Quién

Endika García

Qué

Zeplin, o Demos

Por qué

En el caso de Zeplin, porque nos facilita muchísimo el paso del material de diseño al equipo de programación. En el caso de las Demos, por muchos motivos, para aumentar la consciencia del equipo de lo que cuesta conseguir las cosas, trabajar la empatía con el cliente, con el comercial, etc, para poner en valor el trabajo realizado, dar visibilidad a las mejoras implementadas dentro de la empresa, tener más cuidado con los fallos y no dar por terminadas las cosas a la ligera, ...

Cuándo

Zeplin, durante el desarrollo de una app (Al menos para móviles) Demo Al terminar un proyecto, o parte de un proyecto, presentable, que se pueda considerar terminado.

Links

https://zeplin.io/ https://www.inc.com/geoffrey-james/give-a-great-product-demo-5-rules.html

Quién

Joaquin Engelmo Moriche

Qué

¿De verdad eran necesarios los microservicios?

Por qué

Llevo cuatro años metido en los microservicios, antes incluso de saber que así le llamaba la gente. Cuatro años con experiencias positivas y negativas de un trending topic que no se puede tomar a la ligera para tampoco se debe criticar a lo loco. Mi objetivo es compartir con los demás mis experiencias usando esta solución en un producto que usan millones de personas y una base de código de más de 7 años con un monolito bastante tocho que se ha ido, y se sigue, migrando a microservicios. Incluso, locura máxima, hemos sacado microservicios de un microservicio, wait for it :D

Cuándo

¿Tienes una base de código inmensa donde trabajan a la vez bastantes desarrolladores (> 50)? ¿Qué tal la gestión de ramas? ¿Qué tal el continuous delivery? ¿Cuánto tarda? ¿Y los hotfix? Además por motivos organizativos, ¿te gustaría tener equipos "dueños de componentes" o "áreas/features" de tu producto? ¿Te gustaría introducir lenguajes/tecnologías distintas dependiendo del problema a solucionar y que todo conviva en un "mismo sistema"?

La respuesta a todas esas preguntas podría llevar la palabra microservicio como opción y se podrían aplicar.

Links

- Mega must => https://martinfowler.com/microservices/
- Book => Building Microservices: Designing Fine-Grained Systems

Quién

Fernando Gil

Qué

Todo tecnosexualidad: serverless, docker, devops

Por qué

Busco tecnologías que me faciliten la vida y si es posible que reduzcan costes de implantación/mantenimiento. Muchas veces un requisito oculto que tengo es reducir costes (y prácticamente siempre calendarios ajustados). También ocurre que necesito entornos reproducibles con un hardware muy cambiante.

Cuándo

Mayormente desarrollos a medida donde yo pongo el stack (no suelo tener requisitos al respecto). Docker sobre todo en despliegues locales, para evaluar herramientas o en entornos de desarrollo o controlados. Hace poco tuve un caso de uso para la distribución de un software bastante complicado de compilar/instalar.

Links

Para serverless, los mismos quickstart que da Amazon en su docu son un punto de partida. Hay toolkits (que no he probado) como https://serverless.com/ o https://claudiajs.com/ que facilitan las cosas. Para dockerizar, siempre va todo a medida asi que es difícil concretar, pero suelo basarme en https://github.com/phusion/baseimage-docker . La docu oficial da mucha info extra.

Quién

Imanol Pinto

Qué

Kafka y kafka streams

Por qué

Estamos montando el sistema de notificaciones de nuestra empresa, donde cada notificación (sms, email, chat, push notif. etc) va a pasar por nosotros. Por cada petición vamos a tener que enriquecer/completar los datos, tener en cuenta las preferencias del usuario, compilar la notificación si aplica (con

su plantilla), enviarlos por el sistema que corresponda y almacenar todas las comunicaciones y sus actualizaciones de estado. Teniendo en cuenta este escenario y que manejaremos una gran cantidad de mensajes, kafka nos viene como anillo al dedo. También estamos aprovechando kafka-streams (para el enriquecimiento) y exactly once (incluídos en kafka 0.11).

Puedo enseñar la arquitectura que hemos montado y explicar por qué hemos decidido hacerlo de esa manera. Por otro lado, me he pegado un poco con kafka, y es bastante probable (o no xD) que pueda resolver dudas.

Cuándo

Creo que he respondido a esta pregunta en la anterior también...

Links

- Doc oficial kafka: https://kafka.apache.org/documentation.html
- Exactly once explained: https://www.confluent.io/blog/exactly-once-semantics-are-possible-heres-how-apache-kafka-does-it/
- Cómo usa kafka Uber: https://eng.uber.com/reliable-reprocessing/
- Cómo usar spring-cloud-streams: https://docs.spring.io/spring-cloud-stream/docs/Chelsea.BUILD-SNAPSHOT/reference/htmlsingle/index.html

Quién

Alejandro García

Qué

"CSS - Arquitecturas imposibles" (envío un par de opciones, ésta es 1/2)

Por qué

Escribir CSS es fácil, escalarlo sigue siendo un reto. O cómo podemos intentar hacerlo para evitar el desastre.

Cuándo

Básicamente, cada vez que nos enfrentamos a CSS, ya que gran parte de los problemas que podemos encontrar en proyectos complejos pueden afectar igualmente incluso a los más sencillos.

Links

Me interesaría especialmente hablar de cómo actualmente no es un problema resuelto y por qué estar abierto a las distintas metodologías, incluso a aquellas que inicialmente desafían radicalmente tu postura. https://en.bem.info/methodology/ (Yandex y demás), https://acss.io/ (Yahoo!), http://ecss.io/ (bet365). En mi caso, desde las trincheras, voy integrando estos conceptos en la librería Bento CSS, usado en diferentes proyectos de Doist. https://github.com/Doist/bentocss

Quién

Yeray Darias

Qué

No somos el puto Netflix!!!

Por qué

Hoy en día el diseño de referencia cuando se crea una nueva plataforma, sistema, whatever son los microservicios. Se ha convertido en el standard de facto, y los monolitos ahora sólo son meros ciudadanos de segunda que la gente mira mal cuando pasan por delante. ¿Pero de verdad necesitas microservicios? ¿Sabes cuánto cuesta construir una plataforma de este tipo? ¿Crees que escala chascando los dedos y no cuesta nada?

