







El **perímetro** de su MVP debe **reducirse** mientras le permite comercializar su producto.

Apuesta a **Early Adopters** y recibe un máximo de **feedback**.

Su MVP está implementado y puede utilizarse en **producción**.







Experimente rápidamente la solución (unas semanas), reúna comentarios de sus usuarios y aprenda de sus errores.

No tengas miedo de cambiar todo

No lo olvides, **ifallarás!**







[BESO]

Evite la sobreingeniería , si un modelo de "papel" o un Formulario de Google es suficiente para probar su concepto, no vaya más allá.

iMantente simple! Tanto técnica como funcionalmente.









HORA DE COMPRAR

[PRODUCTIVIDAD]

Limite sus especificaciones a lo esencial, concéntrese en el "qué" en lugar del "cómo".

El producto debe ser lo más autodocumentado posible.

La documentación debe estar versionada de la misma manera que el código.









Las soluciones de SaaS son sostenibles y rentables.

En algunos casos, **SaaS** puede **acelerar** la implementación de **MVP**.

Piense en la visión económica con respecto a las alternativas en términos de costo total (TCO : T otal C ost de O ** wnership) y no solo en términos de costo de la licencia.







El ritmo de evolución y entrega del negocio central debe ser **compatible con la agilidad** de los servicios que lo consumen.

El negocio principal debe **exponer servicios**.

El negocio principal debe adoptar un principio **orientado al evento**, informa acciones de gestión en forma de eventos.



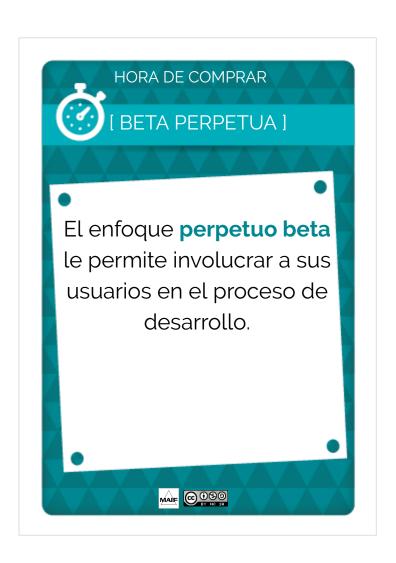




Aproveche **la implementación continua** para adaptar **** producción **a los requisitos** comerciales ** y no al revés.

Las implementaciones en todos los entornos, hasta producción , deben ser automáticas y frecuentes.







[BETA PERPETUA]

Siéntase libre de utilizar el principio beta perpetuo en el que los usuarios participan en el desarrollo.

El término beta perpetuo se refiere a una aplicación desarrollada en just-in-time, **en constante evolución**, y no en un producto incompleto.











[PERCEPCIÓN]

No descuides el trabajo de los diseñadores de UX, es fundamental en el desarrollo de una aplicación.

Integre el **feedback** de sus usuarios, este es esencial.







[ACTUACIÓN]

Las interfaces están orientadas a **eficiencia**.

El **rendimiento** de una interfaz ahorra tiempo **, aumenta la** satisfacción de los usuarios **y por lo tanto** guarda su frustración **.







[MÓVIL PRIMERO]

Los dispositivos móviles son la **parte más importante** del **mercado**.

Pensar en dispositivos móviles es pensar en lo **esencial**.

Diseño receptivo es la norma, es una fuente de ahorro (**MVP**).







[OMNI-CANAL]

El enfoque omnicanal proporciona al usuario una **experiencia unificada** (ejemplo: Netflix).

Los diferentes **canales** están **sincronizados** y **coherentes** (a diferencia de los procesos por lotes).

Todos los actores (clientes, asesores) tienen acceso a la misma información.







[AUTODATOS]

Deje **personas**, en cualquier momento, **control** en sus datos **personales**.

Establezca una **confianza** al permitir a los usuarios la trazabilidad y el control en tiempo real.

subsistemas deben cumplir los mismos requisitos.







[CRM / SFA]

Opte por **CRM** que gestione tanto la relación con el cliente como el liderazgo de la fuerza de ventas (**SFA** : **S** ales **F** orce **A** utomation).

CRM debe estar **abierto** para nuevas oportunidades.

CRM produce eventos correspondientes a las acciones de gestión para ajustarse a la lógica de la plataforma orientada a eventos.







