SOLUCIÓN PRUEBA TÉCNICA | LÍDER DE DESARROLLO TÉCNICO

F2X | FLYPASS

Presentado por: Daniel Ramírez Bedoya - CC 1128271637 - danielramirez738@gmail.com
Fecha de entrega: 01 de Enero 2026

ENTREGABLE A

Definición de KPIs accionables para el área de desarrollo

Premisas de KPI

- Medibles
- Verificable
- Accionadles
- Claros
- Relevante para la gerencia de TI

KPIs

Lead Time

Tiempo desde que llegar la HU priorizadas para su ejecución, hasta que llegue a PRD
Se puede calcular desde Fecha de PRD - Fecha de Asignación de la HU
Para una correcta ejecución del KPI se esperan estrategia de automatización de pruebas y despliegues, minimizar pruebas manuales.

Aseguramiento de la calidad

La ejecución de pruebas y sus tiempos de ejecución.
Posterior a las pruebas que satisfacción encontramos en el negocio. El negocio también percibe la calidad de lo que se construye.
Su medición parte de las pruebas ejecutadas VS la satisfacción de negocio.

Tamaños de cambios Pull Request

Los pull request por evaluar deben ser features pequeños, entendibles para cualquier persona del equipo técnico.
No se debe considerar full Request con múltiples funcionalidades.

Con base a esto se evaluar cantidad de pull Request solicitados Vs aprobados.

Tasas de errores
Dentro del desarrollo es importante mitigar posibles errores desde la etapa de codificación.
Con herramientas de tácticas es posible mitigar que el volumen de errores vaya en aumento.
Dicho KPI propone por cada iteración y delivery evaluar la cantidad de errores identificados

Es posible evaluar cantidad de errores en ambiente de QA VS errores en ambiente PRD

Frecuencia de salida a producción (Delivery)
Partiendo de un escenario de cumplimiento de estándares de desarrollo y ejecución satisfactoria de pruebas, con que periodicidad se esta entregando valor a negocio bajo nuevas funcionalidades
Se puede calcular desde evaluando la cantidad de despliegues satisfactorios de cada iteración ejecutada (Sprints)

Gestión de la deuda técnica
Se debe garantizar la sostenibilidad de los productos construidos, mitigando la deuda técnica identificada.
La deuda técnica se puede estimar por horas que llevaría su total cierre.
El crecimiento de las deuda técnica por iteración debe ser mínimo.
Un alto crecimiento de la deuda técnica en horas, con relación a la capacidad del equipo es una alerta de que debe ser priorizado su mitigación.

Tiempos de codificación
Se parte de un punto pivote de una HU básica de desarrollo
Con base a esto se estima las HU bajo serie fibonacci (como propuesta)
A partir de esto se estiman las HUs asignadas

Ejemplo:
HU de desarrollo funcionalidad básica: 1 punto = 2 horas de desarrollo
Serie fibonacci: 1, 2, 3, 5, 8, 13 (una HU de mas de 13 puntos estimados debería considerarse un proyecto o factibilidad)

Se sugiere que el punto pivote sea acordado con los integrantes del equipo.
Bajo esta estimación, se puede calcular Horas estimadas de cada HU con relación al tiempo ejecutado de desarrollo
La estimación de los tiempos se sugiere que los desarrolladores considerar tiempo de "incertidumbre" para su trabajo.
La variación de los tiempos de desarrollo con relación a los estimados no deben superar 10% más de tiempo, en este caso no fue una estimación satisfactoria.

Cumplimiento de features según necesidad de negocio
Si entendemos bien la necesidad, entendemos bien que debemos construir.
Es importante desde etapas tempranas identificar la necesidad correcta, los riesgos y restricciones posibles.
Negocio también percibe la calidad del producto.
Dado esto será de gran probabilidad que lo que construyamos si es lo que el negocio requiere.
La satisfacción de los features por parte de negocio es la mayor variable de medición de este KPI.

Thoughput de equipo
La capacidad del equipo debe evaluarse de manera real.
Llevar un Backlog de necesidades de negocio para la ejecución del equipo.
Sobre el Backlog realizar un ejercicio consciente de priorización con el negocio.
Con base a esto se crear los HU priorizadlas para asignar.
La asignación debe tener en cuenta varios factores:
- Conocimiento técnico del desarrollador y expertis
- Asignaciones actuales
- La asignación del desarrollador no se realiza sobre el 100% de su tiempo. Se estima sobre un 70% a un 80% de su capacidad.

