Uruguay Presidencia | agesic

Guía práctica para el uso de metodologías agiles con tecnología PaaS

Desarrollo ágil con integración y entrega continua bajo la plataforma OpenShift

Tecnologías de la Información

Versión 1

Año: 2017

Versión	Autor
1	Diego Rosselli Fernando Maidana Rogelio Rumbo

Introducción

Como insumo para la adopción de prácticas de gestión y desarrollo modernos, incluimos en este trabajo una serie de recomendaciones y buenas prácticas a considerar al momento de ejecutar proyectos enmarcados en metodologías ágiles de gestión con un enfoque de desarrollo basado en integración continua. Las recomendaciones aquí descritas surgen de la experiencia obtenida luego de llevar adelante algunos proyectos, sobre la Plataforma Openshift Container Platform (OCP), que requirieron adaptar en forma sucesiva los mecanismos de gestión tomando elementos extraídos de SCRUM y otros provenientes de metodologías tradicionales.

Si bien las metodologías ágiles plantean una construcción evolutiva del producto se deben presentar los lineamientos principales del mismo a los efectos de contar con elementos que permitan estimar, presupuestar y planificar el proyecto. Estos elementos actúan como restricciones al entorno y provienen de diversos frentes, como por ejemplo, de los mecanismos de adquisición y contratación que rigen en el Estado. Estas restricciones hacen que sea necesario abordar previamente algunos aspectos, como ser, asignación del equipo, disponibilidad presupuestal, plazos y los principales hitos, entre otros.

Es importante además considerar desde el inicio, todo lo referente a la aplicación de las prácticas de desarrollo conocidas como Integración Contínua y Entrega Contínua (CI/CD, por sus siglas en inglés) ya que las mismas influyen sobre varios aspectos del proyecto más allá del desarrollo propiamente dicho. Es por eso que incluimos recomendaciones que refieren al manejo del repositorio de código, la gestión de ambientes, el ciclo de promoción y aprobación de los incrementos, entre otros elementos de interés. El ciclo que se describe incluye actividades de calidad y despliegue automático que tienen por objetivo colaborar con el logro de las expectativas de calidad y facilitar las tareas operativas.

Alcance

En este documento se describen las actividades comprendidas en el proceso de desarrollo de software. El documento asume que las actividades relacionadas con el ciclo de vida del producto y en particular aquellas relacionadas con la definición estratégica del desarrollo del producto ya fueron acordadas previamente por el Product Owner ejerciendo su rol de representante de los interesados en el proyecto ante el Equipo de Desarrollo.

Actividades Previas

Conformación del equipo de trabajo

Recomendamos abordar este tema en etapas tempranas, incluso antes de la publicación del pliego. El hecho de contar con estas definiciones previas permite llevar adelante un conjunto de actividades necesarias de cara a las etapas siguientes, como ser:

- Definiciones de arquitectura.
- Objetivos de calidad y automatismos.
- Requisitos de infraestructura.
- Ciclos de promoción y aprobación.
- Acuerdos del grupo de trabajo.
- Herramientas a usar para la gestión del proyecto.

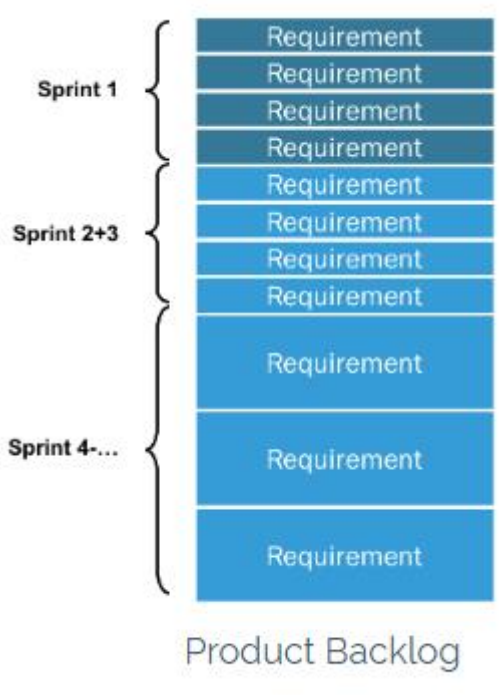
Más allá del interés que tenga cada miembro del equipo, un aspecto importante a considerar es **definir el equipo de trabajo**, considerando los perfiles de cada individuo para poder cubrir los roles con los que se debe contar a fin de asegurar la correcta ejecución de las actividades del proyecto. Los roles antes mencionados son los siguientes:

- Dueño del producto (Product Owner, en adelante PO)
- Responsable de Calidad
- Responsable de Arquitectura
- Responsable de Automatismos
- Responsable de Operaciones
- Responsable de Seguridad

Es posible que algunos de los roles tengan una contraparte directa en el proveedor de desarrollo (si es el caso). Al mismo tiempo es condición excluyente que el rol de Product Owner (PO) sea llevado adelante por un integrante del organismo ya que es el representante del cliente ante el equipo de desarrollo.

Definición del Backlog

En esta instancia, el equipo trabaja en la definición a alto nivel (historias épicas) las necesidades del producto, considerando para cada una de ellas las expectativas relacionadas con los principales atributos de calidad (disponibilidad, performance, tolerancia a fallos, testeabilidad, seguridad, usabilidad, entre otros).



El resultado de este trabajo es la conformación de una versión inicial del Product Backlog (PB) desde el cual se seleccionarán elementos a incluir en cada uno de los Sprints sucesivos mediante la elaboración del Sprint Backlog (SB). En el desarrollo de los sprint estas necesidades van a ser refinadas, detalladas y priorizadas hasta obtener historias de usuario listas para desarrollar en los sprints subsiguientes. Este criterio de "listo para ser desarrollado", es el que abordamos a continuación.

Definición de "Listo" (Definition of Ready)

Es recomendable elaborar una versión inicial de los criterios mínimos que debe cumplir una historia de usuario para estar en condiciones de formar parte de un Sprint Backlog (SB). Esto no es otra cosa que definir todos los elementos necesarios para indicar que una historia de usuario está "lista para ser desarrollada". Estos criterios serán refinados luego, en conjunto con el equipo de desarrollo a fin de que los mismos sean claros, sin ambigüedades y entendidos por todos.

Tomar esta definición y trabajarla en conjunto, tiene un impacto inmediato en la dinámica de trabajo ya que evita las idas y vueltas innecesarias entre el PO y el equipo de desarrollo por falta de entendimiento en el contenido de las historias. A continuación enumeramos algunos criterios que puedan ser de utilidad al momento de elaborar la definición:

- Deben contar con las características “INVEST”. Por sus siglas en inglés:
  - Independiente (Independent)
  - Negociable (Negotiable)
  - Valiosa (Valuable)
  - Estimable (Estimable)
  - Pequeña (Small)
  - Verificable (Testable)
- Debe estar escrita en la forma:
  - “Cómo [Rol]”
  - “Quiero [expresión de deseo]”
  - “Para [objetivo]”
- Todas las Historias de Usuario (en adelante HU) deben contar al menos con un criterio de aceptación.
- Una HU no debe insumir más de 40 horas. De ser el caso, debe refinarse hasta lograr una mayor granularidad de tareas.
- Pueden incluir notas técnicas que sirvan de guía para el equipo de desarrollo.

Definición de “Hecho” (Definition of Done)

Esta definición refiere a los criterios que debe cumplir un incremento para considerarse listo para ser entregado. Como “incremento”, podemos referirnos a una HU o a un Sprint completo. Una regla que es posible tomar, es que el criterio de hecho de todas las HU que componen el Sprint, deben estar alineadas con el criterio de hecho del Sprint del que forma parte.

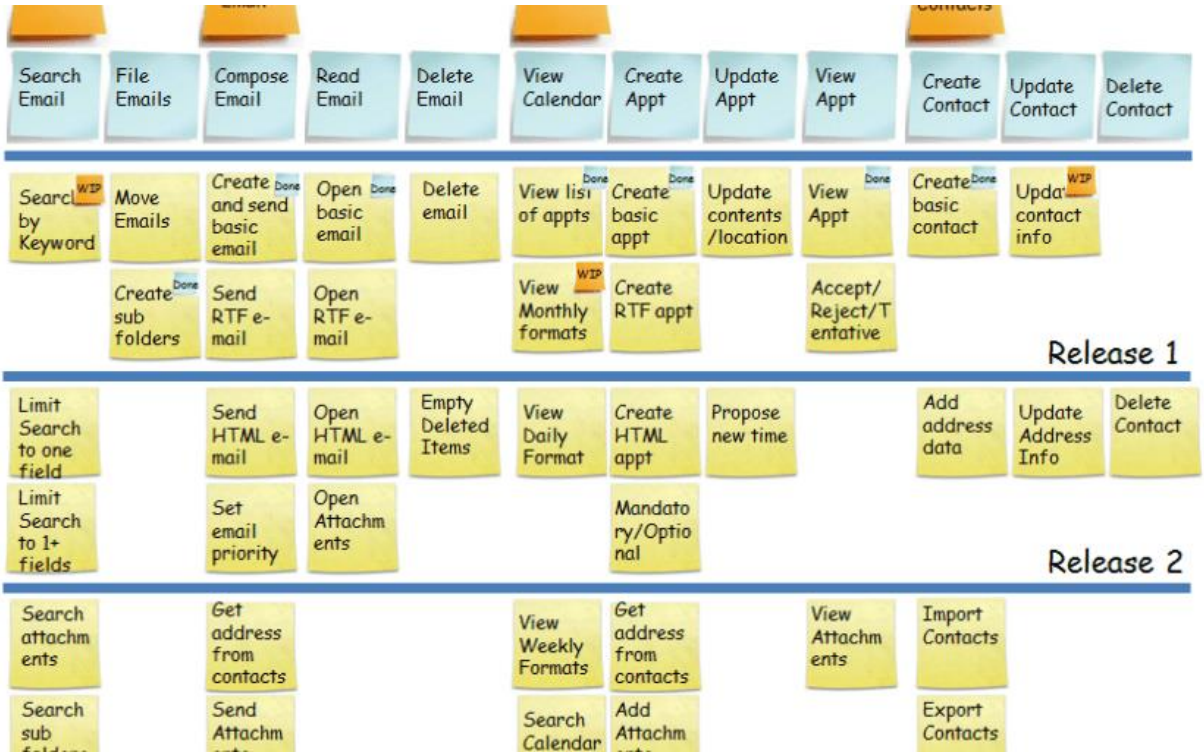
Por otra parte, pueden definirse además algunos criterios, con la suficiente amplitud como para ser aplicables a cualquier incremento durante el ciclo de desarrollo del producto. De contar con esta “definición de hecho general”, es posible adoptar la regla de mantener alineadas las definiciones de hecho de todos los sprint con la definición general antes mencionada.