[BIG DATA]

Centralice los datos **** Maif Group
** Partner **y** Vendor **en una lógica**
pathway **.

La "preparación de datos" y el procesamiento pueden **consolidar** los datos.

Los equipos de Big Data colaboran con los equipos de características para garantizar el gobierno de datos.







[PUESTO DE TRABAJO]

Adopte la **federación de identidades** para una experiencia unificada.

Un **portal** permite ofrecer **una descripción general** , no reemplaza las aplicaciones.

La estación de trabajo debe ser móvil, multicanal y estándar para permitir la apertura dentro de empresa extendida.







[COLABORADORES]

Trate a todos sus usuarios como "clientes": usuarios de Internet, gerentes, operadores, desarrolladores, etc.

No subestime el **esfuerzo de UX** para implementar aplicaciones de administración de uso interno.







[TODA LA MEDIDA]

Piense en las métricas durante el desarrollo de la aplicación. logs debe tener una dimensión comercial así como técnica.

No descuides las **métricas de rendimiento** , son fundamentales.

El equipo de características proporciona **operación**: es responsable de hacer **la aplicación utilizable**.







[Prueba A / B]

En lugar de decidir arbitrariamente entre dos soluciones, no dude en configurar **pruebas A / B**.

Este patrón consiste en presentar dos versiones diferentes de la misma aplicación y elegir una de ellas en función de medidas objetivas de la actividad del usuario.





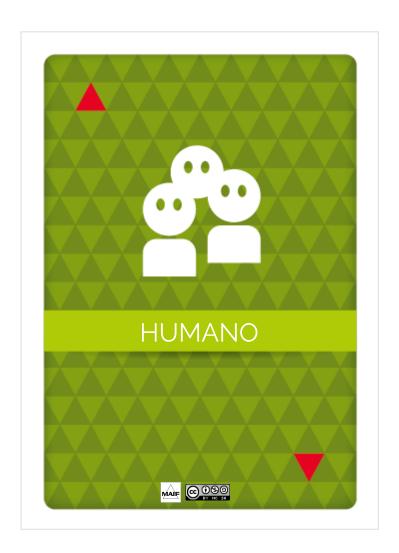


[DETERIORO]

En falla de uno de los subsistemas, una versión degradada del servicio debe considerarse en primer lugar en lugar de una interrupción.

Con **Disyuntores** , **aísle un desglose** para **evitar** su **impacto** y **propagación** en todo el **sistema**.











HUMANO

[EQUIPO PRINCIPAL]

Los equipos son **Equipos destacados**, organizados en torno
a un conjunto funcional coherente,
y compuestos por todas las **habilidades** necesarias para este
conjunto.

Por ejemplo: Business Expert + Web Developer + Java Developer + Architect + DBA + Operational.

La **responsabilidad** es **colectiva**, el Equipo de funciones tiene el poder necesario para esta responsabilidad es **colectiva**, el Equipo de funciones tiene el poder necesario para esta responsabilidad es **colectiva**, el Equipo de funciones tiene el poder necesario para esta responsabilidad es **colectiva**, el Equipo de funciones tiene el poder necesario para esta responsabilidad es **colectiva**, el Equipo de funciones tiene el poder necesario para esta responsabilidad es colectiva, el Equipo de funciones tiene el poder necesario para esta responsabilidad es colectiva.



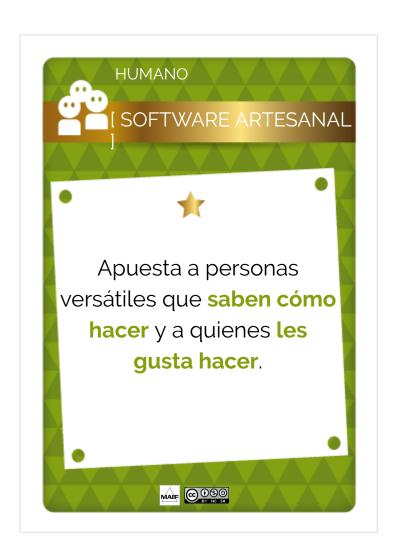


[EQUIPO 2-PIZZA]

Limite el tamaño de un equipo de características: **entre 5 y 12 personas**.