Cuándo

Esta técnica se aplica cuando te viene la gestión que la última línea de código que escribió lo hizo con Java 1.2 y te pide que hay que hacer microservicios.

Links

No tengo links ... por ahora, pero hay muchas referencias a esta técnica.

Quién

Juan Ignacio Sánchez Lara

Qué

Testing sobre Ruby on Rails

Por qué

Porque no queda otro remedio xD. Mantenemos la aplicación central de CARTO, responsable de Builder. Esencialmente, APIs REST de metadatos sobre las visualizaciones. Ruby on Rails (y en parte Ruby) sin tests es algo inmantenible (incluso con ellos lo es).

Cuándo

En básicamente todas las PRs, en general no damos de paso una pull request sin tests.

Links

Meh, es algo bastante estándar. Por destacar algo, usamos zeus para que arranque más rápido: https://github.com/burke/zeus

Quién

Rubén Eguiluz

Qué

Interaction driven design (IDD)

Por qué

En el equipo habíamos oído sobre esta arquitectura, y nos llamó mucho la atención. Hace un par de años empezamos un nuevo proyecto y decidimos probar esta técnica. La verdad es que la experiencia ha sido muy positiva y estamos muy contentos con esta decisión.

Cuándo

Esta técnica creo que aplicaría en cualquier proyecto backend con cierta envergadura

Links

https://codurance.com/2017/12/08/introducing-idd/ https://www.youtube.com/watch?v=n3hOS0Cj5Mc

Quién

Xabier Sáez de Ocáriz

Qué

Matriz de competencias

Por qué

Visualizar: - Qué capacidades tiene que tener el equipo en su conjunto en base a los servicios que ofrece hacia afuera (Ej: Desarrollo, gestión de infraestructuras, soporte, gestión de proveedores, etc) - Cómo están distribuido el conocimiento y las capacidades (y, por tanto, la carga de responsabilidad) dentro del equipo.

Con este trabajo en común detectamos áreas de aprendizaje, riesgos de silos de aprendizaje.

Alineamiento entre las diferentes visiones.

Cuándo

Equipo pequeño (1 PO + 3 Desarrolladores), recién formado (1 mes) con diferentes perfiles de "veteranía en el producto", desde 3 años a 1 mes. Importante un ambiente de "ausencia de juicio" (No sé cómo expresarlo mejor)

Links

- https://theitriskmanager.wordpress.com/2013/11/24/introducing-staff-liquidity-1-of-n/
- https://www.linkedin.com/pulse/20141020101636-18398652-why-your-teams-competency-matrix-may-not-match-reality-and-how-to-fix-it/
- https://management30.com/practice/competency-matrix/

Quién

Asier Ramos

Qué

"Microservicios"

Por qué

Supongo que cuando se planteo fue una buena idea, pero que acabo en un caos de microservicios con un acoplamiento muy bestia entre todos. No se deberia aplicar tecnicas de un alto nivel tecnico a equipos sin un buen nivel tecnico o sera un infierno...

Cuándo

Equipo de mas de 200 desarrolladores, era inevitable ir a esta opción

Links

https://www.dwmkerr.com/the-death-of-microservice-madness-in-2018/
https://martinfowler.com/articles/microservices.html

Quién

Pablo Albizu

Qué

Cómo trabajar con un código/equipo legado

Por qué

El objetivo era trabajar y acompañar a un equipo con un proyecto ya construido y que generaba pasta para ayudarles a mejorar la forma que tenían de construir software. Ese proceso implicaba reducir la deuda, formar a las personas del equipo sin olvidar el lanzar nuevas funcionalidades.

Cuándo

Principales técnicas de trabajo con código legado recogidas mayoritariamente en el libro Working With Legacy Code así cómo parallel change, técnicas generales de refactoring, clean code, buenas prácticas, TDD, testing, etc.

Links

https://www.amazon.es/dp/B005OYHF0A/ref=dp-kindle-redirect?_encoding=UTF8&btkr=1
https://www.youtube.com/watch?v=aKcmbOZV9mA

Quién

Jerónimo López

Qué

Profiling de la JVM

Por qué

En Nextail tenemos procesos que manejan gran cantidad de datos y los magrean mucho. Hasta ahora se había implementado los procesos para que funcionaran sin preocuparse de cuanto tiempo tardaran o cuantos recursos consumieran, y se escalaba verticalmente. Ejecutar esos procesos en local era imposible por los recursos que requería, por lo que dificultaba la ejecución de pruebas (que no test) en entornos locales para validar. Esto hacía que todo se tuviera que probar en un entorno de integración con un proceso largo de prueba y error, llegando a subir cosas a producción con errores por haberse hecho pocas pruebas. En mi empeño de poder hacer pruebas en mi máquina hice profiling de la aplicación, permitiendo reducir el consumo de memoria a 1/10 y el tiempo de proceso a 1/3, y mejorado mi salud mental :) Indirectamente ha favorecido al entorno de producción, reduciendo los tiempos de los procesos y el gasto de AWS

Cuándo

Aplicaciones con gran consumo de recursos donde se está llegando al límite del escalado vertical, o donde se intuye que hay un problema de recursos que impide mejorar el throughtput del sistema. Último recurso, no olvidemos que "La optimización prematura es la raiz de todo mal.

Links

http://www.brendangregg.com/flamegraphs.html http://hirt.se/blog/

Quién

Jordi Marti

Qué

(1) Cómo hacer apps conversacionales para Google Assistant - RIC Escape (2) Cómo hacer que un equipo trabaje con TDD (3) Aprendiendo a trabajar

en un equipo desde cero (4) Tooché - Montando un side project pero tomándomelo en serio

Por qué

(1) Quería hacer un piloto sobre qué posibilidades trae Google Assistant: ¿Qué se puede hacer con Google Home?
(2) Creo que realmente hacer TDD es complicado. Mi propio viaje personal me ha ayudado a reflexionar sobre cómo ayudar a otros equipos a hacerlo (lo he aplicado con dos equipos en la empresa, pero no significa que lo haya conseguido XD)
(3) En los últimos años hemos tenido bastantes rotaciones, y en el último año he tenido la oportunidad de enfocarme en sacar adelante, junto a otros compañeros, un equipo. Muchas reflexiones y aprendizajes en el camino.
(4) Quería hacer un side project que involucrara a mis sobrinos y además me enseñara a enfocar un producto desde cero con total independencia, y además, quería experimentar qué tracción conseguía si abría el proceso de desarrollo al mundo desde el principo.