Calcular sobre las HU asignadas y estimadas, HU entregadas de manera satisfactoria.

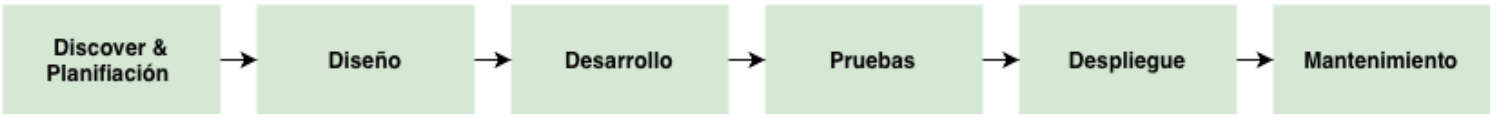
Cumplimiento plan de formación
Los planes de formación requieren una planificación y un posible presupuesto para ejecutar.
Es importante medir la ejecución del plan (porcentaje ejecutado, satisfacción de los colaboradores, etc)

Importante!

- Bajo la definición de los KPI, se sugiere iniciar con la ejecución y medición de **3 a 5 KPI**, los más relevantes para el área y gerencia de TI
- Estos KPI deberán ser veirificables y garantizar que se cumplan de manera satisfactoria
- Posterior a esto involucrar los siguientes KPI definidos.

Procesos SDLC

Etapas de proceso



Ejecución Propuesta

Proceso	Actividad	Metricas	Artefactos
Discover & Planeación	-Identificación de req necesidad -Análisis funcional & técnico -Identificación de posibles riesgos -Estimación de alto nivel -Asignación según la criticada y capacidad de equipo	-Tiempos promedios de estimación - % de riegos identificados - % de requerimientos a probados para su ejecución VS rechazados por alcance o falta de info (Priorización)	- Epicas - Historias de usuario - Lista de requerimientos - Estimación espire técnico
Diseño	- Diseño de arquitectura de alto nivel - Definición y validación de criterios de aceptación	- Tiempo de reproceso por criterios de aceptación, alcance o entendimiento funcional - Incidentes por HU mas definidas	- Criterios de aceptación - ADR (en caso que aplique con trabajo conjunto con arquitecura) - Casos de prueba definidos

Proceso	Actividad	Metricas	Artefactos
Desarrollo	- Codificación - Test unitarios - Code review - Pull Request	- % PR rechazados - Evaluación de revisión de estándares de desarrollo	- Versionamiento de código generado - Checklist de revisión de desarrollo - Test automatizados - Repos de commits ejecutados - Documentacion de desarrollo
Pruebas	- Ejecución de pruebas automatizadas - Ejecución Validación de área de QA - Go técnico	- Test ejecutados satisfactoriamente - Cantidad de errores identificados - Tiempos de ejecución de Pipelines de QA	- Evidencias checklist de pruebas ejecutadas - Reporte sobre errores identificados y tiempos de refinamiento
Despliegue	- Preparación para el release - Ejecución pipermines para ambiente de PRD - Monitoreo de despliegues y comportamiento de entorno	- Frecuencia de despliegues - Tiempo estimado de discover hasta salida a PRD (Leadtime) - Cantidad de rollbaks ejecutados	- Metricas de pipelines ejecutados - Notas ya apuntes al seguimiento del despliegue - Plan de rollbacks
Operación y Mantenimiento	- Monitoreo continuo - Feedback de lo ejecutado - Gestión de incidentes o deudas técnicas pendientes - Ajustes al proceso SDLC si es requerido - Lecciones aprendidas y refinamiento	- Indícentes identificados - Gestión de deuda tecnica (aumento o disminuye con relación a las anteriores iteraciones) - Porcentaje de Cumplimiento a negocio	- Dashboard de consolidación de proceso de SDLC - Backlog técnico ejecutados y por ejecutar - Backlog de deuda tecnica

Consideraciones

- EL anterior proceso SDLC se propone desde la experiencia , marcos de trabajo y buenas practicas de la gestión de desarrollo de software
- Todo proceso es sensible a cambios en sus actividades, métricas y artefactos, se busca lo que mejor funcione para el equipo de trabajo, la gerencia de TI y negocio
- El proceso SDLC se propone desde una visión to-be, con equipos de QA , arquitectura y practicas de automatización de despliegue, todo bajo un esquema de trabajo colaborativo.
- El proceso propuesto es totalmente ejecutable en escenario ágiles o tradicionales. (Se sugiere considerar esquemas de ágiles de trabajo)

