que todo el equipo pueda trabajar en estrecha colaboración, se definen **coding-standards** (estándares de codificación). Estas indicaciones hacen referencia al enfoque y el formato. Es importante que los desarrolladores puedan trabajar en los códigos de sus compañeros y que sepan qué cambios ha realizado cada uno.

La posibilidad de trabajar en el código de forma conjunta fortalece la propiedad conjunta del código (**collective code ownership)**: en lugar de hacer hincapié en qué parte de responsabilidad (y también de errores) tiene cada uno, se contempla el código como un producto conjunto. De esta forma, todo el equipo es responsable de todo, tanto errores como éxitos. Además, esta técnica invita a seguir desarrollando códigos de compañeros y a integrar nuevas ideas.

Para que el texto fuente sea aún más comprensible, en la programación extrema se usa la técnica **system metaphor**. Esta práctica consiste en describir el proyecto de la manera más sencilla posible, incluso usando metáforas. Esta idea también se aplica a la nomenclatura de objetos, clases o funciones dentro del código, de modo que se han de utilizar siempre nombres autodescriptivos en la medida de lo posible. Así, los trabajadores nuevos que se suman al proyecto entenderán todo mucho más rápido. Así, el código no solo será comprensible para los programadores.

**Bienestar de los desarrolladores**

El bienestar del equipo es importante para el éxito del proyecto. Solo un empleado descansado y motivado puede ofrecer un rendimiento y resultados de calidad. Para garantizar este bienestar, el extreme programming impone una semana de 40 horas. Las horas extra deben evitarse a toda costa y solo se permiten si la semana siguiente no se suman más horas.

**Roles**

Aquí los roles sirven para distribuir todas las competencias entre los implicados, tanto desarrolladores como clientes.

**Cliente**

La programación extrema gira en torno al cliente, hasta el punto que el cliente se considera como parte del equipo y siempre hay al menos un representante presente (on-site customer). El cliente marca las exigencias del producto, pero solo marca de forma limitada cómo cumplir las exigencias. Solo es su responsabilidad marcar la **prioridad por los ámbitos parciales**. Esta responsabilidad también implica expresar sus propios deseos de manera comprensible.

El rol del cliente lo puede desempeñar una sola persona o **un equipo de distintos representantes** del cliente. En la práctica suelen encargarse directores de producto o empleados de la sección de marketing (en función del objetivo del proyecto) de estas tareas.

**Desarrollador**

El equipo de desarrolladores no se divide en subcategorías, es decir, todo aquel que participa activamente en la creación del producto en sí, se considera un desarrollador. Por ello, aquí no solo se incluyen programadores, sino también otras personas implicadas en la creación, en función de los requisitos del proyecto. Aparte del trabajo de desarrollo propiamente dicho, los programadores también tienen la tarea de reaccionar a los deseos del cliente: **calcular el gasto, establecer un marco temporal, planificar la implementación**.

Entre los derechos de los desarrolladores está la posibilidad de **buscar recursos necesarios**, es decir, solicitar a la dirección medios adicionales. Además, el XP impone a los desarrolladores la semana de 40 horas. Es importante para el proyecto que los desarrolladores no estén explotados. Por ello, el equipo de desarrolladores establece su propio horario.

**Director**

El director es el elemento de comunicación entre los desarrolladores y los clientes. Los representantes de este grupo son los que organizan las reuniones entre ambas partes y presiden el planning game, por ejemplo. En este contexto, el director debe prestar atención a que se cumplan las normas acordadas previamente y las convenciones generales de un debate constructivo. El director también debe actuar como **mediador** si es necesario.

Este rol a veces también se denomina como **tracker** (registrador). El motivo es que el director también es el encargado de registrar cifras características importantes, (p. ej., el tiempo que emplea cada trabajador en el proyecto).

**Coach**

Todo el equipo (incluido el cliente) debe estar familiarizado con el extreme programming y saber cómo aplicar este método de trabajo de manera consecuente. Un tutor o coach puede ser útil para que todos tengan la misma idea de los procesos. El coach no participa en el desarrollo del producto en sí, sino que solo brinda su apoyo como **ayudante externo**, de forma similar a un [scrum master](https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/scrum-master/). En conversaciones previas se pueden repasar las normas y prácticas con esta persona. En el mejor de los casos, el coach acompaña al equipo a lo largo de todas las fases del desarrollo y ayuda a resolver dudas.

A menudo se emplea un **proveedor de servicios** externo para esta función. También puede ser que el coach sea de la misma empresa, pero en este caso, de otro departamento. Deben evitarse situaciones en las que una persona desempeña varios roles (por ejemplo, un desarrollador que también desempeña el rol de coach).

## Funcionamiento de la programación extrema

Esta metodología sigue el esquema *planificación > diseño > codificación > pruebas*, desde donde se pasa al lanzamiento o se vuelve a la planificación. Si empezamos en la fase de planificación, vemos que es el momento en que se tienen en cuenta los valores, las historias del usuario, y se establecen los criterios de pruebas de adaptación y el plan de desarrollo. En la fase de diseño tenemos que buscar las soluciones en punto para los diferentes prototipos, pudiendo pasar por varios rediseños hasta llegar a la codificación.