Cuándo

(1) No tenía ni idea de qué se podía hacer. Quería experimentar y hacer algo útil con Google Assistant.
(2) Si se quiere empezar a hacer TDD; pero no sabes cómo empezar.
(3) Cuando necesitas montar el equipo, varios sois nuevos y tenéis la posibilidad de auto organizaros.
(4) Un desarrollador web que quiere probar a hacer un juego y compartir el proceso

Links

(1) https://developers.google.com/actions/dialogflow/first-app
(2) https://drive.google.com/drive/u/0/folders/0B5V6zjUzJ7TDZkZNQUd2eUFVQ0U
(3) No sé muy bien para qué es el link, pero en teoría es privado -> https://medium.com/@itortv/a-team-journey-ep-1-un-nuevo-comienzo-ccc552ef6b21
(4) https://twitter.com/itortv/status/961715774753857536

Quién

Javier Acero

Qué

(Parallel change), de Chargify a Stripe sin perder un céntimo

Por qué

Cambiar de un proveedor de pagos a otro y seguir cobrando todos los días

Cuándo

A aplicar cuando quieres cambiar la interfaz de una clase o sistema que tiene clientes actualmente y estos cambios no son compatibles con la interfaz actual del sistema. Necesitas una manera estructurada y segura para migrar los clientes de una versión de la interfaz a la siguiente y parallel change es una manera de conseguirlo.

Links

https://martinfowler.com/bliki/ParallelChange.html, http://www.tddfellow.com/blog/2016/07/31/parallel-change-refactoring/

Quién

Diana Aceves

Qué

Trabajo en remoto desde casa

Por qué

Ai dont nou güat tu sei hier

Cuándo

Ai dont nou güat tu sei hier

Links

Ai dont nou güat tu sei hier (jajajaja! sorry) :-/

Quién

Xabier Amutxastegi

Qué

Nativescript: Desarrollo de apps para web developers

Por qué

Disclaimer: Este último año esta siendo muy raro y con bastantes cambios, por lo tanto hablaré con lo que mas tiempo he estado.

El objetivo era para un cliente con una plataforma web, con desarrolladores web y un sindios de aplicaciones nativas e hibridas para iOS y Android prácticamente todas externalizadas. La idea era matar con un disparo varios pájaros: tener una base de código común para iOS y Android, que la aplicación pareciera nativa, que su equipo pudiera mantener y extender la app y poder realizar marcas blancas de la aplicación con la misma base de código.

Cuándo

Cuando tienes conocimientos web y quieres “reuitilizar’ dichos conocimientos como Javascrpt, Typescript o Angular para hacer aplicaciones que luzcan como

nativas en Android e iOS. Tengo que advertir que la vista (HTML+CSS) son conocimientos que no se pueden reciclar y hay que aprender la forma especifica con la que se hace la UI en Nativescript.

Links

https://www.nativescript.org/ https://github.com/DeviantJS/awesome-nativescript

Quién

Juan Miguel Imaz

Qué

Tecnología: React Mantra: Keep it symple

Por qué

El objetivo del mantra, mantener los proyectos “asequibles” a colaboradores.

Cuándo

Sentido común cuando tienes que trabajar con compañeros que no conoces o conocen la tecnología.

Links

Sin links

Quién

Iker Mariñelarena

Qué

Unit testing

Por qué

Por conseguir un código más flexible en el que realizar cambios no sea un problema. Por velocidad. Porque me ayuda a desacoplarme. Porque me ayuda a pensar lo que estoy haciendo.

Cuándo

Unit testing en mi casa en prácticamente todo lo que programo. A no ser que sea alguna prueba o experimento.

Links

Uuumm... lo dicho anteriormente en prácticamente cualquier caso... Pero por poner un ejemplo: http://codingdojo.org/kata/FizzBuzz/ :)

Quién

Felix Vela

Qué

Analisis y toma de decisiones

Por qué

El objetivo es tener un código no solo que sean fashion y atractivo, sino que además tengamos ciertas reglas para optimizarlo. No reinventamos la rueda, con 4 conceptos básicos el código queda con buen rendimiento y no hace falta tener mucho hierro para proyectos con cierta carga de tráfico.

Cuándo

El contexto puede ser cualquier proyecto web. Yo en concreto utilizaba nginx + php + mysql

Links

No hay link que valga, son cosas tan tontas y que no se hacen casi nunca que nadie escribe sobre ellas.

Quién

Alvaro Salazar

Qué

Kafka, Cloud Foundry and Concourse

Por qué

Kafka crear asincronía entre servicios. CF y Concourse para mejorar la productividad del equipo.

Cuándo

Aplicación legacy con "microservicios" acoplados al compartir la BBDD

Links

https://kafka.apache.org/ https://www.cloudfoundry.org/ http://www.kennybastani.com/2016/04/event-sourcing-microservices-spring-cloud.html

Quién

Sergio Jiménez Mateo

Qué

Back to the future: no siempre estar con las últimas tecnologías da la felicidad. Usando Java (Backend y Restfull), Mysql, Mongo, etc... aplicado a la Industria 4.0

Por qué

El objetivo es que funcione todo y ser eficiente y controlar todo todo, cosa que con tecnologías últimas y siempre "evolucionando" no lo podríamos tener. Al estar cerca de la industria tenemos que poder "hablar" con toda su infraestructura y bastante de ello tiene tecnología antigua pero robusta y nos tenemos que amoldar. También la fiabilidad que nos da lo que usamos es extrema.