Gobierno del Delivery

Proceso de gobierno adaptable al negocio

Roles y responsabilidades claras:

- **Product owner:** Definición de la necesidad, valor al negocio y priorización
- **Lider de desarrollo:** negociación, asignación según capacidad, decisiones técnicas y acompañamiento, responsable de garantizar que el equipo de desarrollo cuente con lo necesario para que todo fluya
- **Developers**: Construcción y buenas prácticas
- **Arquitectura**: Diseño y coherencia con la visión técnica
- **QAs:** Calidad desde el inicio y en todo momento.
- **DevOps y plataforma:** confiabilidad y automatización

Estándares técnicos obligatorios: linea base de desarrollo, que se garantice su ejecución:

- Convenciones técnicas
- Estrategia de ramificación, commits y Full Request
- Estándares de seguridad, por ambientes de trabajo - DEV - QA - PRD
- Estrategia de testing: Unitarias - funcionales - integrales
- Practicas de code review y Par review
- Observabilidad: logs, metricas, trazas
- Pipelines y practicas de automatización

Aseguramiento de la calidad: Pruebas automatizadas, herramientas de Code review.

Cadencia y seguimiento: acompañamiento constante, feedback de equipo y personal, ajustes increméntales y medibles. Iterativo e incremental

Estrategia de transferencia de conocimiento

Se proponen actividades iniciales para conocer el estado actual de las personas del equipo de trabajo, tales como:

- Definición de matriz de capacidades técnicas y habilidades por niveles (Junior - semisenior - senior)
- -Autoevaluación de competencias para integrantes de equipo de trabajo - individual
- -Evaluación de intereses de integrantes de equipo, desde el ser y del hacer - individual
- -Identificación de capacidades requeridas para habilitar dentro de equipo y gerencia TI

A partir de esto se tienen datos para conocer intereses y percepciones de las personas.

Plan de mentoría interna: Se crean espacios de formación, donde las personas de nivel mas experto puedan capacitar a las personas de nivel Junior, espacios prácticos. En estos espacios se puede contemplar de invitación abierta para la compañía, con el fin de identificar colaboradores de otras áreas interesados por lo que viene haciendo la Gerencia de TI y equipos de desarrollo; Adicional genera visibilidad del área. (Ejemplo: cada 8, 15 días, viernes 1, 2 horas de formación)

Capacitación: Programas, plataformas y alianzas con partner técnicos, buscando beneficios de formación, voucher, congresos entre otras actividades. Definir un presupuesto anual de formación, apuntando a lo requerido por el área y tendencias tecnológicas, partiendo de las actividades de exploración propuestas en el primer punto.

Se sugiere que los planes capacitación sea enfocado en temas específicos y prácticos, y de manera continua, en lugar de concentrar el esfuerzo e una única capacitación de gran alcance.

Documentación estandarizada: Definir plantillas, manuales, plataformas internas para la gestión de documentación.

Sistema de gestión de conocimiento: plataformas interactivas que permiten la formación y preservar el conocimiento del área

Cultura: fomentar una cultura de aprendizaje continuo, ambiente que permita el intercambio de conocimiento y valore el desarrollo de las competencias internas.

Es importante considerar que para mitigar posibles dependencias de gestión de conocimiento con aliados externos o proveedores, se deben promover actividades como:

- Solicitar documentación técnicas y entrega de la misma
- En etapa de desarrollo, implementar actividades de pair programming (construyen, nosotros observamos, indagamos y aprendemos…)
- Crear espacios de transferencia de conocimiento, autorizando grabaciones como material de formación.

ONBOARDING & OFFBOARDING

Onboarding

El onboarding debe considerarse tanto para personas internas como externas

- Contexto inicial del objetivo del área, gerencia TI y negocio.
- Invitación a espacios y ceremonias de equipo de trabajo y presentación a equipo
- Presentación de SDLC definido.
- Presentación de buenas practicas definidas, estandares, artefactos y entregables dentro del equipo
- Espacios de autoestudio bajo plataforma de gestión de conocimiento y documentación e equipo.
- Participación en sesiones como espectador
- Acompañar las actividades propuestas desde el área de G.H

Se espera que la productividad real de la persona inicie en los primeros 20 días.