PLANIFICACIÓN

DISEÑO

CODIFICACIÓN

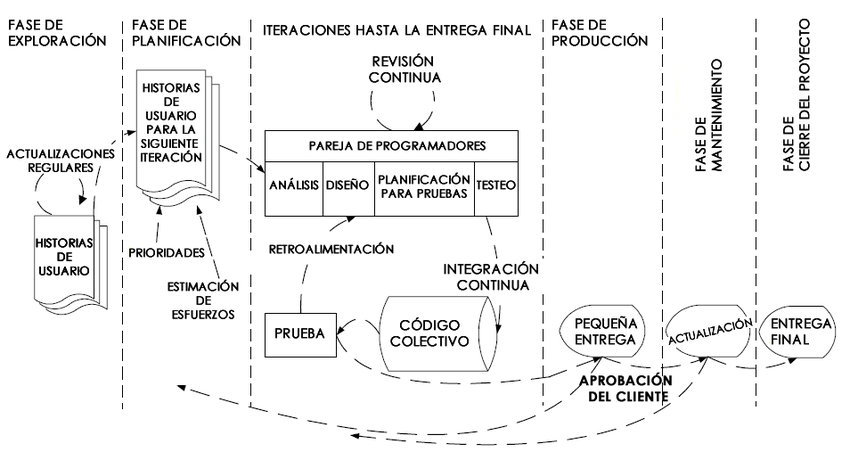
PRUEBAS

En este punto, podemos hacer pruebas unitarias de integración continua, pasando a las pruebas de adaptación. Si el proyecto no cumple con las expectativas volvemos a la casilla inicial y volvemos a iniciar todo el proceso, pero si hemos alcanzado el punto ideal es la hora del lanzar al mercado el proyecto. Conviene destacar que la fase de planificación debe ser flexible y abierta a los cambios, lo que nos permite ofrecer una rápida respuesta a los imprevistos que puedan surgir a lo largo del proyecto.

Ya hemos dicho que el trabajo en equipo es el factor clave para el éxito del proyecto así que es imprescindible que cada actor asuma su rol en el proyecto. Así, los programadores se encargan de estimar los tiempos de desarrollo de cada fase o actividad y programan el proyecto, mientras que el tester se responsabiliza de las pruebas y el tracker de su seguimiento. También hace falta un coacho entrenador, que se ocupa de guiar y orientar al equipo. Un rol que no debemos confundir con el del big boss, gestor o gerente del proyecto, que debe tener una idea general y conocer su estado.

Por último, tenemos el papel del cliente. Será el responsable de definir y conducir el proyecto y sus objetivos, ofreciéndonos un feedback en las diferentes fases donde participe que nos servirán para introducir las mejoras o modificaciones convenientes. Es imprescindible que haya una interacción constante entre el cliente y el equipo de desarrollo, una comunicación en las dos direcciones que nos permita completar el desarrollo de proyecto de la forma más eficiente y efectiva posible.

Partiendo del esquema principal, básico el funcionamiento de XP se desarrolla de la siguiente manera:



Las Fases permiten establecer los momentos en que se realizan las acciones:

**La fase de exploración:** permite la obtención de las historias de los usuarios, requerimientos básicos.

**La fase de planificación:** de define, de acuerdo a prioridades y estimaciones del equipo, las tareas a realizarse en la iteración.

**La fase de producción e iteración:** corresponde a la etapa de codificación y pruebas; aquí se realiza el análisis, diseño, desarrollo y pruebas, que son sometidas a más pruebas y verificaciones, dependiendo del resultado de la retroalimentación se podrá decir que te tiene un entregable correspondiente a esa iteración, una pequeña entrega

**La fase de mantenimiento:** en esta fase se van acumulando todas las pequeñas entregas, estas se adicionan para completar el todo; lógicamente de no ajustarse adecuadamente volverán a una fase anterior.

**La fase del cierre:** comprende la finalización de todas las tareas especificadas en la fase de exploración, incluyendo las tareas o cambios que hayan podido surgir en el camino del desarrollo.

**Ventajas e inconvenientes de la programación extrema**

El extreme programming ha establecido puntos de referencia importantes para el tipo de desarrollo de software, pero no es adecuado para todos los escenarios ni para todos los equipos. El XP parte de la base de que, al comenzar el proyecto, el cliente aún **no tiene una idea definida del producto terminado**. En estos casos, el software se puede planificar y desarrollar de forma ágil, es decir, paso a paso.

Por un lado, esta forma de trabajo satisface al cliente. La solución adecuada se determina con su participación y se **le integra en cada paso**. Por otro lado, los desarrolladores pueden llevar a cabo los proyectos en la manera en que lo consideren apropiado en lugar de tener que estar cediendo siempre. No obstante, si el cliente ya contacta al equipo de desarrolladores con una descripción definitiva del producto y una lista de funciones deseadas, es muy difícil aplicar el método XP.

El concepto de programación en parejas puede suponer un problema para los equipos más pequeños, ya que **carecen de las capacidades necesarias**. Las constantes reuniones con el cliente también implican la dedicación de tiempo que no se puede destinar al trabajo de programación en sí. En una situación ideal, esto no es relevante. El resultado será notablemente mejor si el equipo puede tomarse el tiempo y los recursos necesarios. No obstante, en la práctica, los desarrolladores siempre se ven presionados por un **presupuesto limitado** y unos plazos muy marcados. Además, es posible que el cliente carezca del interés o las posibilidades necesarias para implicarse tanto en el proyecto como lo exige el XP.

Ahora bien, si las condiciones permiten un procedimiento según el método del extreme programming, cualquier equipo puede obtener un **resultado excelente** mediante esta técnica. Las continuas pruebas generan sistemas muy estables y el procedimiento iterativo en colaboración con el enfoque minimalista garantiza que solo se creen funciones que realmente son importantes para el proyecto.