Cuándo

Como comentaba, el contexto es fácil : Industria 4.0. No necesitamos tener las últimas tecnologías, lo que es necesario es robustez y que todo funciona 365 días sin cortes. Además, tenemos 2 partes: hardware con firmware y web. En la primera no es necesario lo último porque a veces no tienen soporte para el hardware y en la segunda hay algo más de "últimas tecnologías" pero sin pasarse.

Links

No hay links clave. Las tecnologías descritas tienen mil enlaces.

Quién

Jorge Maroto

Qué

Hemos usado bastante docker, AWS y migramos a Google App Engine. Esto ha mejorado mucho mucho nuestra calidad de vida a la hora de desarrollar :)

Por qué

Sobre todo cambiar del modelo IaaS hacia un PaaS que nos permita centrarnos en el desarrollo de producto.

Cuándo

Hubieron bastantes salidas en el equipo, y como la gente nueva necesitaba coger know how de todo, se decidió tratar de quitar complejidad a la infraestructura, ya que no contamos (ni nunca lo hemos tenido) con equipo de sistemas propiamente dicho.

Links

https://cloud.google.com/appengine/docs/flexible/

Quién

Gonzalo Ayuso

Qué

jQuery

Por qué

El título es un poco troll. La herramienta es OpenUI5. Usa jQuery en el fondo (muy en el fondo). En realidad cuando la usas no te enteras. La sintaxis es la de un fw de front moderno . El motivo era que tenía que elegir un framework de front con un ciclo de vida grande (en el 2017). Me sorprendió mucho esta herramienta. Casi nadie la conoce (lo que es malo), o al menos casi nadie que está en el último hype de js, pero si buceas encuentras una comunidad muy grande y mucha documentación (hay vida detrás del hype). Tiene por detrás a SAP, que ha metido mucha pasta en eso y la ha sacado como open source. No es tan molona como Vue o React, pero bueno, yo tampoco soy tan moderno :)

Cuándo

Desarrollo de front puro. Pero no solo framework, sino una serie de widgets o toolkits para desarrollar aplicaciones empresariales (y andar buscando en github herramientas cada una de su padre y de su madre)

Links

https://openui5.hana.ondemand.com/

Quién

Javier Anaya Martinez

Qué

React-native

Por qué

El objetivo era crear una app sin mucho esfuerzo y sencilla y que no tuviéramos que hacerlo de manera nativa para ios y android

Cuándo

Si decides hacer una aplicación móvil con javascript esta tecnología te soluciona muchos problemas. Es fácil de usar y los resultados son muy buenos.

Links

https://facebook.github.io/react-native/ https://medium.freecodecamp.org/how-to-get-started-with-react-native-8ef42f65160a?gi=bd3b5b37d27a http://www.reactnativeexpress.com/

Quién

Jose Ignacio Andrés

Qué

scrum + añadidos

Por qué

Mejorar nuestro workflow

Cuándo

Trabajo en equipo, especialmente en equipos de desarrollo

Links

paso 1: leer los capítulos que más te interesen de “scrum: the field guide” paso 2: poner en práctica desde el día 0 lo aprendido paso 3: iterar

Quién

Rodrigo Gómez

Qué

Metodología: Implantación de DDD a nivel de toda la empresa

Por qué

Me parece interesante el resultado que se llega a obtener en empresas de producto, ya que viene a resolver la desconexión que se va produciendo en los

distintos departamentos de las empresas a lo largo que los mismos van evolucionando/creciendo. También a nivel técnico, aunque no es totalmente necesario, suele ir ligado de metodologías y patrones de diseño que implican que el código y el lenguaje usado en la empresa, se exprese de la misma manera.

Cuándo

El momento de aplicar estas técnicas diría que es en el momento que empieza a ser clave, y es, el momento que la empresa/departamentos van creciendo y evolucionando, para que con el paso del tiempo no se produzcan desconexiones.

Links

Lo más conocido los dos libros, que más desarrollan este tema: https://www.amazon.es/Implementing-Domain-Driven-Design-Vaughn-Vernon/dp/0321834577 https://www.amazon.es/Domain-Driven-Design-Tackling-Complexity-Software/dp/0321125215

Quién

Rubén Bernardez

Qué

En busca de la entrega continua a través de Trunk Based Development

Por qué

Buscábamos eliminar los principales problemas que nos ocasionaban las ramas de larga duración de funcionalidades que tardaran semanas e incluso meses en ser aprobadas para pasar a producción. Aunque conscientes de que el principal causante de esa larga vida era el ciclo de vida de las historias, algo complicado de cambiar en una empresa por la tipología de empresa cliente, buscábamos tener todo el código lo más sincronizado posible (integración continua), feedback lo más rápido posible de posibles conflictos entre el código de los desarrolladores del eqipo, conseguir hacer refactorizaciones horizontales sin morir en el intento, compartir código más rápidamente, eliminar el dolor y sufrimiento que puede causar mergear ramas que llevan desincronizadas meses, mejorar la comunicación y ser algo más felices en nuestro día a día.

Cuándo

Mi experiencia a sido aplicar este método de trabajo a proyectos Front-End donde debíamos evolucionar una aplicación sprint a sprint con equipos de entre 4 a 12 desarrolladores. Bajo este contexto nos ha funcionado de maravilla.

Si que hay ciertas premisas que ayudan a potenciar la técnica como son: tener una batería automatizada de test, compilaciones automatizadas, servidor de IC con despliegues automatizados y un equipo con cierta experiencia que te permite tener el código de la rama principal de desarrollo siempre listo para pasar a producción.

Links

https://trunkbaseddevelopment.com/ https://www.martinfowler.com/articles/continuousIntegration.html

Quién

Sara Ortega Muñoz

Qué

Swift en iOS

Por qué

Este último año ha sido muy raro. He estado prácticamente sin trabajo en la empresa hasta que lógicamente fui despedida. Cuando tenía trabajo principalmente desarrollaba aplicaciones web para clientes en un entorno LAMP y con un framework propio de la empresa (!). Ahora estoy en el paro y soy feliz aprendiendo Swift en iOS

Cuándo

Para crear apps nativas en iOS frente a desarrollar en Objective-C porque Swift es el nuevo lenguaje de referencia para el desarrollo en iOS y ya sabemos como se las gasta Apple con lo viejo.