Posterior a esto se propone “aprender haciendo” bajo un acompañamiento Par de las personas mas de mayor experiencia del equipo.

Desde el onboarding se impulsa un feedback continuo, un entorno flexible, autónomo, creando cercanía y de confianza entre líder y colaborador.

Pdt: Desde el Onboarding también se promueve la buena onda!

Offborading

- Proponer un feedback Bi-direccional
- Plan agil de entrega y gestión del conocimiento. (Garanticemos que lo que se logro hacer, quede a conocimiento del equipo de trabajo)
- Documentación técnica de cierre.

Modelo de aseguramiento de la calidad sin QA embebido

Se debe pensar en un modelo operable
El modelo debe tener como premisa que la calidad no debe ser un negociable
El equipo es dueño de la calidad, se considera un modelo de responsabilidad compartida

Actividades

- Aseguramiento de la calidad desde etapas tempranas: riesgos identificados, HU y funcionalidades bien definidas y medibles. Criterios de aceptación acordados.
- Test como responsabilidad compartida de equipo: al no tener QA embebido en equipos, la responsabilidad debe considerarse compartida. Se proponen ejecución Pruebas unitarias (innegociables), Revisión par de código, revisión par de pruebas. Adicional pruebas funcionales por solicitante o product owner
- Casos de prueba definidos
- Definir y mantener el cumplimiento de estándares de seguridad, pensar en frameworks de testing
- Uso de herramientas de revisión de código: implementar herramientas open source por el equipo de desarrollo para garantizar la calidad de código.
- Estándares de calidad mínimos y acordados con el equipo: Aunque QA no este embebido se pueden definir reglas y estándares por contemplar, serán los responsables de su ejecución.
- Code review: Par de revisión del producto desarrollado.
- No se aprueban PR sin test ejecutados.
- Auditoria de equipos
- Automatización como prioridad sobre manualidad (esquema de madurez de QA no embebido en los equipos de trabajo)
- Pipelines obligatorios
- Observabilidad minimante en ambientes de PRD
- Acompañamiento continuo

ENTREGABLE B

Consideraciones
- Incremento de incidentes en PRD
- Incremento de los tiempos totales de entrega - Leadtime
- Se debe sacar nuevas funcionalidades
- No se cuenta con equipo de QA
- Dependencia con proveedor

Restricciones
- No aumentar costos

Proposito de la estrategia
- No seguir apagando incendios
- Recuperar la calidad del productos de software
- Considerar la velocidad de manera sostenible
- No aumentar costos.

Posibles hipótesis sobre el escenario planteado

1. No garantizar el aseguramiento de la calidad dentro del proceso de desarrollo, puede recaer en ambientes productivos generando afectación en el producto y tiempos de entrega de nuevas funcionalidades

2. Si existen deudas técnicas sin gestionar es posible que afecte el proceso de desarrollo (código poco mantenible, decisiones técnicas postergadas sin tomar)

3. La no priorización de requerimientos con negocio, puede llevar a una presión al equipo de desarrollo sin un rumbo claro, es importante realizar priorizaciones realistas, que generen valor al negocio, sobre posibles deseos particulares. Esto es un ejercicio consciente y colectivo entre equipo y negocio.

4. Mitigar posibles fricciones con el aliado - proveedor - de forma que se pueda trabajar de manera conjunta, acompañamiento par dado el caso lo requiera, con un propósito claro, garantizando el satisfacción de los requerimientos solicitados.

5. Medir la ejecución del proceso de desarrollo, eliminando posibles quiebres del mismo.

Plan 30 - 60 - 90 días

PRIMERA FASE - 30 DIAS

- Entender e identificar la causa real y de raíz de las consideraciones identificadas. Se actúa de manera rápida para la ejecución de esta etapa.
- Se priorizan el backlog a ejecutar - medible y lograble - este ejercicio se realiza de manera conjunta entre equipo de negocio y equipo de desarrollo - se consideran **MVP**. Ejecutado esta etapa se debe considerar pausar los posibles nuevos features.
- Desarrollo de features garantizándo ejecución de pruebas (mínimamente pruebas unitarias y funcionales, no necesariamente automatizadas, manuales pero que se garanticen)
- Revision par de desarrollo para mitigar posibles bugs

SEGUNDA FASE - 60 DIAS

- Madurar acciones de QA y los estándares ejecutados en primera etapa- uso de herramientas open source para evaluación de código
- Ejecución de pruebas par entre equipo de desarrollo.
- Evaluación de resultados de la primera etapa - métricas generadas sobre tiempos, cantidad de bugs identificados, cumplimiento funcional, etc.