En estas definiciones pueden incluirse aspectos técnicos, métricas de calidad o cualquier elemento que se considere relevante respecto a la entrega de valor asociada con cada incremento. A continuación, enumeramos algunos de los elementos que es posible considerar dentro de la definición:

- Cumplimiento de requerimientos funcionales y criterios de aceptación
- Deuda técnica
- Cobertura de pruebas unitarias
- Documentación (arquitectura, operación y mantenimiento, Acuerdos de alcance y técnicos)
- Automatización de despliegue y pruebas funcionales
- Pruebas funcionales manuales

Plan de *release*

Se debe elaborar un plan de entregas de lo que se consideren mínimos productos viables a lo largo del proyecto. Este plan, tiene por objetivo principal dirigir la entrega de valor, facilitando de esta forma el manejo de expectativas respecto al producto. Adicionalmente, es un insumo importante a la hora de priorizar las historias épicas del product backlog que formarán parte de cada incremento.



Si bien la responsabilidad de elaborar dicho plan es del PO, es posible que se pueda delegar esta actividad al equipo de trabajo si existen las capacidades adecuadas. En caso de que no sea posible abordar esta tarea, es recomendable elaborarlo en conjunto con el equipo de desarrollo, ajustando lo planificado en la medida que se avanza en la consecución de los Sprints. Cabe recalcar la importancia de contar con visión del negocio para realizar esta actividad.

En caso de que el plan sea desarrollado dentro de las actividades previas, es de suma importancia contar con la participación de todos los miembros del equipo a fin de enriquecer la visión del producto con los diversos enfoques y colaboren en la concreción de las expectativas de forma conjunta.

Desarrollo y Puesta en Producción

Kick-off del Proyecto

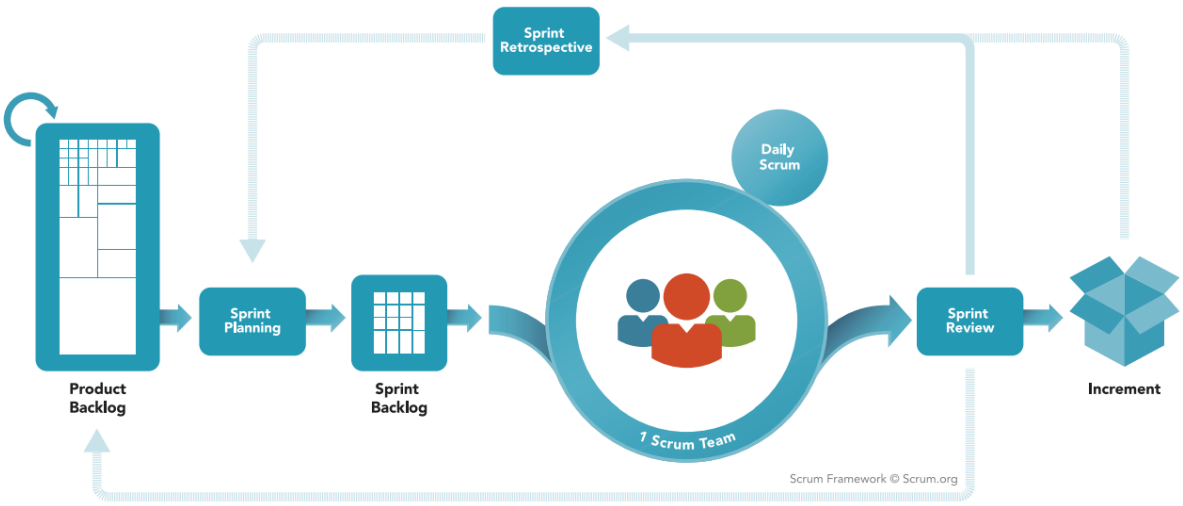
El objetivo de esta actividad es ajustar los puntos tratados y acordados en la sección "Actividades Previas" conjuntamente con el equipo de desarrollo (sea este interno o mediante un proveedor seleccionado), a fin de preparar el terreno para el inicio de las actividades del proyecto.

Modalidad de Ejecución de Sprints

En este apartado presentamos las principales adaptaciones a partir de la ejecución de sprints sugerida por SCRUM. El objetivo principal de las adaptaciones aquí descritas, buscan propiciar las condiciones para trabajar en equipo, aprovechando las potencialidades aportadas por el organismo respecto al negocio y el expertise técnico del equipo de desarrollo sea este interno o externo.

Al mismo tiempo, las adaptaciones surgen de ensayos sucesivos, prueba y error, para sacar el mejor provecho de la modalidad de gestión y las prácticas de desarrollo. Este ejercicio de adaptación y mejora es continuo, por lo cual exhortamos a los equipos a construir su propia visión, eliminando, modificando y adaptando lo necesario para lograr el mejor funcionamiento del equipo.

A continuación, presentamos un esquema de la ejecución del sprint con sus principales ceremonias y productos de trabajo, los cuales iremos detallando a lo largo de esta sección:



En este apartado, describiremos algunas ceremonias con las recomendaciones y adaptaciones que llevamos adelante en base a la experiencia. A su vez, detallaremos algunas actividades que no son ceremonias, pero deben ser llevadas adelante para dar

sustento al ciclo de “Promoción y Aprobación”, como es el caso de la “Aprobación del Incremento” y la “Revisión de Documentación”.

Reunión técnica

En esta ceremonia podrían participar todos los roles, ya que sus múltiples enfoques enriquecen la solución técnica y al mismo tiempo minimizan el retrabajo. Se dan acuerdos técnicos que serán utilizados a posteriori en las ceremonias siguientes (refinamiento y planning).

Pueden surgir de esta ceremonia, acuerdos que tengan distinto alcance en lo que refiere al proyecto, por lo cual recomendamos fuertemente dejar registro de los acuerdos alcanzados, sea cual sea el producto de trabajo sobre el que tengan impacto. Algunos de los elementos que pueden verse influidos son:

- Otras Historias (Épicas o de Usuario)
- Definición de “Hecho” General
- Definición de “Hecho” del Sprint
- Definición de “Listo”
- Documentación de Arquitectura
- Objetivos de Calidad, Quality Gates u otros
- Flujo de promoción y Aprobación
- Aspectos Operacionales (Manuales, Monitoreo, Backup u otros)
- Automatismos
- Guías y/o Manuales de Usuario

Entradas

- Product Backlog

Salidas

- Historias de Usuario refinadas desde el punto de vista técnico
- Documentos actualizados con los temas discutidos

Participantes

- Dueño el producto (Product Owner)
- Equipo de desarrollo
- Dependiendo de la temática, todos los roles que sean necesarios.

El resto de los roles podrían participar dependiendo del alcance del sprint.

Refinamiento

Antes de esta ceremonia de Planning, el equipo del organismo, encabezado por el PO deberá seleccionar del product backlog las historias épicas que se deseen desarrollar, para refinarse en historias de usuario. El producto de trabajo de esta ceremonia, debe ser un

conjunto de HU listas para ser parte del siguiente Sprint, por lo cual, deberán cumplir con la "Definición de Listo".

Es importante refinar más HU de las que pueden ser incluidas en el sprint, de forma de contar con elementos que aporten flexibilidad al momento de planificar el mismo. Un aspecto a considerar es hacer el ejercicio de refinar tarjetas pequeñas, medianas o grandes (en esfuerzo) a fin de poder aprovechar el "remanente" del sprint al máximo, asignando las historias de forma más óptima. Dichas historias de usuario deberán estar priorizadas a los efectos de darle más insumos al equipo de desarrollo a la hora de planificar el sprint.

En una ceremonia de refinamiento posterior, es recomendable que el equipo de desarrollo estime y divida en tareas todas las historias de usuario refinadas por el equipo del organismo, para de esta forma optimizar los resultados de la ceremonia de planificación del sprint (Sprint Planning). Al contar con tarjetas pre-refinadas el equipo de desarrollo puede también estimar previamente las mismas para optimizar los tiempos y resultados de dicha ceremonia.

Si no fuese posible que el equipo lleve adelante el refinamiento, este se puede hacer con la tutoría del proveedor de desarrollo, reestructurando las ceremonias para contemplar este caso.

El ejecutar la secuencia de ceremonias descrita anteriormente, permite optimizar los resultados de la reunión de Planificación ya que llegado el momento, las HU cumplen con el criterio de listo y están priorizadas. Lo que se busca evitar es salir de la ceremonia de planificación con un tentativo de las tareas a realizar, sin certezas respecto a la factibilidad del sprint.

Entradas

- Product Backlog

Salidas

- Historias de Usuario
  - Cumplen con el criterio de listo
  - Posiblemente deban ser refinadas en conjunto con el equipo de desarrollo

Participantes

- Dueño del producto (Product Owner)
- Responsable de Calidad

El resto de los roles podrían participar dependiendo del alcance del sprint.

Sprint Planning

Teniendo como insumo las HU propuestas por el PO y el equipo de trabajo, las cuales fueron priorizadas, refinadas, estimadas y cumplen con el criterio de listo, se procede a planificar el sprint. Esta actividad consiste en seleccionar las historias de usuario a ser incluidas en el sprint, utilizando como insumo principal la prioridad de cada una de ellas.