Links

https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language

Quién

Eduardo ferro

Qué

Blameless Incident Report / Bug Report

Por qué

Crear una cultura de aprendizaje compartido y generar poco a poco un sistema mas resiliente a problemas. Es una muy buena táctica para compartir aprendizaje, priorizar finalizar cosas y generar un equipo y un sistema software antifragil.

Cuándo

Se puede aplicar para caidas/incidentes en producción y para bugs no triviales.

Links

https://codeascraft.com/2016/11/17/debriefing-facilitation-guide/ https://codeascraft.com/2012/05/22/blamele postmortems/ https://www.infoq.com/news/2014/07/blameless-post-mortems

Quién

Beñat Espina

Qué

Domain-Driven Design

Por qué

Es un conjunto de técnicas y patrones para hacer un mejor código muy alineado con los requisitos de negocio. Trabajo en una empresa muy orientada al marketing y trabajar bien alineados con los requisitos de negocio nos hace ser más ágiles.

Cuándo

Este enfoque aplica cuando tenemos una alta carga de requisitos funcionales, si nuestro producto no contiene mucha lógica de negocio, Domain-Driven Design puede implicar una sobreingeniería injustificable.

Links

https://martinfowler.com/tags/domain%20driven%20design.html https://www.youtube.com/watch?v=dDofYA

Quién

Sergio Sáenz

Qué

Spring Boot

Por qué

En realidad eran varios objetivos. Lo he usado en la cárnica en la que trabajaba a diario durante 4 años. Al ser una empresa orientada a servicios me ha tocado pelearme con muchos entornos distintos. Siempre con mucha indefinición y mucho cámbio de última hora. El objetivo realmente ha sido la versatilidad y rapidez de desarrollo así como la facilidad de integrar nuevos perfiles en poco tiempo de arranque. Hay suficiente comunidad y librerías en el ecosistema Spring Boot como para facilitar integraciones de casi cualquier tipo. Desde una base de datos a un REST pasando por un sistema de colas o incluso websocket.

Cuándo

Buff... esta es difícil... yo lo intento usar para casi todo. Pero solo en entornos web o entornos de escritorio (casos puntuales concretos). Yo no lo veo como buena herramienta para sistemas embebidos. En ese caso he tirado más por un motor de javascript y pantallas html5. Quizás tampoco sería mi primera opción para una aplicación de escritorio pura, sin web en ningún sentido. Habría que darle unas vueltas. Para scripting tampoco le veo mucho sentido. Básicamente fuera del mundo web o de integración habría que analizar el caso despacio.

Links

https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/ https://spring.io/guides/gs/spring-boot/ http://www.baeldung.com/

Quién

Fernando Fariña

Qué

RxJS

Por qué

El objetivo era migrar nuestro framework de cabecera, de AngularJS a Angular. Como principal novedad, además de TypeScript, Angular ofrece/invita/obliga a utilizar programación reactiva y poder así sacar máximo partido al flujo de datos dentro de la aplicación.

Si bien el manejo de Observables se hace duro en un principio, la flexibilidad que permiten a la hora de manipularlos hace que muchas marcianadas que había que hacer en AngularJS se hagan de manera mucho más elegante con Angular.

Cuándo

Las utilizamos para sacar partido de los formularios reactivos, peticiones HTTP a servidor, escucha de eventos, cambio de valores de Query Params.

Links

http://reactive.how/ https://github.com/ReactiveX/rxjs/blob/master/doc/pipeable-operators.md

Quién

Irune prado

Qué

Dando cobertura a Google Search Appliance mediante productos Open: Apache Nutch y SOLR

Por qué

Tras el corte de servicio de Google de su search appliance, se requería unificar varias páginas webs de distintos dominios (intranet, extranet, subdominios, ...) , tecnologías (HTML, JSP, ASP), naturalezas (con trabajo de SEO, sin SEO, con iframes, flash...) y tipos de documentos (html, doc, pdf, ...) en un mismo buscador con capacidad avanzadas, para el usuario del portal corporativo.

"Una solución para gobernarlos a todos. Una solución para encontrarlos, una solución para atraerlos a todos y atarlos en las tiniebla.... digo pantalla de búsqueda avanzada."

Cuándo

Apache Nutch es una solución de crawleo construida sobre Java que nos puede servir para extraer información "pseudo-homogénea" de diferentes orígenes / semillas, y mediante un esquema de conversión, converger su información en un sistema de índices, como puede ser SOLR

Links

http://nutch.apache.org/ http://lucene.apache.org/solr/ https://www.zylk.net/en/web-2-0/blog/-/blogs/como-configurar-nutch-1-13-para-que-use-solrcloud-6-6-0

Quién

Nestor Salceda

Qué

Arquitectura Hexagonal + Outside In

Por qué

Huir del framework.

La velocidad con la que cambia la tecnología es diferente a la velocidad a la que cambian los negocios. ¿Recuerdas cuando salió el último framework Javascript? Compáralo ahora con cuando cambió el IVA en este país.

Muchas veces nos vemos forzados a rehacer trabajo por tener que estar a la último con un determinado framework web. ¿Recuerdas todo el valor aportaste al migrar de Rails 2 a Rails 3?

Empezar por el valor de negocio, y después conectarlo a los mecanismos de entrega.

Cuándo

Desarrollo de aplicaciones en general.

Links

https://herbertograca.com/2017/11/16/explicit-architecture-01-ddd-hexagonal-onion-clean-cqrs-how-i-put-it-all-together/

Quién

Dani Latorre

Qué

IDD con arquitecturas limpias / Cucumber como herramienta de colaboración / Desacoplando de tu framework JS favorito

Por qué

Las 3 cosas son básicamente para hacer mejor software y software mejor.