TERCERA FASE - 90 DIAS

- Considerar reservar capacidad del equipo (10% de capacidad) para continuar madurando escenario de aseguramiento de calidad y cierre de posible deuda técnica pendiente a nivel de código
- Continuar con ejercicio de priorización de Backlog de negocio, realistas, dando satisfacción a negocio pero mitigando escenario que hagan perder el rumbo del producto a construir y entregar - desde el valor no desde el deseo
- Estándares de diseño y practicas de revisiones ágiles - Sobre features medibles, bien hechos y no funcionalidades gigantes

Tradeoff por comunicar

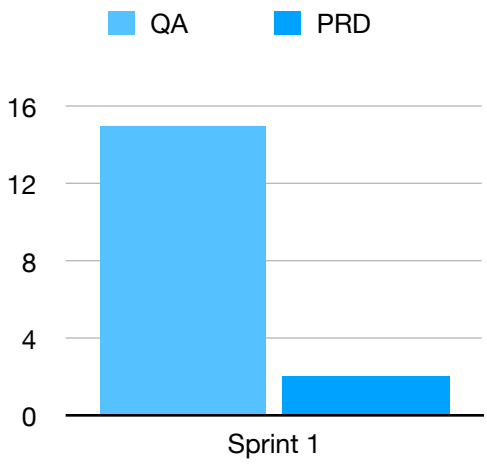
Con relación a los aliados o proveedores
Acompañamiento mas cercano al proveedor, No desde la falta de confianza, sino como un trabajo colaborativo para cumplir un mismo propósito: garantizar un producto funcional, que cumpla las expectativas de negocio y que permita mejorar nuestros tiempos de entregar

Con relación al aseguramiento de la calidad.
Se debe buscar un correcto balance entre velocidad y calidad, para esto es requerido incluir actividades de aseguramiento de la calidad el producto, siendo sensible a los costos que un equipo de QA o automatización de pruebas puede traer, debemos ejecutar en los equipos pruebas manuales y mínimas, que minimicen el crecimiento de bugs, en esta actividad se requiere participación par de las personas del equipo de desarrollo.

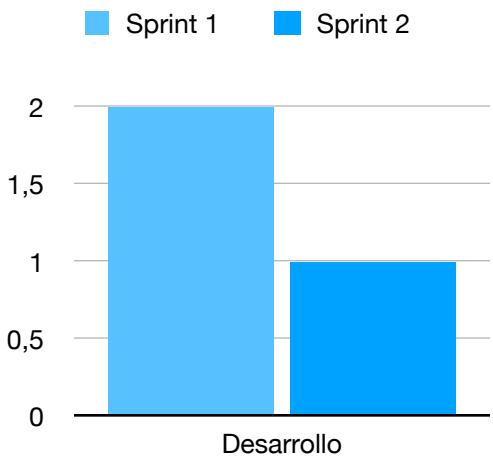
Con relación a negocio.
Queremos trabajar de una forma positiva y realista, es importante tener una priorización consciente, que genere valor al negocio, saber priorizar es crear un producto incrementar y evolutivo en el tiempo. Siendo sensibles a la prioridad de nuevos features, también seremos sensibles en priorizar lo que tener valor, de forma colaborativa entre equipo técnico y de negocio. Busquemos un balance entre velocidad y calidad, no es necesario sacrificar ninguno de los 2, si logramos definir en periodos de trabajo - sprints - features alcanzables, funcionales y de calidad.