Por debajo de 5, ella es demasiado sensible a los eventos externos y carece de creatividad. Por encima de 12, pierde productividad.

El término "Equipo de 2 pizzas" indica que el tamaño del equipo de características no debe exceder el número de personas que pueden ser alimentadas con dos pizzas.





Lo más importante es la **cultura** de desarrollo, escalabilidad y adaptabilidad.

Reclutando artesanos de software y desarrolladores de software completo, aportan un valor agregado real a través de su conocimiento y su visión general.

Sin embargo, los desarrolladores móviles, por ejemplo, suelen ser desarrolladores especializados.







[RECLUTAMIENTO]

Proponer modos de trabajo adaptados a los empleados: movilidad, trabajo a domicilio, CYOD CYOD (Choose Your Own Device.

Deje tiempo para la experimentación y hágalo realidad en tiempo de trabajo.







La organización debe ser un motor de cuidado diurno mediante el establecimiento de sistemas como educación continua o universidades empresariales.

Siéntase libre de combinarlos con otras formas **más informales como**: Codificación de Dojos , Almuerzos de Brown Bag , Conferencias ** Externas.







[CO-CONSTRUCCIÓN]

Para romper las barreras entre los intercambios, no es suficiente agrupar a las personas en torno a un producto común en un lugar común.

Los **Enfoques ágiles** eliminan estas barreras para asegurar **la** convergencia de objetivos.

Estas prácticas son una parte integral de las claves del éxito, la organización es el garante.









[DEVOPS]

Adoptar **DevOps** para converger **Dev** y **Ops** hacia un objetivo común: **servir a la organización**.

iLos intercambios siguen siendo diferentes! DevOps no significa que la misma persona realiza las tareas de Dev y Ops. Los desarrolladores y Operativo deben colaborar para beneficiarse de **** habilidades *y para mejorar* empatía **.







[DOLOR]

En una organización tradicional, **la falta de comprensión** entre los equipos suele estar relacionada con la distancia y **la falta de comunicación**.

Los miembros de un equipo de características son co-responsables y solidarios para todas las tareas.

Dolor es un factor clave en **Mejora** continua.







[CDS]

Los equipos de características se basan en principios que se basan en gran medida en **colaboración** y **participación colectiva**.

Los centros de servicios están avanzando hacia la racionalización y consolidación de TI por parte de las empresas, lo que es contrario a esta noción de compromiso colectivo.







[VALIDACIÓN]

Asegúrese de que la organización conserve **su función de validación** en las herramientas y usos. En particular sobre las herramientas que afectan el patrimonio ** (ejemplo: gestión del código fuente).

Proporcione Equipos destacados con **medios** para respaldar sus elecciones.

No seas **dogmático** y asegúrate de estimular la experimentación.









[TRANSVERSALIDAD]

No cree barreras entre **Equipos destacados**.

Configure una **organización** y la **agilidad** necesaria para que los equipos de características se comuniquen entre sí y compartan sus habilidades y experiencias.

La organización de la transversalidad en **Spotify** (Tribus, Capítulos y Gremios) es un ** ejemplo elocuente.











I API PARA TODOS I

Abra su organización para nuevos usos y nuevos clientes con **Public APIs**.

En **sociedades** comerciales , clientes **como** proveedores ** , las API son el formato de intercambio estándar.

Las API también están destinadas a usos internos de la organización.







El uso de API debe ser lo más simple posible. Piensa en **la experiencia del desarrollador**.

La mejor solución para validar la adecuación con la necesidad es **probar la API rápidamente** : iunos pocos minutos deben bastar!

La plataforma debe ofrecer una **interfaz gráfica** para probar la API simplemente.







Implemente una solución de administración de API para administrar cuotas, aceleración, autenticación y registro.

Recoja métricas para administrar **monitoreo**, **filtrado** y **informes**.







Requerir que **sistemas externos** cumplan los mismos **requisitos** que **sistemas internos**.

Los sistemas externos deben publicar **eventos** y permitir la supervisión **técnica**.

En el caso donde los datos del sistema externo deben integrarse, la sincronización **total** debe ser **posible**.







Incluso si la marca blanca no se considera en la base, configure una arquitectura multi-tenant. Su aplicación **inicial** es la primera **celebración**.

Piense en la **instanciación multifuncional** del sistema desde el principio.