Cuándo

En proyectos hechos "para que duren". Que requieran mantenimiento en el tiempo, flexibilidad/cambiabilidad... Supongo que en "cualquiera" de cierta envergadura o prototipos rápidos

Links

No tengo muy claro qué links poner, así que aquí van un par sobre el que se sustentan los 2 primeros temas. http://www.ustream.tv/recorded/61480606 https://gojko.net/books/specification-by-example/

Del tercero no sé, más allá de las charlas que hicimos con Néstor y Gualis XD

Quién

Alberto (APA)

Qué

Practica con katas enfocadas a tus necesidades

Por qué

Para practicar python + TDD con dobles Además, generas una utilidad para al equipo

Cuándo

Usadas cuando los nuevos miembros del equipo no están en pair con otros miembros.

Links

Se puede hacer en solitario o en parejas. Damos tiempo en el horario laboral para realizarlo, aunque se empezó como un extra de las vacaciones :-)

Quién

Itziar López

Qué

Generación y lectura de códigos de barras mediante una app móvil para gestionar los productos de un almacén.

Por qué

El objetivo era generar y mantener actualizado el stock de productos de un almacén y poder llevarlo a cabo mediante el escaneo de los códigos de barras de los productos, de cuya generación también nos ocupábamos nosotras. El sistema de gestión en el que se registran dichos movimientos de almacén está desarrollado con symfony y la app mediante la que se escanean los códigos de barras de momento sólo la tenemos terminada para iOS (Objective-C) y estamos desarrollándola para Android.

Cuándo

Nos plantemos comprar un lector de códigos de barras para que los escaneos funcionaran de la forma más eficiente posible pero finalmente nos decantamos por desarrollar una app móvil para ello para poder comunicarnos con el sistema de gestión mediante una API propia y también para que estos escaneos pudieran realizarse fuera del almacén ya que, a veces, parte del stock sale a ferias, etc. De esta manera, desarrollamos un sistema y su comunicación a medida de nuestras necesidades específicas.

Links

Nos plantemos comprar un lector de códigos de barras para que los escaneos funcionaran de la forma más eficiente posible pero finalmente nos decantamos por desarrollar una app móvil para ello para poder comunicarnos con el sistema de gestión mediante una API propia y también para que estos escaneos pudieran realizarse fuera del almacén ya que, a veces, parte del stock sale a ferias, etc. De esta manera, desarrollamos un sistema y su comunicación a medida de nuestras necesidades específicas.

Quién

Usue Napal

Qué

BLE en iOS

Por qué

El trabajo con BLE se distribuyó entre dos proyectos. En uno el objetivo era la comunicación entre la app de iOS y unos mandos copiadores para realizar copias de mandos de garaje. En la otra era la comunicación entre la app de iOS y Beacons, que permitían la distribución de ofertas personalizadas por usuario.

Cuándo

En ambos proyectos la tecnología bluetooth encajaba perfectamente por el modelo de negocio.

Links

https://developer.apple.com/bluetooth/ https://developer.apple.com/ibeacon/

Quién

Alex Epelde

Qué

Primeros pasos con Vue + Vuex

Por qué

El objetivo era “modernizar” (no sé si es el término adecuado) partes de la interfaz de una aplicación “antigua”.

Cuándo

Intentar traer un poco de “cordura” y homogeneidad al desarrollo front de la aplicación. Reutilizar código.

Links

https://vuex.vuejs.org/en/installation.html https://medium.com/coding-stones/abstracting-vuex-redux-action-patterns-8df36b0e2fcc

Quién

Guille ([@ggalmazor](http://twitter.com/ggalmazor))

Qué

Chugen Krüngen

Por qué

chungenizar los krungens

Cuándo

All the time

Links

hinkli

Quién

David

Qué

Integración continua basada en git+jenkins

Por qué

Un equipo que empezó a crecer en una empresa que por su contexto no podía acotar el alcance de las funcionalidades. Muchos proyectos que se abrían y cerraban, cargas de trabajo que variaban de un proyecto a otro, un día todo el equipo en un proyecto y dos días después cada miembro de equipo con un proyecto distinto. Nos proporcionó mucha agilidad.

Cuándo

Supongo que en el momento en el que hay más de un desarrollador aunque he de reconocer que hay escenarios en los que me parece que se podría estar igual sin ello. He estado en un equipo que éramos dos y desarrollábamos una app android ¿Que sentido tiene un Jenkins cuando los dos podemos hacer "botón derecho->generar release" a partir de un hash de git?

Links

https://www.amazon.com/Phoenix-Project-DevOps-Helping-Business/dp/0988262592 (es una novela pero creo que explica claro como el agua la finalidad de la

integración continúa, que es agilizar el delivery y que hay que mantener el foco siempre en el cuello de botella) https://continuousdelivery.com (si estás empezando puede ser un buen punto) https://jenkins.io (para empezar y es el servidor de integración más extendido)

Quién

Álvaro García Loaisa

Qué

Subcontratar servicios que te hacen la vida más fácil

Por qué

Definir una arquitectura que nos ayudase a solucionar un problema complejo de la manera más automatizada posible. Teníamos un API con unos datos que se actualizaban de una vez, con un preprocesamiento bastante pesado. Teníamos el requisito de mantener en modo backup las anteriores cargas de datos para tenerlas disponibles al momento por si la carga nueva tenía algún problema. Teníamos el añadido de que el tiempo de respuesta fuera ínfimo, cosa que solucionamos con una carga en memoria de todos los datos procesados (64GB de Ram de datos...)

Para solucionar todo esto usamos varios servicios de AWS como Lambdas, ECS, Cloudformation, y el sdk para Java. Nos atamos muchísimo a AWS pero el cliente quedó encantado con la solución y su rendimiento.

Cuándo

Cuando al cliente no le importe atarse a una tecnología concreta (AWS en este caso) y acepte el gasto de esta. Si el equipo es pequeño y los tiempos de entrega cortos.