Tablero de Control para el escenario evaluado

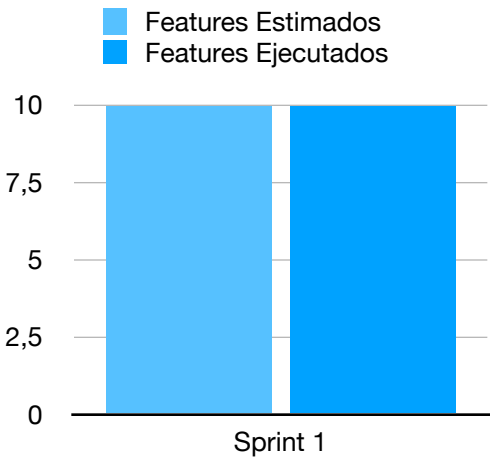
Bugs identificados QA VS PRD *



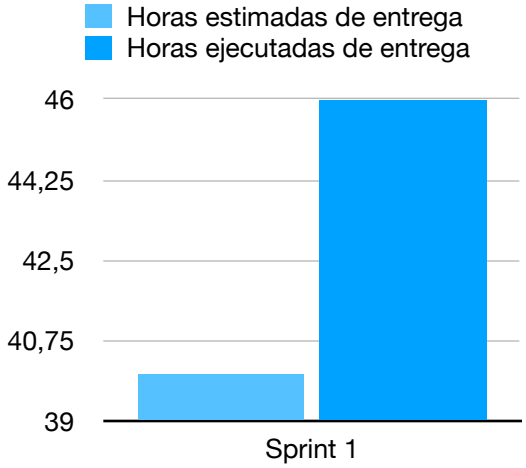
% Rollbacks Ejecutados *



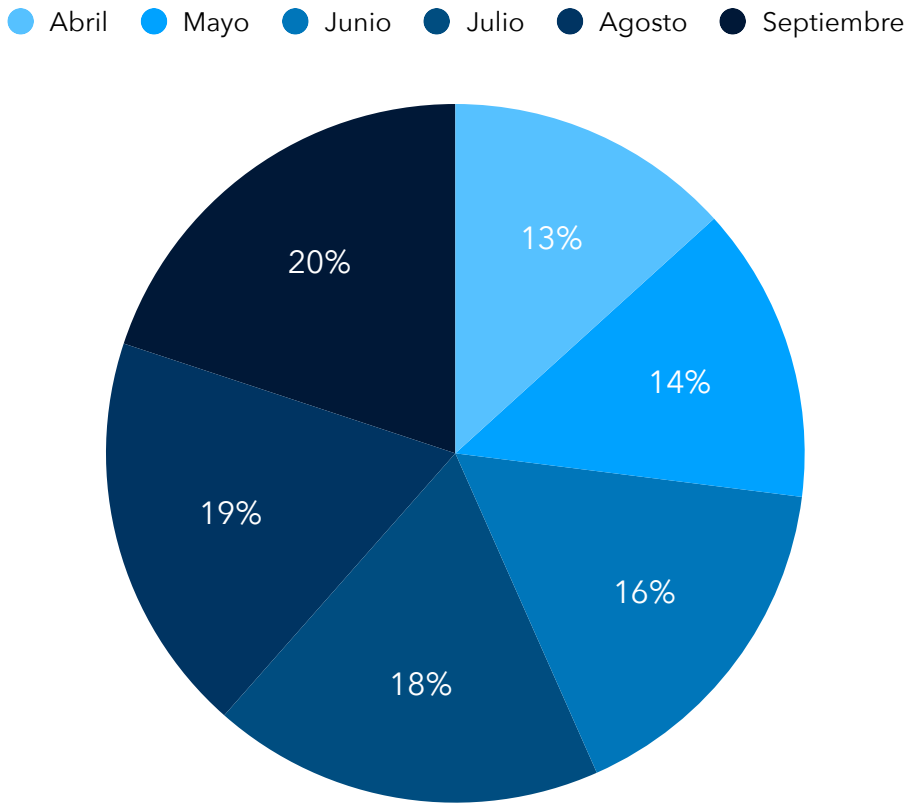
Features Estimados VS Ejecutados *



Lead time | Variación en los tiempos de entrega - horas*



Cumplimiento de Delivery | features Priorizados por mes *



***(*)** Gráficas ilustrativas para efectos de la prueba técnica*

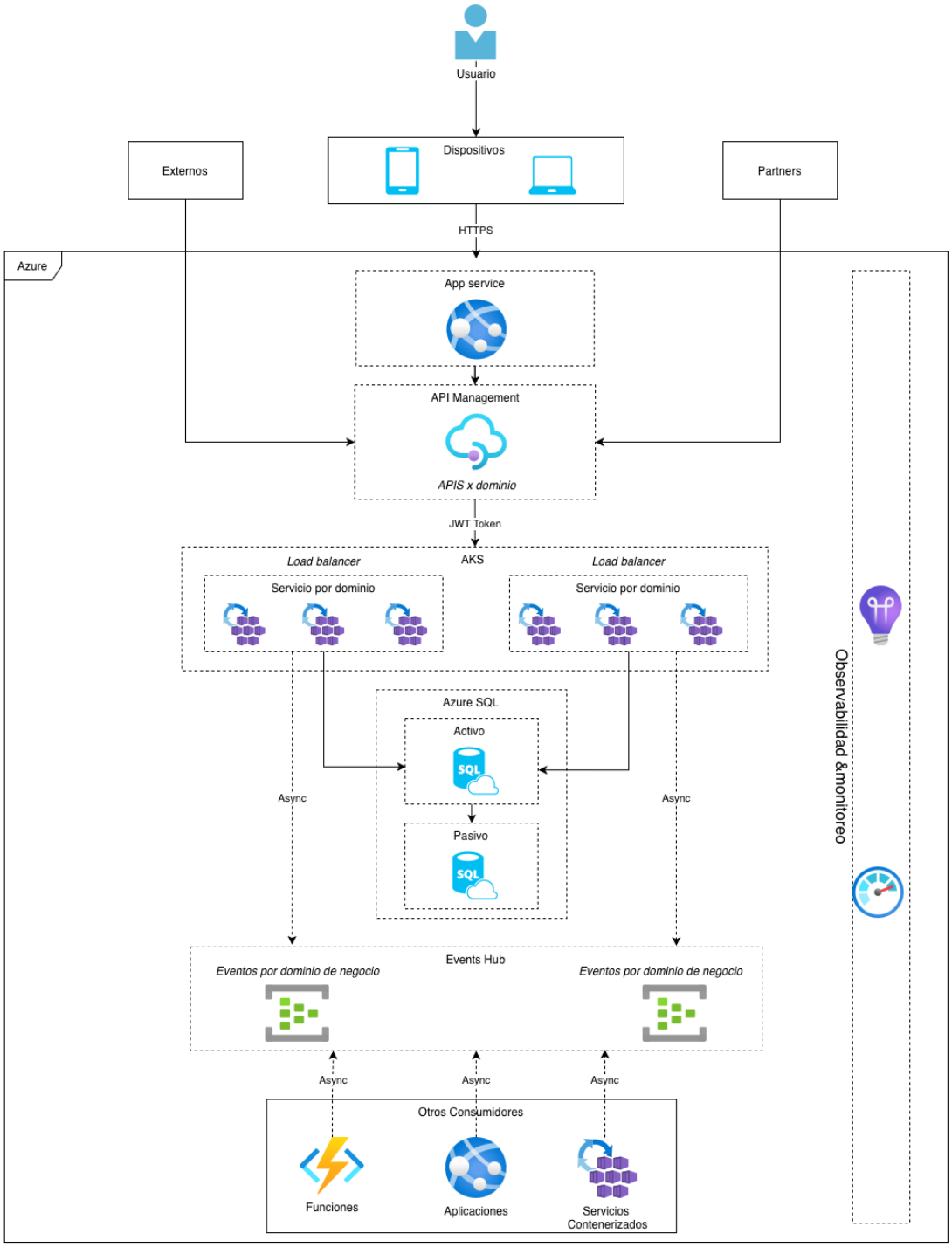
Consideraciones dentro de la estrategia 30 - 60 - 90

- Dentro de los Sprint de cada fase es importante continuar evaluando posibles bloqueos críticos que se puedan presentar
- Es importante un seguimiento continuo a la estrategia.
- La estrategia es sensible a ajustes, buscando lograr los propósitos trazados, con un seguimiento continuo es fácil ajustar, rápido y barato.
- Cada sprint de debe evaluar el impacto de calidad y velocidad, como premisas dentro del ejercicio propuesto.

ENTREGABLE C

Arquitectura de servicio de alta transaccionalidad

Notación: UML
Entorno: CLoud - Azure
Diagrama: Componentes



Solución *Bedoya*

Atributos de calidad

Resiliencia
Timeout mediante APIs administradas
Patrones de <<Circuit Breaker>> mediante APIs y Eventos publicados
Patrones de <<re-intentos>>
Consistencia
Separación de eventos por dominios de negocio
Base de datos transaccionales
Observabilidad
Componentes de monitoreo
Centralización de monitoreo
Logs estructurados para observabilidad
TPS por APIs de negocio como métrica

Seguridad *
Validación mediante JWT para consumo de servicios
Oauth2 para interacción de APIs
Identidad entre pods de servicios
Cifrado en reposo y transito

(*) Atributo de calidad innegociable para cualquier diseño

Lista de Hallazgos relevantes dentro de un <<PullRequest>>

Violación de principio de de desarrollo de Software
Bajo Acoplamiento - Alta cohesión
Principos SOLID - KISS - DRY - SoC

No aplicar Patrones de diseño
Los patrones son Soluciones comprobadas a problemas recurrentes dentro de la construcción del software

Código con baja legibilidad y mantenibilidad
Desarrollo que solo "lo entiende" Quien lo construyo

No cumplimiento de estándares definidos para el desarrollo.
Nombramientos? CamelCase? en ingles o en español?