Es posible que sea necesario ajustar algún elemento de las historias de usuario en conjunto con el equipo de desarrollo, ya que el mismo cuenta generalmente con mayor conocimiento de la solución. De ser el caso, se deberá modificar la descripción, estimación o incluso la prioridad de las historias de acuerdo a las propuestas recibidas.

A medida que el equipo va ganando experiencia, es de esperar que las estimaciones mejoren en precisión con el correr de los sprints. Por tal motivo es fundamental el feedback del equipo de desarrollo, tanto para ajustar sobre la marcha como para las futuras reuniones de planificación.

Entradas

- Historias de usuario refinadas por el equipo del organismo y estimadas por el equipo de desarrollo.
- Horas de desarrollo disponibles.

Salidas

- Planificación del sprint

Participantes

- Dueño del producto (Product Owner)
- Scrum Master
- Responsable de calidad
- Equipo de Desarrollo
-

Al igual que en otras ceremonias, de acuerdo al alcance y los objetivos del sprint, es posible que concurran otros miembros del equipo. Por ejemplo, si el sprint involucra cambios de arquitectura, se puede contar con la presencia del responsable de Arquitectura. Lo mismo sucede con otros aspectos relevantes del proyecto, como puede ser el caso de los automatismos.

Daily

Si bien esta ceremonia se lleva adelante principalmente por el equipo de desarrollo, conjuntamente con el Scrum Master, hemos comprobado que participar periódicamente en algunas de ellas, trae aparejados algunos beneficios. Debido al propósito y dinámica de esta ceremonia, es recomendable que asista únicamente el Product Owner (PO).

Además de contribuir a generar sinergia y mantenerse al tanto de los detalles en el avance del proyecto, aporta al PO mayor conocimiento técnico sobre diferentes aspectos del proyecto y ayuda a aportar visión de producto al equipo de desarrollo. El principal efecto que esto tiene, refiere a encauzar las decisiones del equipo y el PO, evitando decisiones que se alejan de los objetivos del proyecto y a la postre puedan provocar desvíos.

Debido a que la naturaleza de esta ceremonia indica que la misma debe ser destinada al Equipo de Desarrollo, recomendamos participar de forma esporádica, acordando la cadencia en conjunto, a fin de no invadir el ámbito del equipo. Por ejemplo, podría establecerse una participación del PO dos veces por semana.

Entradas

- Intercambio con el equipo sobre el avance y los principales obstáculos
- Aportes del PO (cuando participa) y el SM respecto a su visión del estado del sprint

Salidas

- Acciones a tomar para despejar los obstáculos que visualiza el equipo
- Coordinación y distribución del trabajo

Participantes

- Equipo de Desarrollo
- Scrum Master
- Dueño del producto (Product Owner, esporádicamente)

Reunión de seguimiento

Esta ceremonia, tiene por objetivo contar con un punto de control, preferiblemente en la mitad del sprint en ejecución. Su objetivo principal es tomar una actitud proactiva, previniendo desvíos, principalmente en lo que refiere a las horas insumidas en las distintas historias incluidas en el sprint en contraste con las estimaciones de las mismas.

De estas ceremonias, se pueden extraer aprendizajes valiosos y una idea más ajustada del desempeño del equipo de desarrollo, siendo posible incluso, contar con información para

elaborar métricas de productividad como la “velocity” que puede encontrarse en la bibliografía de SCRUM.

La experiencia nos ha demostrado que el hecho de llevar adelante esta ceremonia previene que los imprevistos y modificaciones de alcance se encuentren demasiado tarde, como por ejemplo al momento de la review, donde ya no es posible tomar medidas para corregir los posibles desvíos.

Entradas

- Información proveniente de la planificación del sprint (historias, estimaciones, obstáculos u otros)
- Tiempo insumido en cada historia

Salidas

- Acciones a tomar para corregir eventuales desvíos
- Información procesada para la extracción de métricas al final del sprint

Participantes

- Scrum Master
- Product Owner
- Equipo de Desarrollo

Review

Al final del sprint para validar que se cubrió el alcance acordado, se realiza esta ceremonia en la que debe llevar adelante una demo del producto, con especial énfasis en los elementos incluidos en el incremento y su funcionamiento. Es importante destacar el hecho de que la demostración juega un papel fundamental en esta reunión ya que el centro de la misma es contar con un producto funcional, más allá de que el mismo sea considerado o no, el mínimo producto viable. Es importante aclarar que la demo no es para realizarle testing a la misma, sino para tener un primer acercamiento al producto desarrollado.

Conjuntamente con la demostración, se deberán revisar los criterios de aceptación de cada una de las historias, dejando claro la forma en la que se alcanzan los criterios planteados. Por otra parte, se debe demostrar la forma en la que se cumple con la definición de hecho de las historias, con la del sprint que acaba de finalizar y con los criterios de hecho generales en caso de que existan.

Además de las funcionalidades incluidas en el incremento, se debe contar con los reportes de resultados de cada una de las actividades automáticas provenientes del ciclo de Integración Continua (sean estas automáticas o manuales), a fin de que el equipo pueda verificar el cumplimiento de los objetivos de calidad definidos para el sprint y que los mismos se encuentran alineados con las definiciones generales del proyecto.

Recomendamos mantener la reunión los más enfocada posible. Para lograrlo será necesario elaborar cuidadosamente la lista de participantes, tratando de prevenir actores que puedan generar conflictos con escaso valor.

Entradas

- Producto funcional
- Alcance definido para el sprint
- Definiciones de hecho aplicables (para las historias, el sprint y generales)
- Reportes de actividades automáticas y manuales

Salidas

- Historias de usuario aprobadas
- Historias de usuario modificadas (si no fueran aprobadas)
- Nuevas historias agregadas al Product Backlog (PB)

Participantes

- Product Owner
- Scrum Master
- Interesados en el proyecto (principalmente con visión de “cliente”)
- Equipo de Desarrollo

Aprobación del incremento

Una vez realizada la entrega del incremento en el ambiente de Staging, se procede a la validación del entregable. Esto implica ejecutar las actividades manuales definidas, así como también la ejecución del Pipeline a fin de obtener los resultados de las actividades automáticas. Algunos de los elementos a tener en cuenta en este momento pueden ser:

- Resultados del análisis estático de código.
- Manuales de Arquitectura, Instalación y Mantenimiento
- Cobertura de pruebas unitarias.
- Resultados de las pruebas automáticas.
- Resultados de las pruebas manuales

En este punto del ciclo de promoción, es en donde el equipo de calidad tiene la oportunidad de aportar mayores niveles de garantía agregando y/o modificando el conjunto de pruebas automáticas o manuales que se llevan adelante. En caso de modificar cualquiera de las pruebas sobre este ambiente, recomendamos fuertemente volcar este nuevo conjunto de pruebas al equipo de desarrollo para que las mismas puedan ser ejecutadas en el próximo incremento. Esto permite evitar “idas y vueltas” innecesarias en el sprint siguiente ya que de no mantener sincronizada la base de pruebas, los criterios de aprobación del incremento serían diferentes para QA y Staging.

Simultáneamente a la revisión funcional, el equipo del organismo deberá verificar que la documentación haya tenido un incremento acorde a los elementos incluidos en el sprint y

que los productos de trabajo afectados por el incremento hayan sufrido las modificaciones necesarias.

Dependiendo del carácter del producto en cuestión, es posible que se requiera la intervención de distintos miembros del equipo a fin de que evalúen el estado de los documentos u otros elementos comprendidos en el incremento. Recordamos que puede darse el caso de que determinado incremento no sea aceptado por cuestiones que van más allá de lo estrictamente funcional, como por ejemplo, el hecho de no contar con evidencia de pruebas, documentación de arquitectura incompleta o cualquier otro elemento que el equipo considere pertinente.

En caso de que se den las condiciones para continuar con el ciclo de aprobación, promoviendo así el incremento al siguiente ambiente, se deben repetir las actividades aquí descritas con las diferencias que correspondan a cada uno de los ambientes sucesivos, considerando nuevamente en cada caso, el cumplimiento de los criterios para promover el incremento hasta su llegada a producción.

Entradas

- Producto funcional en Staging (o cualquier otro ambiente sucesivo)
- Resultado de actividades de calidad
- Feedback de usuarios u otros interesados respecto a las características visibles del producto

Salidas

- Producto aprobado y promovido o en su defecto rechazado
- Registro de actividades de cada una de las promociones y sus respectivos resultados.
- Lecciones aprendidas y recomendaciones para tomar acciones futuras

Participantes

- Todos el Equipo

En ocasiones, dependiendo de las tareas a realizar, puede que se requiera la participación del equipo de desarrollo.

Retrospectiva

Esta ceremonia propicia la mejora continua analizando diversos aspectos de la ejecución del proyecto. A fin de ayudar a enfocar la reunión, comúnmente la misma se centra en los siguientes aspectos :

- Qué cosas han funcionado bien (a repetir o continuar haciendo)
- Cuáles hay que a evitar (no volver a hacer)
- Qué cosas se quiere probar hacer en el siguiente sprint (acciones)
- Qué se ha aprendido (incorporar)

- Adelantarse a los problemas que podrían impedir progresar adecuadamente. (a prevenir)

Para que los efectos de esta ceremonia sean sostenidos en el tiempo, es importante el compromiso con la incorporación de las acciones definidas por el equipo. Una aspecto importante que puede ayudar en esta dirección, es realizar de forma periódica lo que se conoce como la “Retrospectiva Ampliada”, que además del equipo de desarrollo y el PO, incluye interesados en el proyecto, dependiendo de el carácter de los temas que se deseen abordar en la ceremonia.

Es importante resaltar que el objetivo principal de esta reunión es tomar acciones sobre aquellos elementos que el equipo entienda de mayor relevancia. De eso se desprende que la misma debe enfocarse en las acciones a seguir en el siguiente incremento para reforzar los aspectos positivos y aprovechar las oportunidades de mejora que se presentaron en el incremento anterior.