Links

https://aws.amazon.com/es/ https://cloud.google.com/ https://azure.microsoft.com/es-es/

Quién

Jimena Martínez

Qué

"De la necesidad del usuario a la feature request"

Por qué

Durante los últimos años en mi vida profesional siempre he tenido presente la problemática de "entender realmente lo que necesita el usuario", con el objetivo de trasladarlo al equipo de producto, de encontrar oportunidades de negocio o de mejorar los procesos internos de la empresa. En mi actual trabajo, parte de mi cometido es buscar soluciones que hagan ese "camino de la información" más fácil, sencillo y entendible por todos: desde cómo obtener la información adecuada del usuario, hasta cómo el equipo de desarrollo interpreta esa información, pasando por cómo se comunica y transfiere internamente (equipos de soporte, soluciones, customer success, etc.) Esto requiere coordinar y alinear las perspectivas de diferentes personas con una visión muy diferente del mismo producto, de las necesidades y experiencia del usuario y de las potenciales oportunidades de negocio.

Cuándo

Esta "idea" es algo que está presente durante todo el ciclo de venta (incluyendo pre y post venta) y asimilación posterior en el roadmap de producto. Creo que debería ser un proceso transversal que estuviera presente en cada deal, release o modificación de un producto que interactúe con usuarios. Ahorraría recursos, mediocre UX y churn.

Links

De momento estamos explorando con una combinación de Salesforce, Supportbee, GitHub, GDrive. También este link interesante: https://m.signalvnoise.com/a-new-approach-to-feature-requests-21bea562c083 Pero aún estoy en ello ;)

Quién

Rafa Fernández

Qué

Making FrontEnd Decisions

Por qué

Porque en los últimos años, en todos los proyectos de tipo Web app donde he participado han existido problemas de modularidad y de cancelación de llamadas asíncronas. El objetivo era desarrollar un SaaS basado en componentes con una fuerte carga de APIs. La arquitectura para este proyecto está basada en React, Redux y la utilización de observables (RxJs) que permitieran la cancelación de las llamadas.

Cuándo

Explotación de un SaaS basado en lllamadas APIs que calculan indicadores inmobiliarios en el back utilizando machine learning y una fuerte componente geográfica. Para esto último, se utiliza la extensión PostGIS en PostGreSQL

Links

www.pulse.urbandataanaytics.com

Quién

Julen

Qué

Actualizacion de AngularJS y tooling

Por qué

Para poder utilizar una arquitectura más enfocada a componentes y que el proyecto sea más coherente

Cuándo

Cuando se hereda un proyecto AngularJS legacy y se pretende actualizarlo para acercarse más a enfoques desarrollo por componentes como Angular, React o Vue

Links

Por un lado leyendo a fondo la documentación de AngularJS y por otro basándose en las buenas prácticas y guías de estilo como por ejemplo https://github.com/johnpapa/angular-styleguide https://github.com/toddmotto/angularjs-styleguide

Quién

Agustín Herranz

Qué

Pipeline para generar imáges (AMIs) de maquinas virtuales.

Por qué

Automatizar y hacer repetible, y testeado, la personalización de imágenes sobre las que crear maquinas virtuales. También el tardar menos y ser más agiles a la hora de actualizar dependencias y sistemas. Es la típica 'bakery' con packer. En mi caso en AWS con AMIs, usando Ansible u testinfra para el testing, pero también se puede hacer con otros provedores Cloud y/o hypervisores, y con otras maneras para hacer Configuration manegement.

Cuándo

Si queremos usar infraestructura como código, y tratamos de acercarnos a una infraestructura inmutable necesitamos un pipeline que se encargue de construir y testear los artefactos necesarios, en este caso las imágenes (AMIs/plantillas) sobre las que inicializar máquinas virtuales.

Links

https://www.packer.io/guides/packer-on-cicd/index.html https://thenewstack.io/bakery-foundation-container-images-microservices/ https://keithmsharp.wordpress.com/2017/06/12/devops-on-aws-building-an-ami-bakery/

Quién

Marta Manso de las Heras

Qué

Badass Product Owning

Por qué

He visto a gente flaquearle las piernas hablando con el cliente, como si fuera un ente superior... no sé por qué. En mi curro anterior era developer pero trataba con cliente. Un cliente tocho y con motivos pérdidas de millones por los que quejarse si algo no funcionaba. Así que... eran intensitos. Como desarrolladora es fácil que el cliente te toque los pies con peticiones absurdas... así que te apetecía ponerte un poco badass con ellos cuando no era lógico lo que pedían... Nunca vi al cliente como un ente intocable si como a un compañero a veces complicado de gestionar. Y me ha ido realmente bien... soy super paciente, super respetuosa y educada... pero si juntas eso con no dejar que traspasen tus límites, en mi experiencia te respetan aún más y la relación mejora muchísimo. Así que por si alguien aún no lo hace así... os puedo contar mis formas :)

Cuándo

Aplica si hablas mucho con el cliente... Si eres parte de la decisión de qué es más importante que qué o qué debe salir antes o en qué hay que invertir esfuerzo. Aplica si el cliente a veces se sale con la suya y acabas comprometiendo a tu equipo a hacer algo que no querías o en un tiempo que tú no hubieras puesto o de una manera que no os hace sentir orgullosos.

Links

Es de cosecha propia :) Pero una buena educación sobre lo que hace o cómo prioriza un product owner también puede aplicar. https://vanesatejada.com/2015/03/22/herramientas-de-priorizacion-no-porque-entonces/

Quién

Fran Mosteiro

Qué

Empoderar a tu equipo técnico. El "Método Hulk"

Por qué

Trabajas en un equipo eminentemente técnico. Sus capacidades como desarrolladores de software, son muy buenas, sobresalientes, incluso. Dominan el backend, el frontend y no tienen mayor problema por enfrentarse a nuevos retos técnicos cada día. Pero ..... necesitan "que les den permiso" o que les marquen la dirección en cuanto a tomar decisiones de negocio, de arquitectura o incluso "personales" se refiere.

¿Cómo deshacer ese nudo gordiano? ¿Cómo conseguir que todo el equipo tenga una responsabilidad común sobre? - El producto que desarrollan - La infraestructura que los soporta - Los problemas (incluso personales) del equipo - Decir que sí y decir que no ante debates internos importantes para el equipo/negocio - Participar de las decisiones del equipo como si el negocio fuese suyo - Verse cómo algo más que músculo técnico

En definitiva, cómo hacer que desarrolladores con experiencia pero con sesgos/prejuicios y experiencias en consultoras al uso, consigan verse como un equipo y empoderarlos para que actúen como algo más que programadores.