Los idiomas, las monedas, las reglas comerciales, los perfiles de seguridad deben ser fáciles de configurar.

Tenga cuidado con hipergenericidad, a menudo es inútil y fuente de costo.

la configuración debe ser **escalable** y rápida según sea necesario.







la función voltear se trata de diseñar una aplicación como un conjunto de **funciones** que se pueden **habilitar** o **desactivar** caliente, **producción**.

En una aplicación **multi-tenant**, la función flipping le permite personalizar a los seguidores.

Le función voltear **simplifie l'A / B** testing.













El Equipo de características debe actuar **responsablemente** para identificar las opciones que lo afectan exclusivamente y las elecciones que impactan a la organización.

Las opciones que exceden el alcance del Equipo de características (por ejemplo, licencia, lenguaje de programación infrecuente) deben validarse por la organización o por el proceso de convergencia entre pares.





Una mala herramienta impuesta a todos es un riesgo. El uso indebido de una buena herramienta puede tener consecuencias muy dañinas **. Por ejemplo, los métodos ágiles mal utilizados son peligrosos.

herramientas debe ser cuestionado.

Excel a menudo es una opción racional pero no es una herramienta para hacer todo **
(CRM, ERP, D





Cuanto más una herramienta tenga una función de característica diferenciadora para la organización, más se pretende que se cree. El negocio principal debe permitir la especificidad y adaptarse rápidamente ya menudo. Algunos paquetes de software a veces se adaptan a esta necesidad.

Para **el resto** : SaaS, Open Source, Build o Owner se estudiarán **caso por caso**.





Las **soluciones propietarias** son un **riesgo** para la organización que debe poder reanudar el mantenimiento si es necesario.

Hay pocas herramientas propietarias que no tienen **alternativas de código abierto**.

La organización se beneficia de la Comunidad de Código Abierto y puede devolver sus contribuciones.







acoplamiento débil debe ser la norma.

Cada micro-servicio tiene una interfaz claramente definida.

Esta **interfaz** determina el **enlace** entre los **micro servicios**.

Domain Driven Design permite, especialmente con **Contextos acotados**, anticipar este problema.







Un **Data Store** está destinado **a ser acoplado** solo con **un solo micro- servicio**.

El acceso a los datos de un microservicio a otro se hace exclusivamente a través de su interfaz.

Este diseño implica consistencia en el tiempo en toda la plataforma. Debe ser aprehendido en todos los niveles, incluido UX.







Un micro-servicio ofrece **un número razonable de funciones**.

No dude en cortar un microservicio cuando este empiece a crecer.

Un servicio de tamaño razonable permite **considerar** serenamente **la reescritura**, si surge la necesidad.







[CAPACIDAD DE RESPUESTA]

La programación sensible se centra en el flujo de datos y la propagación del cambio. Se basa en el patrón " Observer " al contrario del enfoque " Iterator ", más tradicional.

El Manifiesto reactivo establece los ejes fundamentales:
disponibilidad y velocidad,
resiliencia a las interrupciones,
flexibilidad, elasticidad y
orientación del mensaje.





El intercambio entre aplicaciones debe ser **asincrónico** primero.

Los intercambios asíncronos naturalmente **permiten un acoplamiento** débil , aislamiento **y** control de flujo (contrapresión **).

la comunicación síncrona solo debe considerarse cuando la acción lo requiera.







Los procesos **** *de* procesos impulsados por eventos están naturalmente implementados de forma asincrónica.

La **orientación del evento** permite favorecer la implementación de enfoques tales como **C** ommand **Q** uery **R** sponsibility **S** egregation (**CQRS**) y **Event Sourcing**.







ESB mostró límites : el mantenimiento escalable es crítico , tanto desde el punto de vista técnico como organizacional.

Los mensajes de Broker como Kafka ofrecen una solución simple , duradera y flexible.

Puntos finales inteligentes y Tuberías simples es una arquitectura que funciona a escala: es Internet.







Si la sincronización entre dos sistemas está garantizada por un flujo de eventos , la resincronización total de estos sistemas debe planificarse en el momento del diseño.

Una **** *auditoría de sincronización* automática (ejemplo: por ejemplos) permite medir y detectar posibles errores de sincronización.







La **configuración** de los **microservicios** está **centralizada** para todos los **entornos**.