Entradas

- Aspectos relevantes a ser tratados, según entienda cada uno de los participantes de la ceremonia

Salidas

- Acciones a tomar para aprovechar las oportunidades de mejora

Participantes

- Product Owner
- Scrum Master
- Equipo de Desarrollo

En la retrospectiva ampliada pueden participar otros actores según se entienda necesario.

Ciclo de hotfix

¿Qué sucede si detectamos un bug en producción cuando estamos en medio de la ejecución de un Sprint? Hay muchos caminos para tomar aquí y la decisión corresponde mayoritariamente al Product Owner, con apoyo del equipo de desarrollo.

Existen además otras consideraciones a tener en cuenta como ser:

- ¿Existirá un ciclo de promoción y aprobación abreviado para este tipo de casos?
- ¿Cómo se llevará adelante el manejo de repositorio de código al momento de corregir código que hoy se encuentra en producción?

- ¿Cómo incorporamos el nuevo código al desarrollo actual del sprint?

La respuesta a las preguntas planteadas anteriormente puede servir como guía al momento de tomar acciones de darse la situación en cuestión. De hecho, estas definiciones pueden ser tomadas por el equipo dentro de las “Actividades Previas” a fin de facilitar el trabajo y tener un plan de acción definido al momento de afrontar la situación. Pueden encontrarse recomendaciones para abordar las interrogantes anteriores en el documento de “Aportes al Marco de Calidad”.

Las consecuencias sobre el sprint en desarrollo deberán ser evaluadas por el PO, con apoyo de los miembros del equipo que se estimen relevantes. La decisión final sobre la afectación del sprint será responsabilidad principal del PO, quien tiene la potestad de definir las modificaciones de alcance del sprint.

Pruebas Rutinarias

Las prácticas de integración contínua, el ciclo de promoción y aprobación y la definición de los ambientes, tienen por objetivo aumentar los niveles de certeza desde que el equipo de desarrollo entrega un incremento, hasta su llegada a Producción.

Sin embargo, aun en los ciclos más maduros, existen ocasiones en las que no se pueden anticipar ciertos comportamientos hasta que no ocurra una combinación de factores sobre el ambiente productivo propiamente dicho. Es por eso que recomendamos efectuar de forma rutinaria, cada cierto tiempo, pruebas que tengan por objetivo verificar el comportamiento del sistema en su entorno productivo.

Estas actividades pueden comprender por ejemplo, pruebas de stress, pruebas de penetración u otras actividades que, al llevarse a cabo sobre el ambiente productivo, aporten información concluyente sobre diferentes aspectos del comportamiento del sistema, que puedan brindar información valiosa para aprovechar las oportunidades de mejora que ofrece el producto en determinadas situaciones.

Estas actividades deben ser programadas adecuadamente ya que posiblemente requieran la definición de ventanas de indisponibilidad, notificación a los distintos usuarios del producto y otras actividades como respaldos y monitoreos específicos para la actividad en cuestión. Es posible además que el área de operaciones indique un conjunto de pruebas tales como pruebas de restore, HA, entre otras.

Operación y Mantenimiento

Estrategias de Despliegue

Hoy en día existen varios prácticas que pueden aplicarse al momento de desarrollar, desplegar y mantener software. Si bien podemos pensar en modelos eficientes dentro de IaaS, asegurar la continuidad del negocio en ciertos escenarios puede ser dificultoso. Sin embargo en PaaS, es posible asegurar la continuidad del negocio inclusive en casos desfavorablemente complejos como por ejemplo, durante la actualización de una solución, definiendo estrategias de despliegue y re modelando procesos hacia la puesta en producción.

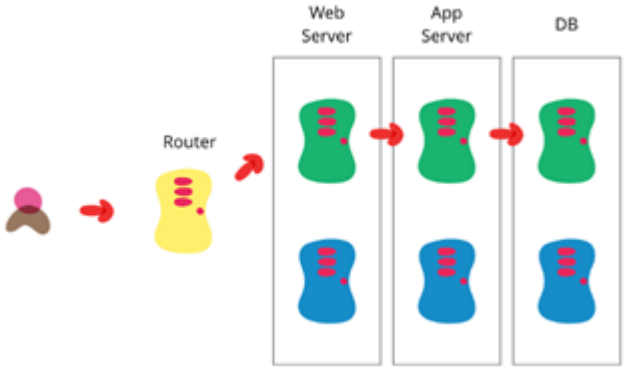
En este contexto; ¿qué hace Operaciones?

Ofrece **Recomienda** Acuerda

…las estrategias de despliegue más acordes a la solución en cuestión, comprendido en el marco de un proyecto. Las herramientas de CI-CD permiten automatizar las distintas etapas del proceso, desde las etapas iniciales del desarrollo hasta el despliegue final en el ambiente productivo.

Estrategia de despliegue “Blue-green”

Esta es una de las estrategias más recomendadas por Operaciones y si bien no es la única, permite llevar adelante una actualización, manteniendo los niveles de servicio definidos para la solución, de forma transparente hacia los usuarios. Su funcionamiento es el siguiente:



1. Partimos de un incremento A (green) ya en producción y recibiendo el 100% de tráfico.
2. Se despliega de forma independiente la nueva versión B (blue) de forma de que ambas estén ejecutándose al mismo tiempo.
3. Luego de validado el nuevo incremento, se pasa el tráfico de la versión A a la versión B (en caso de una maniobra exitosa se elimina la versión antigua quedando operativa únicamente la nueva)

Esta estrategia, obliga a duplicar los recursos destinados a la solución con el objetivo de minimizar el impacto o downtime y al mismo tiempo propone un rollback mucho más sencillo y efectivo, en el caso de una maniobra fallida.

Es importante recalcar que el uso de esta estrategia tiene implicancias a nivel de desarrollo, ya que por ejemplo, en caso de que ambas versiones compartan un repositorio (como por ejemplo una base de datos), se requiere que las mismas sean compatibles

respecto al repositorio compartido. Esta técnica de desarrollo se conoce como “N+1 Compatible”.

Flujo de aprobación y despliegue:

Para tener éxito en un entorno de este tipo, es conveniente automatizar actividades comprendidas dentro del ciclo de vida de la solución. Para transitar el el ciclo definido, es necesario establecer las actividades a llevar acabo para aprobar y promover la versión, desde el ambiente de desarrollo hasta el productivo. Operaciones tiene allí su cuota de participación y las actividades a ejecutar dependerán de la estrategia pautada. Por lo tanto y como se mencionó anteriormente, es necesario definir una estrategia de despliegue a seguir, de acuerdo a las necesidades de negocio y en las fronteras de la aplicación a implementar.

A modo de resumen, las necesidades de negocio pueden implicar considerar los siguientes temas:

- Despliegue continuo
  - Blue/Green
  - A/B
- Técnicas de Desarrollo para soportar el despliegue
  - N+1 compatible

Considerar estos y otros aspectos de la operativa de la solución en etapas tempranas del proyecto, tiene implicancia directa en los distintos procesos operativos que se apliquen a futuro. Por ejemplo, en una estrategia Blue/Green con N+1 Compatible respecto a la base de datos, el proceso de rollback debe definirse específicamente para estos casos ya que difiere de un proceso tradicional.

Hemos enumerado algunos elementos que ilustran la importancia que tiene acordar, según la estrategia de despliegue establecida anteriormente, un completo flujo de aprobación y definir la participación del equipo de Operaciones en las instancias del ciclo de vida, lo cual incluso puede llegar a tener implicancia directa en las actividades de desarrollo.

Monitoreo y Manejo de Logs

Estos aspectos deben ser considerados desde las etapas de diseño de la aplicación de manera de cubrir las necesidades del negocio y mantener la máxima compatibilidad con la plataforma.

La plataforma PaaS se encuentra monitoreada, es decir que la infraestructura que le da soporte a toda la plataforma de contenedores como servidores virtuales, routers, bases de datos y los nodos de contenedores (donde viven las aplicaciones) se cuentan con los servicios necesarios que permiten determinar su estado de salud.

Es necesario definir el monitoreo de los componentes externos que den soporte a la aplicación, entre los cuales puede considerarse:

- Bases de datos
- Storage externos
- Dispositivos de enrutamiento y monitoreo de tráfico de red

De todos modos se recomienda verificar las capacidades de monitoreo existentes para aquellos componentes que puedan afectar la disponibilidad de la aplicación.

Por otro lado, es necesario definir de manera conjunta el monitoreo específico que se precisa para la aplicación. Estos monitoreos incluyen pero no se limitan a:

URLs de estado de salud de la aplicación, páginas que indican la disponibilidad de la aplicación: Checks que pueden ser consultados para conocer la condición de la aplicación.

Estados de negocio: Checks que pueden ser consultados por sistemas de monitoreos como Zabbix que presentan un estado de la aplicación basado en métricas de negocio.

Logs de consola: La plataforma dispone para la captura y el manejo de logs (EFK, Elasticsearch-Fluentd-Kibana) y la posibilidad de enviar logs a un syslog externo. Es importante entender que es un requerimiento que los logs se escriban en consola y no en disco para que el stack funcione.

Utilización de recursos: La utilización de CPU y Memoria forman parte de los checks que la plataforma realiza sobre los contenedores para entre otras cosas toman acciones automáticas en escenarios de auto escalado. Es muy importante resaltar que no se monitorea la utilización del storage local asociado al contenedor, pero se monitorean los volúmenes externos asociados.

Respaldos y recuperación

Es importante resaltar que los respaldos de los componentes de la aplicación que forman parte de la plataforma se encuentra incluidos en el plan de respaldo de la misma:

- Repositorios de código (como Git).
- Repositorios de imágenes.
- Configuración de los ambientes de la plataforma de contenedores (como ConfigMaps, Secrets).
- Volumenes de storage persistentes.

Se debe definir un plan de respaldo y recuperación para aquellos componentes que sean externos a la plataforma, como pueden ser:

- Bases de datos.
- Repositorios externos: de código, de imágenes, etc.
- Otros dispositivos de almacenamiento.

Es sumamente importante considerar que la información almacenada en un contenedor no es, por la naturaleza de la plataforma, persistente y en consecuencia no se respalda.