Cambios funcionales sin pruebas ejecutadas
Pruebas unitarias automatizadas, ejecución con herramientas de verificación de código (Ej SonaQube, entre otras)

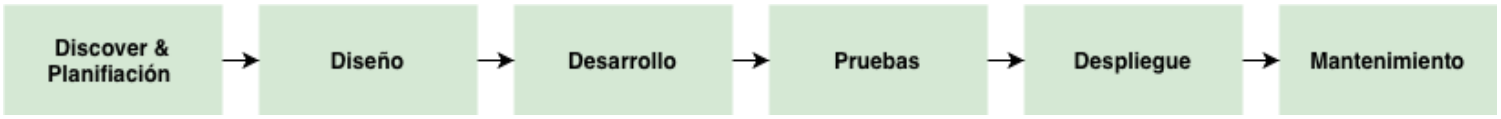
Riesgos de seguridad
Credenciales o secretos quemados en código
Uso inseguro de dependencias

No manejo de errores
No incluir excepciones dentro del código
Errores sin contexto.

Ruptura de la arquitectura propuesta
Mantener el diseño hace parto de la correcta construcción de software

Pull Request con múltiples funciones u objetivos
Full request extensos mas complejos de verificar, de mantener y de identificar posibles errores

Estándar SDLC



Actividades SDLC

- Fases definidas y criterios de aceptación claros, definidos desde el inicio y con equipo de trabajo
- Commits medibles, pequeños, evaluadles
- Código alineado a los estándares definidos
- Ningún cambio sin una evidencia de solicitud se debe ejecutar (tickets, HU, etc, siempre dejar evidencia de lo solicitado y ejecutado)
- Pull request con descripción clara de los cambios ejecutados
- Evidencias de pruebas ejecutadas
- Pipeline Falla si el test Falla (no debería pasar el pipeline)
- Monitoreo continuo
- Feedback Continuo
- Deuda técnica gestionada por equipo, no se ignora

Modelo de ejecución propuesto SDLC
Iterativo - Agil

Métricas de evaluación de proveedor externo de desarrollo

Cumplimiento de buenas practicas
Principios de desarrollo de software, lineamientos internos de desarrollo

Cumplimiento de tiempos de entrega
Miden su On-time Delivery? Alguna métrica de lo ejecutado?
Variación de tiempo de entrega que cumplan estándares técnicos

Cumplimiento de estándares técnicos
Ambientes de desarrollo y QA independientes
Automatización de despliegues
Herramientas de Evaluación de código

Documentación técnica
Garanticemos que tenemos insumo para saber lo que se construyo.

Aseguramiento de la calidad
Pruebas automatizadas? Manuales? Pero garanticemos su ejecución
Evidencias del resultado de QA
Bugs detectados después de “finalizado” en DEV

Para las anteriores métricas mencionadas se puede considerar evaluar datos porcentuales del proveedor tales como (%)

% de bugs identificados por HU en QA
% de rollback ejecutados por ejecuciones no satisfactorias
% de cumplimiento de lineamiento de desarrollo
% de cumplimiento de estimación de tiempos de delivery
% busg identificados en PRD

Escenario de Iteración - Proveedor de desarrollo software

Tiempo estimado: 2 semanas
Consideraciones: Alto volumen de retrasos
Diagnostico: Se prioriza velocidad, no calidad

Semana 1

- Reducir Scope: Se prioriza solicitudes bajo una priorización de valor, minimizando el volumen, no la calidad
- PR pequeños, medibles y revisables
- Criterios de aceptación muy claros
- Evaluación técnica temprana, que se cumplan los mínimos requeridos para un correcto desarrollo

Semana 2

- Revisión par de código en linea dado que sea requerido y ejecución de Pruebas mínimamente funcionales.
- Mido y genero métricas de resultados de semana 1
- Según el resultado se continua bajo el mismo plan, en caso contrario se evalúa al proveedor.