Un directorio central garantiza descubrimiento dinámico de micro-servicios.

La escalabilidad global depende de este directorio **.







Los equipos especiales mantienen un entorno de **recinto de seguridad** (versión actual y próximo lanzamiento) para permitir que otros **equipos amplíen**.

En algunos casos no nominales, funciones pueden estar inhabilitados en el entorno de desarrollo.







Tu **sistema fallará**, es inevitable. Debe estar diseñado para esto (**Design For Failure**).

Predecir **redundancia** en todos los niveles: **hardware** (red, disco, etc.), **aplicaciones** (instancias múltiples de aplicaciones), **zonas** geográficas, **proveedores** (ejemplo: AWS + OVH).







Atención a los componentes técnicos casas y transversal! Son restrictivos, caros y difíciles de mantener.

aceleradores, juegos de herramientas, acumulaciones técnicas pueden ser agrupados, gratuitos Equipos destacados, evitando un enfoque dogmático.







Los servicios de PaaS son preferidos, simples y se escalan rápidamente.

Los servicios **laaS** le permiten abordar casos que requieren una mayor **flexibilidad** pero requieren más trabajo operativo.

Una nube privada no es un entorno de virtualización tradicional, sino que se basa en hardware básico.







Los problemas de infraestructura no están dentro de **Equipos destacados**. La infraestructura debe **proporcionarse** y **mantenerse** mediante un servicio ** con funciones cruzadas.







Los contenedores proporcionan la flexibilidad que necesitan los equipos de características para habilitar herramientas heterogéneas en un contexto homogéneo.







Los contenedores (ejemplo: **Docker**) hacen posible **ser liberado** de las diferencias de entorno.

El proceso de **implementación** debe ser **agnóstico** para el entorno.

Algunos componentes como las bases de datos no se deben implementar en contenedores. Su implementación aún está automatizada.







Las **métricas** son **accesibles** para todas las personas con diferentes niveles de granularidad: vista detallada de la función del equipo relevante, agregaciones para otros miembros de la organización.

El acceso a **las métricas no** implica el acceso a los datos de la unidad , debe controlarse para mantener la confidencialidad.

Todos los entornos están afectados.









las revisiones de los códigos son sistemáticas. Son dirigidos por miembros del Equipo de funciones u otros miembros de la organización, como parte de **Continuous Improvement**.

Eso **no está siendo auditado sino su código** : "iUsted no es su código!".

El **qualimetry** puede ser parcialmente automatizado, pero nada supera al "nuevo ojo".









[PRUEBAS AUTOMATIZADAS]

Pruebas automatizadas asegura calidad del producto a lo largo del tiempo.

Es **un prerrequisito** para la implementación continua, permite **** cambios *y* implementaciones frecuentes **.

iEl lanzamiento de la producción se convierte en un evento anecdótico!







Las pruebas de **integración** y **funcional** son las más importantes **garantizan** la **operación** efectiva **.

Las pruebas de **unidad** son adecuadas para **desarrollo**.

las pruebas de rendimiento miden el rendimiento a lo largo del tiempo **.

Las pruebas de resiliencia ayudan a anticipar fallas.



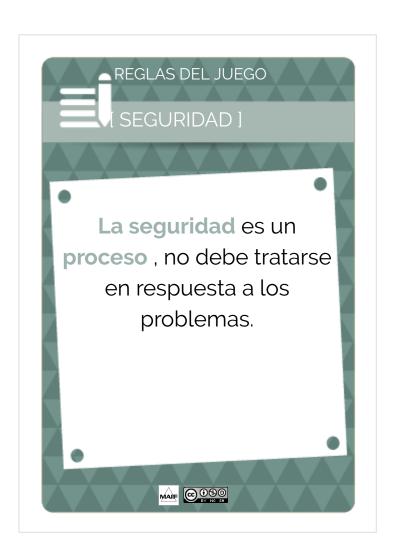




La **cobertura del código** según las pruebas es una **buena** métrica de la calidad del código.

Esta es una condición necesaria pero no suficiente, la cobertura de una estrategia de prueba incorrecta puede ser alta sin garantizar la buena calidad del código.







expertos en seguridad pueden integrarse directamente en los equipos de características si es necesario.

expertos en seguridad están disponibles en la organización para auditoría , conocimiento y reenvío.