Mantenimiento

Rotación de logs

Se debe definir de manera temprana la retención de logs requerida para las aplicaciones, ya que esto impactará en los recursos asignados a la solución. Pueden dividirse en al menos dos categorías en función de la disponibilidad de los logs: “logs en línea” o “logs archivados”.

Aquellos que se guardan “en línea” están disponibles de manera inmediata para consultas y en general registran en tiempo real. Son más costosos de mantener y la retención usualmente está limitada a una cantidad fija de días que se puede ajustar en función de los recursos disponibles y las necesidades del negocio. Por otro lado los “logs archivados” permiten retener información por periodos más extensos (como ser años) pero habitualmente no se encuentran disponibles de manera inmediata, pudiendo ser necesario recuperarlos e indexarlos desde dispositivos de almacenamiento “lentos”.

Limpieza de BD

Se deberán aplicar los procesos para el mantenimiento de las bases datos de acuerdo a las prácticas recomendadas para el motor elegido y la naturaleza de la aplicación con el objetivo de asegurar la óptima performance y confiabilidad, esto independientemente de que las BD estén sobre la plataforma de contenedores o sea externas.

Elasticidad de los contenedores

La “Elasticidad” es una de las grandes ventajas de la plataforma, el término hace referencia a la capacidad de manejar de manera dinámica la cantidad de contenedores que atienden los servicios en función de la demanda del sistema. Esta habilidad de aumentar (escalar) o reducir (des escalar) la capacidad del sistema puede controlarse de manera manual o automática definiendo reglas. Estas reglas no son triviales y requieren de un proceso iterativo para lograr el comportamiento esperado en cuanto a performance dentro de los límites de recursos asignados al proyecto/aplicación.

Enfoque de calidad

En esta sección se especifican las actividades de calidad que deben ejecutarse durante del ciclo de desarrollo de software.

Definiremos los siguientes tipos de pruebas que se podrían ejecutar:

- Test unitario
- Análisis de código
- Pruebas funcionales manuales
- Pruebas funcionales automatizadas
- Pruebas de aceptación de usuario UAT
- Pruebas de performance
- Pruebas de integración
- Pruebas de deploy

Dependiendo del proyecto, algunas de estas pruebas pueden no ser necesarias y en otros podría ser necesario agregar nuevas. Las decisiones que se adopten deben justificarse y documentarse.

Ambiente	Compilación	Test Unitario	Análisis de Código	Build y Ensamble	Deploy	Pruebas de Humo	Pruebas Automatizadas	Pruebas Funcionales	Pruebas de Performance	Pruebas de Seguridad
Dev/Local	x	x	x	x						
CI/CD	x	x	x	x	x	x	x		x	x
Testing					x	x	x	x	x	x
Staging					x	x	x	x	x	
Integración Terceros					x	x	x	x	x	
Producción					x	x	x		x	x

	Ambientes manejados por desarrollo		Ambientes manejados por el organismo

Es necesario acompañar los requerimientos o tarjetas con sus respectivos criterios de aceptación los cuales orientarán las pruebas a realizar para validarlos.

A los efectos de que las pruebas sean más flexibles y fáciles de seguir por el equipo, recomendamos realizar test basado en sesiones diagramados en mapas mentales.

Test unitario

Un objetivo de calidad podría ser fijar la cobertura de pruebas unitarias que se tendrá para cada sprint. Se justificará la realización de dichas pruebas si la criticidad del proyecto lo amerita, por tanto será una decisión del equipo realizarlas o no.

Ejemplos de coberturas de pruebas unitarias:

- 100% de cobertura en módulos prioritarios y 80% en módulos no prioritarios.
- 100% de cobertura de métodos para módulos que se consideren de mayor criticidad y riesgo con 100% de cobertura de saltos
- 70% de cobertura de línea para módulos no críticos

Debe entenderse que el test unitario forma parte del desarrollo, de lo contrario tiende a considerarse opcional o poco prioritario lo que indefectiblemente termina en insignificantes niveles de cobertura. Es por esto que todas las estimaciones de cada desarrollo deben considerar el testing unitario.

Es necesario definir una estrategia para alcanzar los valores deseados de cobertura de test unitario para el producto y así poder planificar las actividades que deberán ser realizadas en cada sprint.

Para el caso de los proyectos desarrollados con tecnología java, una herramienta posible es JUnit o similares.

Análisis de código

Si bien es un atributo interno de calidad no visible al usuario, los problemas de calidad del código se transforman en visibles cuando afecta el esfuerzo necesario para realizar nuevas funcionalidades. Además suele haber una proporcionalidad entre los problemas de calidad del código y los defectos del producto.

El equipo de proyecto deberá definir las verificaciones apropiadas al mismo y los valores mínimos que se consideren aceptables.

Algunas de las principales verificaciones son:

- Código duplicado
- Ausencia de comentarios
- Complejidad ciclomática
- Tamaño de archivos de código

- Tamaño de métodos
- No adecuación a estándares y convenciones de código
- Vulnerabilidades conocidas de seguridad.
- Deuda técnica

El conjunto mínimo de verificaciones y valores aceptables podría ser:

- No existen issues nuevos catalogados como críticos o bloqueantes
- Cobertura de pruebas unitarias del código nuevo es 80%
- 10% de cantidad máxima de líneas duplicadas
- Ausencia de vulnerabilidades de seguridad

Es necesario definir una estrategia para alcanzar los valores deseados de calidad de código para el producto y así poder planificar las actividades que deberán ser realizadas en cada sprint. El análisis debe realizarse desde el ambiente de desarrollo para que cada desarrollador tenga conocimiento en todo momento de los niveles calidad del código aunque los indicadores formales se obtengan de un ambiente más estable como testing o stage.

Se sugiere utilizar la herramienta SonarQube o similares.

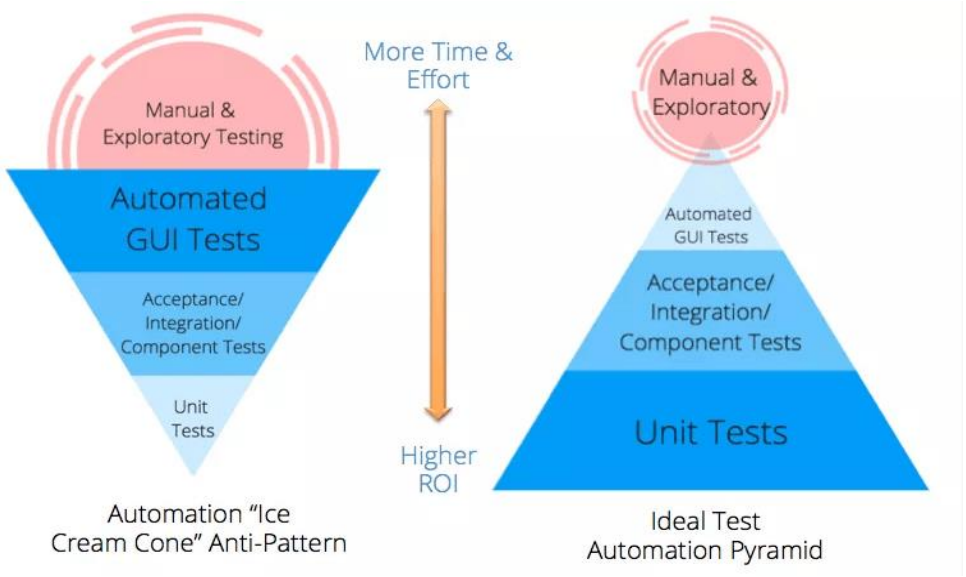
Pruebas funcionales

Los desarrollos deberán contar con pruebas funcionales realizadas por analistas de calidad de software. Los analistas de calidad forman parte del equipo de proyecto y se involucran con el desarrollo del producto desde las actividades de definición de requerimientos. Es importante que conozcan en detalle las funcionalidades y los aspectos técnicos utilizados en el desarrollo que consideren relevantes para la especificación y ejecución de las pruebas. Además deben tener presente toda la información vinculada a la gestión del proyecto como son la planificación de sprints, grado de avance de las actividades de desarrollo de las funcionalidades que deberán probar, alteraciones al cronograma y replanificaciones. Es responsabilidad del equipo definir la forma de trabajo y herramientas para que eso ocurra.

Las pruebas funcionales de cada requerimiento se realizarán de forma manual o automática dependiendo de diversos factores:

- Criticidad y riesgos.
- Esfuerzo de automatización y valor que la misma entrega.
- Esfuerzo y frecuencia de regresión.

El equipo de proyecto deberá definir qué pruebas automatiza siguiendo la estrategia de pruebas que se muestra a continuación:



Las decisiones adoptadas deben ser justificadas y documentadas.

Pruebas funcionales manuales

En los casos que el equipo de proyecto defina que algunos componentes o funcionalidades se verificarán con pruebas manuales, se utilizará una herramienta de gestión pruebas o Test Manager. La estrategia de gestión de pruebas deberá permitir establecer una correspondencia entre cada caso de prueba con la funcionalidad o requerimiento que verifica, el sprint en que fue desarrollado y el release a que corresponde.

Si se utiliza una estrategia de prueba guionada, la herramienta sugerida es Testlink o alguna otra que el equipo de proyecto considere apropiada, mientras la misma permita almacenar al menos lo detallado en esta sección.

El equipo de calidad agrupará los casos en test suites por el criterio que considere adecuado y definirá los test plan de acuerdo a las necesidades de cada sprint registrando los resultados obtenidos en cada build o release.

Si se utiliza una metodología de pruebas basada en sesiones, debe especificarse una forma de documentación de las mismas.

Pruebas funcionales automáticas

En los casos que el equipo de proyecto defina que algunos componentes o funcionalidades se verificarán con pruebas automáticas, es necesario establecer las herramientas que se utilizará. Se recomienda el uso herramientas como Selenium para las pruebas de interfaz preferentemente con un enfoque BDD utilizando Cucumber o similares. El patrón Page Object debe ser utilizado como elemento necesario para mejorar la escalabilidad y

mantenibilidad de las pruebas. Además debe utilizarse un enfoque Data Driven Testing para permitir que los mismos casos de pruebas puedan utilizarse con diversos datos para así reducir el esfuerzo necesario de aumentar la casuística y cobertura probada.