Tarea ardua y complicada ... si no fuese porque es el "Método Hulk" =)

Cuándo

- Ante una situación en la que tu empresa pretende que su equipo técnico se convierta o simplemente sea parte core o importante de tu negocio.
- Si pretendes que tu equipo técnico de un paso más allá a lo meramente técnicos se imbuya también en el negocio. Se implique en el día a día de la toma de decisiones relevantes y se sienta parte de ellas.
- Si crees que cuanta más gente aporta a tu negocio (contextualizado cuando corresponda) mejor será la solución que portas al mercado. Menos sesgada.

Links

https://www.youtube.com/watch?v=t_z_-aHIFWI&t=26s&index=7&list=PLKxa4AIfm4pVVRXkUvrYNB_e
https://www.youtube.com/watch?v=xCu_T0iRvsA

Quién

Gorka Moreno

Qué

Python

Por qué

- Aprender una nueva tecnología
- Ampliar negocio
- Cumplir la necesidad del cliente trabajando en el idioma que domina.
- Implementar prácticas conocidas en un nuevo lenguaje (testing, modularización, análisis estático de código, gestores de paquetes, etc)
- Ver nuevos patrones en nuevos idiomas (generators, listas en línea, etc, funciones de primer nivel, entornos virtuales, etc.)

- Salir de la zona de confort :)
- Realizar web scraping (Scrapy)
- Captar un nuevo y potencial cliente

Cuándo

- Web scraping (framework scrapy es de los más usados)
- Multiplataforma y portable
- Multiproposito
- Aprender un nuevo lenguage

Links

- Python koans: https://github.com/gregmalcolm/python_koans
- Curso codeacademy: https://www.codecademy.com/learn/learn-python
- Guía python: http://docs.python-guide.org/en/latest

Quién

Javier Gamarra

Qué

Diseño de APIs públicas para librerías, funcional con java, hypermedia, kotlin...
Me quedo con diseño de APIs -> API-Tradeoffs

Por qué

API-Tradeoffs -> crear una librería fácil de usar por desarrolladores, tipada, guiada y que no exponga internals pero al mismo tiempo sea expresiva y me permita crear APIs hypermedia. El objetivo es balancear economics: que el desarrollo de la librería sea rápido y reaccione a cambios de prioridad y necesidades de equipos pero al mismo tiempo sea fácil de usar y tipada. Con restricciones: soportar APIs existentes, Java...

Cuándo

Aplica a la hora de diseñar una librerías/APIs.

Links

Pasapalabra :)

David - Integración continua basada en git+jenkins

Juan Ignacio Sánchez Lara - Testing sobre Ruby on Rails

Jerónimo López - Profiling de la JVM

Álvaro García Loaisa - Subcontratar servicios que te hacen la vida más fácil

Jerónimo López - JFleet: Construyendo el mapeador de BD más rápido del mundo

Marta Manso de las Heras - Badass Product Owning

Alvaro Salazar - Kafka, Cloud Foundry and Concourse

Sergio Jiménez Mateo - Back to the future: no siempre estar con las últimas tecnologías da la felicidad. Usando Java (Backend y Restfull), Mysql, Mongo, etc... aplicado a la Industria 4.0

Joaquin Engelmo Moriche - ¿De verdad eran necesarios los microservicios?

Javier Gamarra - Diseño de APIs públicas para librerías, funcional con java, hypermedia, kotlin... Me quedo con diseño de APIs -> API-Tradeoffs

Joaquin Engelmo Moriche - Software para el bien común

Iker Mariñelarena - Unit testing

Diana Aceves - Flexbox

Diana Aceves - Trabajo en remoto desde casa

Sergio Sáenz - Spring Boot

Imanol Pinto - Kafka y kafka streams

Joaquin Engelmo Moriche - Transformaciones digitales globales, ¿qué puede salir mal?

Javier Acero - (Parallel change), de Chargify a Stripe sin perder un céntimo

Beñat Espina - Domain-Driven Design

Fernando Fariña - RxJS

Jose Ignacio Andrés - scrum + añadidos

Endika García - Zeplin, o Demos

Fernando Gil - Todo tecnosexualidad: serverless, docker, devops

Yeray Darias - No somos el puto Netflix!!!

Joaquin Engelmo Moriche - Mamá ahora soy recruiter

Rubén Eguiluz - Interaction driven design (IDD)

Joaquin Engelmo Moriche - La formación dentro y fuera de la empresa

Jordi Marti - (1) Cómo hacer apps conversacionales para Google Assistant - RIC Escape (2) Cómo hacer que un equipo trabaje con TDD (3) Aprendiendo a trabajar en un equipo desde cero (4) Tooché - Montando un side project pero tomándomelo en serio

Irune prado - Dirty Little Things

Nestor Salceda - Arquitectura Hexagonal + Outside In

Jorge Maroto - Hemos usado bastante docker, AWS y migramos a Google App Engine. Esto ha mejorado mucho mucho nuestra calidad de vida a la hora de desarrollar :)

Xabier Amutxastegi - Nativescript: Desarrollo de apps para web developers

Fran Mosteiro - Empoderar a tu equipo técnico. El “Método Hulk”

Rubén Bernardez - En busca de la entrega continua a través de Trunk Based Development

Rafa Fernández - Making FrontEnd Decisions

Eduardo ferro - Ultimamente estoy usando (no se si correctamente) A3 thinking de Toyota

Felix Vela - Analisis y toma de decisiones

Jimena Martínez - “De la necesidad del usuario a la feature request”

Juan Miguel Imaz - Tecnología: React Mantra: Keep it symple

Pablo Albizu - Cómo trabajar con un código/equipo legado

Agustín Herranz - Pipeline para generar imáges (AMIs) de maquinas virtuales.

Asier Ramos - “Microservicios”

Usue Napal - BLE en iOS

Rodrigo Gómez - Metodología: Implantación de DDD a nivel de toda la empresa

Guille - Chugen Krüngen

Sara Ortega Muñoz - Swift en iOS