Las pruebas de servicios web pueden realizarse con SoapUI automatizando los assertions de cada testcase.

Debe considerarse las pruebas automatizadas igualmente que el código. Esto significa que debe ser versionado y revisado periódicamente para comprobar que sigue siendo adecuado, que no contiene malas prácticas y para identificar oportunidades de mejora.

Los resultados de las pruebas deben almacenarse con alguna herramienta de reporting que permita comparar las distintas ejecuciones pudiendo reportar de forma automática a la misma herramienta utilizada para las pruebas funcionales manuales.

Pruebas no funcionales

El equipo de proyecto debe considerar todos los requerimientos no funcionales y la forma en que serán probados. El ambiente para probar estos requerimientos debería ser testing o en su defecto stage. Todas las pruebas realizadas deben documentarse al igual que las pruebas funcionales. Puede utilizarse la misma herramienta de Test Manager o alguna otra que el equipo considere oportuna.

Dentro de estas pruebas se consideran las de accesibilidad, usabilidad, seguridad, entre otras. Algunas herramientas para estas pruebas son: W3C Markup Validation y OWASP ZAP.

Las pruebas de performance pueden considerarse como no funcionales y serán consideradas en otra sección del documento.

Pruebas UAT

Cada release debe contar con una prueba de aceptación de usuario. La misma no tiene como finalidad probar si la aplicación funciona correctamente porque eso debería comprobarse en las de más pruebas. Si un release llegó a una prueba de UAT es porque se conoce el grado de calidad de lo que se está entregando. El objetivo de esta prueba es validar el grado de adecuación entre lo solicitado y lo desarrollado. Estas pruebas deben realizarse en el ambiente de stage debido a su similitud con el ambiente de producción en recursos y datos.

Se espera que mínimamente la prueba de aceptación permita validar que la aplicación cumple al menos con las funcionalidades que implementan los flujos básicos.

Pruebas de performance

Las pruebas de performance tendrán al menos los siguientes objetivos:

- Detectar degradaciones del producto. Se recomienda la ejecución en ambientes de testing para que en caso de detectarse degradaciones, la versión no se promueva a los ambientes del cliente
- Verificar el cumplimiento de los requerimientos no funcionales vinculados a la performance del producto: los requerimientos deberían estar acordes a la capacidad del ambiente de producción. Por esa razón su ejecución debe ser en un ambiente de preproducción o stage con características similares a producción. Si el equipo considera necesario, puede ejecutarse un subset de casos como una pequeña muestra en el ambiente de test ya que el ambiente cuenta con recursos limitados a los efectos de detectar grandes degradaciones en el rendimiento.
  De todas formas definir requerimientos performance en un ambiente de testing no es recomendable porque condicionaría la comparación de resultados entre los ambientes a un análisis de proporcionalidades de recursos entre los mismos y su rendimiento

- Encontrar puntos de quiebre mediante stress del producto y análisis de endurance para conocer el comportamiento de la aplicación en tiempos prolongados

El equipo de proyecto debe definir qué pruebas de performance ejecutará en cada ambiente y en qué momento. Además se deben definir valores límite aceptables para cada prueba y ambiente utilizando promedios y percentiles que se consideren adecuados.

Los tipos de pruebas que deben considerarse son:

- Prueba de stress
- Prueba de carga
- Prueba de endurance
- Prueba de picos

La estrategia de pruebas debe considerar dos tipos de pruebas:

- Prueba de performance en pipeline: se busca que ejecute diariamente, en cada integración de código que tiene como objetivo encontrar degradaciones lo antes posible. Para esta prueba generalmente se ejecuta un subset de casos del total de la prueba de performance.
- Prueba de simulación de carga: prueba más completa buscando simular la carga esperada de la aplicación.

Se sugiere que las pruebas recolecten métricas de TPS (Transacciones Por Segundo), percentil 95 de tiempos de respuesta y porcentaje de errores para posteriormente comparar con datos históricos. Los valores límite usualmente se establecen de forma de aceptar una desviación de un 20% en los valores promedio. Esto significa que el 95% de los tiempos obtenidos no deben superar en 20% el valor promedio.

Es importante contar con un informe detallado sobre los resultados de las pruebas de performance, debiendo almacenarse de forma que se permita la comparación entre las

mismas, lo cual se utilizará como insumo para visualizar las variaciones de rendimientos entre versiones.

En lo que refiere a herramientas, algunas recomendadas son Jmeter y Gatling. El equipo de proyecto definirá las herramientas más apropiadas dependiendo de la tecnología utilizada.

Pruebas de deploy

Las pruebas de deploy tienen como finalidad comprobar que una versión de la aplicación fue desplegada de forma correcta, principalmente en el ambiente de producción. Si bien algunas de estas pruebas pueden ser manuales, se considera necesario automatizar la mayor cantidad posible de las mismas.

Estas pruebas contienen un conjunto de verificaciones funcionales, consideradas de humo, y no funcionales siendo importante que el equipo de proyecto revise las mismas en cada sprint y las actualice si considera necesario.

Si bien puede aplicarse al deploy en cualquier ambiente, deben mitigar los principales riesgos que tiene la instalación de la nueva versión en el ambiente de producción.

Pruebas de integración

El objetivo de las pruebas de integración es verificar el correcto ensamblaje entre los distintos componentes una vez que han sido probados unitariamente con el fin de comprobar que interactúan correctamente a través de sus interfaces, tanto internas como externas, cubren la funcionalidad establecida y se ajustan a los requisitos no funcionales especificados en las verificaciones correspondientes.

Pruebas, ambientes y pipelines

El equipo de proyecto debe definir una estrategia de ejecución de las pruebas determinando el momento y ambiente en que se ejecutará cada prueba, y construir los pipelines necesarios para cumplir la misma.

Esta estrategia debe estar en concordancia con las sugerencias y metodología mencionadas en la siguiente sección del documento.

Deuda Técnica

La deuda técnica es un concepto en el desarrollo de software que refleja el costo implícito del retrabajo adicional causado por elegir una solución fácil en lugar de utilizar un enfoque que llevaría más tiempo en su desarrollo e implementación.

La metáfora de la “Deuda Técnica” aplicada al desarrollo software la introdujo hace dos décadas Ward Cunningham para explicar a los “no técnicos” la necesidad de refactorizar el código fuente.

Desde entonces, la deuda técnica se ha utilizado para describir muchos otros tipos de deudas o males del desarrollo de software, y se ha aplicado a cualquier cosa que aumente innecesariamente este tipo de esfuerzos, se interponga en la futura evolución o venta de un sistema de software. Hoy podemos encontrar la deuda de las pruebas, la deuda de las personas, la deuda de la arquitectura, la deuda de los requisitos y la deuda de la documentación entre otras.

Para profundizar en el tema, ver en detalle los tipos de deuda técnica y la forma de gestionarla recomendamos la lectura del documento **Introducción al concepto Deuda Técnica.**

Enfoque para Integración Continua

En este apartado detallamos las principales actividades que involucra la puesta en marcha del ciclo de integración y entrega continua. Estas actividades serán responsabilidad principal del “Responsable de Automatismos”, más allá de que este rol sea tercerizado o no.

Los aspectos aquí descritos tienen suma importancia en la concreción de los objetivos de calidad del proyecto y en el aumento paulatino de los niveles de garantía con los que el producto puede llegar a producción. Los automatismos constituyen el habilitador principal del ciclo de integración continua y como tal, deben ser abordados desde el inicio del proyecto, acompañando cada uno de los incrementos del mismo, manteniendo, mejorando y aumentando la base de actividades automáticas que brindan cobertura al proceso de desarrollo.

Ambientes

En cada proyecto será necesario analizar cuál será la mejor configuración de ambientes a fin de alcanzar los objetivos de calidad establecidos. Es fundamental poder realizar una correcta gestión de los ambientes para que las pruebas se ejecuten en un contexto conocido, asegurando así que los resultados sean predecibles y se puedan validar.

A continuación, presentamos una recomendación de los ambientes a considerar, tomando

en cuenta principalmente el objetivo de cada uno de ellos más allá de los nombres, que pueden variar según las características de cada proyecto. Recomendamos entonces, tener en cuenta la creación de los siguientes ambientes:

- **Desarrollo**: El ambiente más inestable, el que utilizan los desarrolladores para pruebas puntuales. Es local a cada desarrollador, a no ser que se requieran integraciones complejas con determinados componentes. En muchas ocasiones puede ser la máquina de cada desarrollador.

- **Integración Continua**: En este ambiente se realiza el build y se ejecutarán las pruebas y chequeos automatizados desde el motor de CI/CD, tanto a nivel funcional, seguridad, accesibilidad, performance, etc. Es en este ambiente donde los desarrolladores prueban en forma integrada su trabajo, analizando los resultados de las actividades automáticas como insumo principal para el cumplimiento de los objetivos de calidad y evolución del producto.

- **Testing**: Ambiente desde el cual se promueven los cambios de manera controlada, sólo si pasaron los chequeos automatizados. En este ambiente se realizan las pruebas funcionales exploratorias. El equipo de desarrollo promueve el incremento a este ambiente con el propósito de que el equipo de QA lleve adelante las pruebas necesarias.ver temas de espacio para imágenes.

- **Staging**: En este ambiente se llevan adelante las pruebas de aceptación de la versión. Se lo conoce también como UAT (User Acceptance Testing) o Pre-producción. Por lo general tiene datos que fueron obtenidos de producción, con el objetivo de probar el incremento en condiciones lo más similares posibles al entorno productivo en el que ejecutará el software. Muchas veces se utiliza este ambiente para las pruebas de performance de aceptación (simulando una carga esperada), en el caso que no se haya definido un ambiente separado para este fin.

- **Integración**: Este ambiente tiene por objetivo servir como ambiente de pruebas a los organismos, por lo cual debe estar conectados con todos los servicios externos que la aplicación requiera para ejecutarse adecuadamente. Al mismo tiempo, es un ambiente que permite un acercamiento distinto en lo que respecta a las pruebas, ya que el mismo cuenta con carga real de usuarios.

- **Producción**: Sistema accesible para los usuarios, ambiente productivo. Controlado y restringido, prioridad máxima. No se suele utilizar para pruebas a menos que esté claramente controlado y focalizado el impacto que puedan ocasionar.

Es altamente recomendable que las pruebas de aceptación se lleven adelante en un ambiente controlado y separado del resto. Es allí donde el ambiente de Staging marca una diferencia importante en los controles de calidad profesionales, y es fundamental su adecuada gestión. En ciertas ocasiones se puede, con criterio y fundamentación, reducir la cantidad de ambientes, de acuerdo al riesgo del proyecto o a los recursos de hardware y software con los que se cuenta, entre otros.

Actividades por Ambiente

En la tabla que se muestra en el apartado "Enfoque de Calidad", se recomienda el conjunto de actividades a ejecutar en cada uno de los ambientes según los objetivos descritos para cada uno de ellos en el apartado anterior:

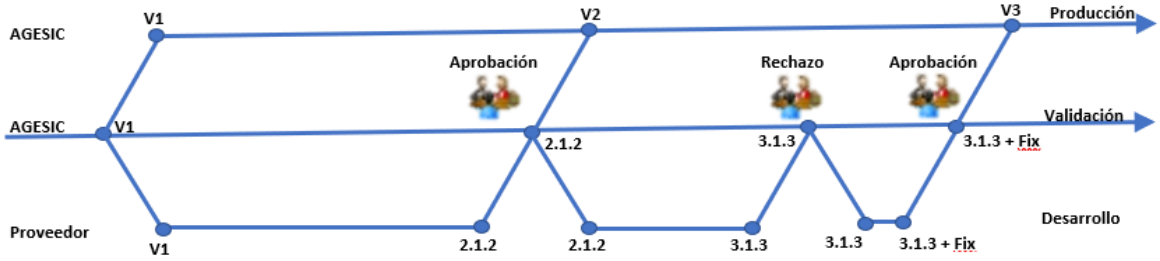
Promoción y Aprobación

Uno de los beneficios que trae aparejados la aplicación de prácticas de CI/CD es la detección de errores en forma temprana durante el ciclo de desarrollo. Como consecuencia, a medida de que los incrementos del producto "avanzan" de un ambiente a otro, se va ganando certeza sobre el correcto funcionamiento del mismo.

Los automatismos juegan un papel más que importante en este aumento del grado de certeza y confianza en el producto, sin embargo, no pueden ser el único elemento sobre el cual depositar todas nuestras esperanzas, sobre todo considerando el hecho de que es poco eficiente automatizar por completo las actividades repetitivas. Es aquí donde toma especial relevancia el compromiso del equipo del proyecto, tomando control sobre la calidad del producto y el cumplimiento de las expectativas en sus atributos de calidad de arquitectura.

Sugerimos entonces establecer una revisión formal de cada uno de los incrementos desde que son entregados por el equipo de desarrollo en el ambiente de Staging, hasta su llegada al ambiente productivo. Esta actividad consiste en analizar los resultados de las pruebas automatizadas incluidas en cada uno de los ambientes, así como también obtener los resultados de las pruebas manuales (del carácter que sean) sobre cada uno de los entregables.

Presentamos un ejemplo de la intervención del equipo en el flujo de aprobación para los incrementos:



Teniendo en cuenta el flujo de trabajo anterior, es que recomendamos en los ambientes de staging en adelante, la promoción se lleve a cabo en forma manual, ya que las actividades necesarias para aprobar los incrementos pueden requerir tiempos variados dependiendo del momento del proyecto y otros factores externos que pueden influir.

El flujo de aprobación, la gestión de los ambientes y los automatismos incluidos en cada uno de ellos, conjuntamente con el manejo del repositorio de código, conforman una trilogía que constituye la piedra fundamental para aprovechar al máximo los beneficios de las prácticas de CI/CD, contribuyendo así al logro de los objetivos del proyecto. Seguidamente incluimos una propuesta de promoción de ambientes que acompaña el flujo de aprobación definido anteriormente.

Como puede verse en el esquema anterior, el despliegue hasta Staging puede realizarse en forma automática, mientras que de Staging en adelante, recomendamos hacerlo en forma manual, al menos hasta alcanzar la madurez necesaria para promover de forma automática los incrementos a producción. Esta aproximación permite al equipo adaptar su forma de trabajo, ajustando sistemáticamente e incorporando mejoras al flujo a medida que avanza el proyecto y se evalúan los resultados.

Automatismos

Los automatismos son un habilitador imprescindible para aplicar las prácticas de CI/CD en el ciclo de desarrollo. Al mismo tiempo Openshift Container Platform (en adelante OCP), facilita la implementación no solamente de los automatismos, sino también del flujo de promoción y aprobación del producto hasta su llegada a producción.

La implementación de las actividades que constituyen el ciclo de integración y entrega continua, deben estar alineadas con las definiciones tomadas respecto al manejo del repositorio de código y las ceremonias involucradas en la promoción del producto. Esta concordancia entre todos los elementos involucrados, en ocasiones no es fácil de alcanzar, por lo cual recomendamos construir los automatismos con un enfoque incremental, agregando automatismos de forma paulatina, acompañando los incrementos del producto en cada sprint.

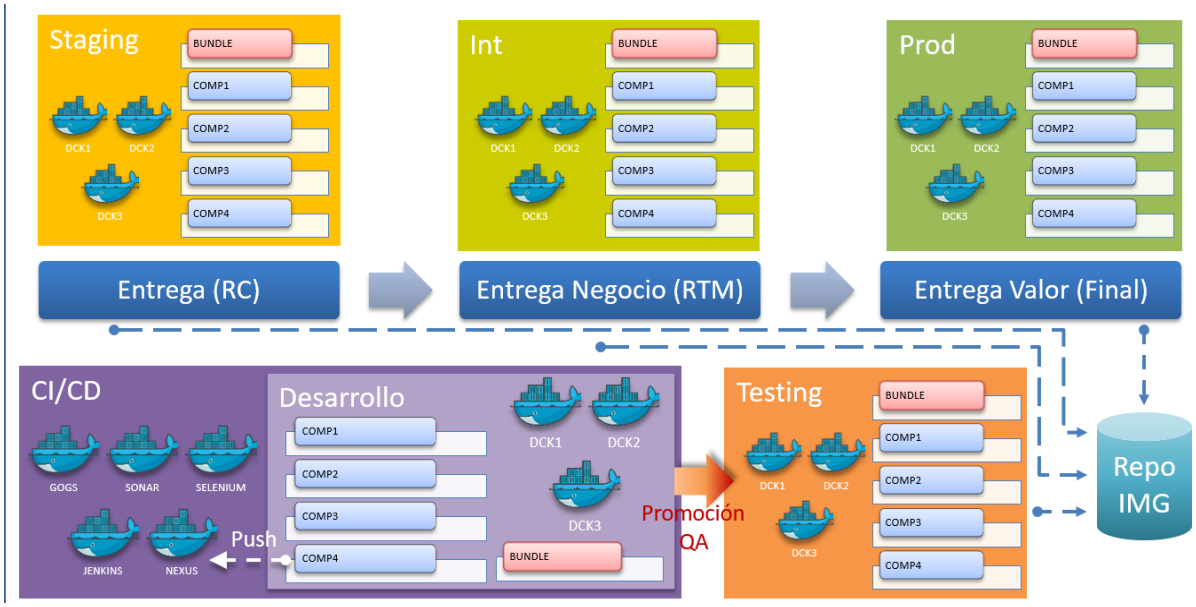
Un espacio adecuado para intercambiar ideas y definir puntos de avance, es la reunión técnica. En la misma, pueden abordarse distintos aspectos técnicos de la solución, como ser Arquitectura, Calidad y Automatismos. Este enfoque multidisciplinario, permite alinear los incrementos en automatismos principalmente con los objetivos de calidad y al mismo tiempo con los atributos de calidad que guían el diseño arquitectónico y los aspectos relacionados con la operativa del producto una vez promovido a producción.

En la sección siguiente, presentamos un esquema de automatización que puede adaptarse a las definiciones previas sobre el ciclo de promoción y aprobación y el manejo del repositorio de código.

Implementación de CI/CD

El esquema que sigue, muestra una posible implementación de CI/CD teniendo en cuenta las actividades comprendidas en cada uno de ellos y el ciclo de promoción y aprobación definido anteriormente.

Detallaremos aquí los elementos y actividades que componen cada etapa y la forma en la que esta aproximación a CI/CD sustenta aspectos de gestión y operación de la solución en etapas posteriores del proyecto. Cada ambiente representado en el esquema se implementa mediante un proyecto de OCP que estará compuesto por distintas imágenes de contenedor, según el cometido de cada uno de los ambientes.



Herramientas para facilitar su adopción

Repositorio de Imágenes y Manejo de Tags

El flujo de promoción se refleja en el pasaje de las imágenes de contenedores que representan cada uno de los componentes que tendrá la aplicación en tiempo de ejecución. A medida que las imágenes van superando los controles de calidad (sean estos automáticos o manuales), impuestos en cada ambiente, las mismas se van marcando con distintos tags que reflejan su promoción de un ambiente a otro, hasta llegar a producción.

Recomendamos utilizar una nomenclatura de tag y número de versión para promover los incrementos que podría ser de la forma “TAG-#.#.#”, dónde cada “#”, representa versión, versión menor y revisión respectivamente. La correspondencia de cada uno de los tags con los ambientes se detalla a continuación:

Ambiente	TAG
Desarrollo	Snapshot
QA	Beta
Staging	Release Candidate (RC)
Integración	Release to Market (RTM)
Producción	Final

Esta forma de promover las soluciones, implica que se contará con el histórico de imágenes promovidas (sea cual sea su ambiente), dentro del repositorio de imágenes de OCP. Esto permite desplegar de manera sencilla cualquier versión del producto en cualquiera de los ambientes.

Por otro lado, tiene el beneficio de evitar posibles errores humanos en el despliegue, ya que el personal de operaciones sólo podrá ver en cada ambiente las imágenes marcadas con el tag que corresponde a cada uno de ellos.

Flujo de Trabajo por Ambiente

Ambiente de Integración Contínua

Este es el ambiente que cuenta con toda la infraestructura necesaria para posibilitar la ejecución de las actividades de CI/CD. En ella se encuentran los elementos que ejecutan las actividades automáticas. Entre otros elementos, este ambiente puede contener:

- Servidores de integración
- Repositorios de artefactos
- Analizadores de código
- Ejecutores de pruebas funcionales
- Ejecutores de pruebas de Interfaz de usuario
- Ejecutores de pruebas de seguridad

Este ambiente cuenta con varios Pipelines, uno por cada componente del proyecto (o uno por unidad de prueba), los cuales culminan almacenando en el repositorio de artefactos.

Esto facilita la construcción de la imagen de contenedor ya que los artefactos están listos para ensamblar a la imagen.

Por otro lado, es recomendable contar con un Pipeline de “bundle”, que ejecuta los pipelines de componente en el orden necesario para generar las imágenes y promoverlas al siguiente ambiente. Este pipeline “bundle”, es de especial utilidad para el equipo de Operaciones, ya que lo deslinda del conocimiento interno de la solución y ejecuta las tareas necesarias para dejar disponible las imágenes en el ambiente siguiente.

Otro aspecto importante a considerar, es que el pipeline de “bundle” ejecuta todas las actividades de calidad necesarias para promover el producto en cada ambiente. Esto se lleva adelante invocando todas las actividades de los pipelines individuales de cada componente, en el orden correspondiente para lograr una versión desplegable de la solución. Es decir que, sólo a través del este pipeline es posible promover los incrementos. Esto toma especial relevancia si consideramos que esta mecánica debe respetar el flujo de promoción y aprobación definido.

Este pipeline debe ofrecer las garantías mínimas para promover el producto en cada uno de los ambientes, desde staging a producción, considerando especialmente los objetivos de calidad. Por este motivo, es posible considerar la opción de contar únicamente con los pipelines de bundle en los ambientes de Staging hacia adelante, para reforzar el hecho de que se ejecuten las actividades mínimas necesarias para promover la solución completa, sin contar con la posibilidad de desplegar componentes de forma independiente.

Al ser el ambiente en el que se llevan adelante las tareas automáticas, es sumamente importante considerar el almacenamiento de reportes de resultados y la retención del histórico de los mismos durante el proyecto. Los reportes son el principal insumo para conocer los aspectos que habilitan o impiden la promoción de un incremento.

Contar con un adecuado manejo de reportes, no solamente posibilita la ejecución del flujo de promoción y aprobación, sino que además contar con el histórico de los mismos, permite tomar acciones proactivas para adecuar paulatinamente distintos aspectos del proyecto según los objetivos que el mismo persigue.

Ambiente de Desarrollo

Si bien es un ambiente independiente, alojado en un proyecto individual de OCP, este ambiente tiene una relación extremadamente cercana con el ambiente de Integración Continua. De hecho, podríamos decir que es el ambiente de “Prueba de Integración Contínua” ya que es el utilizado por los desarrolladores para probar los elementos de software durante su desarrollo haciéndolos pasar por el ciclo de integración continua.

De lo anterior, es fácil advertir que es el ambiente más inestable de todos y es el que requiere contar con distintos pipelines para cada unidad de prueba. Esto se debe a la implementación de integración continua debe permitir el desarrollo concurrente de los distintos elementos de software que componen la solución. Adicionalmente, este ambiente puede contar también con un pipeline de “bundle” para que el equipo de desarrollo lo utilice al momento de promover la solución al ambiente de QA.

En este ambiente se encuentran las actividades automáticas “de caja blanca” que analizan la estructura interna del código. Ejemplo de esto son las herramientas de análisis estático de código y la ejecución de testing unitario de cada una de las unidades de prueba. Adicionalmente, pueden incluirse otros tipos de prueba con el fin de detectar desvíos en forma temprana como ser pruebas de performance “livianas”, un subconjunto reducido de pruebas de seguridad y algunas pruebas funcionales que sirvan al equipo de desarrollo como garantía funcional al momento de promover el incremento a QA.

El equipo de desarrollo es el responsable de promover el producto completo al ambiente de QA una vez que todas las unidades de prueba han alcanzado los niveles de garantía deseados. Una vez que se promueve el incremento, se marca la versión SNAPSHOT de la imagen con el tag “BETA”.

Ambiente de Testing

Este ambiente es en el que el equipo de Calidad realiza las pruebas que le permiten determinar si el incremento se encuentra en condiciones de ser promovido al ambiente de Staging para que se inicie el flujo de promoción y aprobación. Estas pruebas pueden ser manuales, automáticas o una combinación de ambas.

Los automatismos en este ambiente tienen un enfoque netamente funcional y debe tomar como insumo el resultado de las actividades de calidad del ambiente de Desarrollo ya que el equipo de calidad debe actuar como garante en lo que refiere al cumplimiento de los objetivos de calidad del proyecto y en particular a los objetivos asignados al incremento bajo prueba. Estos pueden ser métricas de cobertura de prueba unitaria, “code-smells” o vulnerabilidades de seguridad encontradas en el análisis estático u otros elementos relacionados con los resultados de las actividades de calidad de desarrollo.

Ambiente de Staging

El equipo de Calidad mediante el uso del pipeline de bundle es el que promueve el incremento al ambiente de Staging. Este es el primer ambiente fuera del entorno del equipo de desarrollo y como tal, es aquí donde se inicia el ciclo de promoción y aprobación.

Es conveniente que el equipo de Calidad ejecute al momento de promover el incremento, al menos las mismas pruebas automáticas que se ejecutan en este ambiente a fin de evitar los “rebotes” innecesarios de los incrementos entre el último ambiente manejado por el equipo de desarrollo (Testing) y el primer ambiente manejado por el negocio (Staging).

Es recomendable que este ambiente tenga las condiciones de entorno lo más similares posible al ambiente de Producción, incluso clonando la Base de datos (si la hubiera) de producción para mejorar los niveles de garantía que pueden aportar las actividades de calidad.

Adicionalmente, es conveniente ejecutar en este ambiente, pruebas de performance y seguridad al incremento ya que es Staging el único ambiente sin carácter productivo ya que no cuenta con carga de usuarios reales, siendo su único propósito la aceptación de los incrementos entregados por el equipo de desarrollo. Es por esta razón que resulta más sencillo realizar pruebas de performance, penetración, interfaz de usuario, accesibilidad, usabilidad u otras sobre este ambiente, ya que no estamos afectando a ningún usuario ejecutando el incremento sobre este ambiente.

Al momento de recibir el entregable de parte del equipo de desarrollo, deben revisarse los reportes de las actividades automáticas y los resultados de pruebas manuales a fin de determinar si se cumplen los criterios acordados. De esta forma, evitamos iniciar actividades de aprobación y promoción si los objetivos de calidad no se cumplen.

Una vez finalizadas las actividades automáticas y pruebas manuales definidas por el negocio, el equipo de proyecto vuelve a revisar los reportes de las actividades a fin de determinar si es posible promover o no el incremento. Una vez definida la promoción del mismo, el responsable definido por el negocio promueve el incremento utilizando el pipeline de bundle, marcado como “BETA”, marcándolo con el tag “RC” (Release Candidate).

Ambiente de Integración

Al ser este un ambiente de integración de terceros, es el primero que tiene carga real de usuarios ya que los organismos utilizan este ambiente para pruebas de sus propios desarrollos e integraciones. De esta manera, este es el primer ambiente que se encuentra integrado con todos los componentes externos que conforman la solución completa. Usualmente los componentes externos que utiliza este ambiente no son productivos, sino de prueba.

De lo anterior se desprende la relevancia de este ambiente en cuanto a las garantías adicionales que puede aportar. En este entorno, las pruebas automáticas se enfrentan a una base de datos (si existe) ejecutando transacciones, componentes externos activos y una carga más representativa que la del ambiente anterior. Desde este punto de vista, podemos inferir que aunque ejecutemos el mismo conjunto de pruebas que en el ambiente anterior, al cambiar el entorno de ejecución, podemos obtener información adicional de relevancia sobre el comportamiento del incremento.

Es posible además, ejecutar pruebas de aceptación manuales o pruebas de interfaz de usuario completas ya que es en este ambiente en el que contamos con todos los servicios externos conectados. En otro sentido, es importante coordinar las actividades a realizar sobre este ambiente con el equipo de operaciones ya que es posible que tenga SLAs productivos por ser utilizado como ambiente de prueba por otros organismos.

Una vez finalizadas las actividades definidas para este ambiente, el responsable definido por el equipo, promueve el incremento mediante el uso del pipeline de bundle, marcando la versión “RC” con el tag “RTM” (Release to Market).

Ambiente de Producción

Una vez cumplidas satisfactoriamente las actividades previstas para el ambiente de Integración de terceros, están las condiciones dadas para promover el incremento a producción.

Usualmente, en este ambiente no se ejecutan pruebas, no se cuenta con pipelines de componentes, siendo posible contar con un pipeline de bundle con el único propósito de ejecutar de forma rutinaria y en intervalos definidos pruebas de performance, seguridad u otras destinadas a obtener información del desempeño del producto sobre el ambiente productivo.

Un aspecto importante a tener en cuenta es la definición del ciclo de hotfix, destinado a resolver incidentes graves sobre el ambiente de producción utilizando un flujo de promoción y aprobación especial y usualmente mucho más rápido, con el objetivo de obtener las garantías necesarias e implantar los arreglos de forma rápida en producción. Este ciclo tiene implicancias en el manejo de repositorio de código y la información requerida por el equipo para este tipo de situaciones especiales, es por eso que recomendamos definir el proceso con anterioridad a su ocurrencia.

Acceder a referencias