Quién

dgonzalezdiez@gmail.com # 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

juanignaciosl@gmail.com # 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

kinisoftware@gmail.com # Qué Software para el bien común # Por qué Siempre he querido ayudar a los demás, me he metido en mil movidas a título personal pero incluso me he involucrado en cosas para aportar a la sociedad a título profesional. Es un tema que siempre tengo en la cabeza, que siempre vuelve y vuelve y que siempre me deja la misma sensación: no hago suficiente. Pero, ¿qué podría hacer? ¿cómo podría hacerlo? Creo que vivimos momentos donde nuestras capacidades son esenciales y pueden marcar muchas diferencias pero, a la misma vez, no quiero rechazar ciertas comodidades y ventajas que me da mi posición actual. ¿cómo se hace eso? ¿el bien común siempre exige sacrificios? ¿qué caminos tengo?

Llevo tiempo evaluando opciones y caminos y el objetivo es compartir con vosotros lo que he andado, saber si más gente está en mi situación y ver que se podría hacer para el bien común desde nuestra profesión. # Cuándo No tengo claro como responder a esto en esta propuesta :D # Links - https://www.youtube.com/watch?v=bSBh6Cm2Hpg - https://www.youtube.com/watch?v=n-l4vSVJkNY - Libro > Doing Good Better: How Effective Altruism Can Help You Make a Difference - https://hackforgood.net/ - https://app.code4socialgood.org/ - https://github.com/codeforspain - http://populate.tools/ Un favor, que igual no sirve para nada pero, a mi propuesta de "Código por el bien común" o algo así, le puedes añadir este enlace? https://youtu.be/L6iu9bAz5i8

Quién

Iker@540deg.com # 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

d.aceves@gmail.com # Qué Flexbox # Por qué Pues aunque suene triste, el 80% de las veces que lo he utilizado ha sido para centrar elementos en vertical y horizontal. Se prometía como la solución a todos nuestros problemas a la hora de resolver layouts modernos, y realmente nos solucionó unos cuantos quebraderos de cabeza, pero siendo honestos se utiliza más para centrar que para crear realmente layouts complejos. Con la llegada de Grid Layout se va a complementar muchísimo pero, igual se va a encontrar un poco apartado como "solucionador de layouts" y cada vez más será un "centrador" de cosas? mmm... Eso sí, no hay un solo día que no lo utilice. # Cuándo Maquetación web moderna # Links https://css-tricks.com/snippets/css/a-guide-to-flexbox/ https://developer.mozilla.org/en-US/docs/Web/CSS/flex

Quién

d.aceves@gmail.com # 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

ssaenz010@gmail.com # 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

ipintoc@gmail.com # 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

kinisoftware@gmail.com # Qué Transformaciones digitales globales, ¿qué puede salir mal? # Por qué En el último año he vivido intensamente la "transformación digital a nivel global" de una empresa como Telefónica (y lo que aún queda por hacer). Esto ha incluido trabajar con tres países diferentes, APIs, requisitos, etc. Toda una fiesta. Nunca he sido "sólo un programador" y me he preocupado por muchas áreas de los proyectos. El objetivo de esta charla es compartir las cosas que nos han pasado, como hemos solucionado algunas, como hicimos bien otras y como aún no conseguimos dar con la tecla en el resto. Aquí he venido a aprender.

Una tragicomedia de Antena 3 un sábado por la tarde :D # Cuándo Lidiar con third-parties es una movida y se necesita una mezcla de: sinceridad, comunicación, transparencia, saber decir NO, saber escuchar y saber entender al otro. Además la organización de equipos es esencial y detectar los roles claves y como trabajar con remotos y distintos usos horarios.

No son "técnicas a nivel técnico" :D sino acciones que hemos ido tomando e iterando sobre el trabajo en cada proyecto para poder hacer el día a día mejor. # Links No tengo :D

Quién

j4cegu@gmail.com # 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

benatespina@gmail.com # 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=dDofYAOkpts&t=6592s

Quién

fernandofarinagarmendia@gmail.com # 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

jerolba@gmail.com # 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

joserobleda@gmail.com # 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

endika@baturamobile.com # 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

fgilg55@gmail.com # 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

ydarias@gmail.com # 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

kinisoftware@gmail.com # Qué Mamá ahora soy recruiter # Por qué En los últimos dos años estoy participando activamente en los procesos de contratación de Tuenti, pasando de la oferta de trabajo que publicamos, los pasos del proceso, el contenido de cada uno y mi papel como entrevistador en distintas fases. Me gustaría contar como lo hacemos nosotros, los resultados que nos da y escuchar como lo hacen otras personas y contrastar ideas. # Cuándo Técnicas que podría contar: - Code tests (ejercicios y puntos a mirar) - Entrevistas telefónicas & skype - Entrevistas presenciales - Pizarras? - Problemas y contenido de las entrevistas - Contenido de las ofertas - Salario vs No Salario publicado # Links No tengo :D

Quién

regiluze@hotmail.com # 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

kinisoftware@gmail.com # Qué La formación dentro y fuera de la empresa # Por qué Creo que estamos en una profesión exigente y, además, nosotros lo

empeoramos más con nuestra obsesión por el aprendizaje continúo. No digo que sea algo malo el aprender, me parece algo imprescindible, pero la forma en la cuál llevamos a cabo ese aprendizaje puede no ser la más saludable, no sólo personalmente sino también profesionalmente. El objetivo de la charla es compartir algunas prácticas que llevo años aplicando en varias empresas, la mayoría en mi paso por Tuenti y que han tenido buenos resultados. # Cuándo Estas técnicas aplican para todas aquellas empresas y para todas aquellas personas que busquen un equilibrio entre la formación continúa, el día a día laboral y la vida personal. Toda empresa, ya sea producto o projecto, ya sea consultora o no, ya sea de la naturaleza que sea, creo que se puede beneficiar. # Links Autobombo - https://medium.com/makingtuenti/creating-a-culture-of-learning-at-tuenti-da810d2e6623 - https://www.youtube.com/watch?v=-x127e6T2Zg

Quién

jmarti@theinit.com # 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

wideawakening@gmail.com # Qué Dirty Little Things # Por qué Así como hay entradas de blogs de “life hacks”, en nuestro día a día laboral también tenemos nuestros “work hacks” que rara vez compartimos con los demás, salvo que alguien venga a nuestro sitio y en medio de una explicación nos suelte un “Cómo has hecho eso?” o “Qué es eso?” o “Cómo haces tú para XX?” El objetivo sería sacar una lista de estas pequeñas cosas que nos dan gustito y queramos compartir con los demás. Minimezes que hemos adoptado y se han quedado en nuestra navaja suiza.

No es objetivo hacer un monólogo de ello (yo al menos creo que no tengo recursos para hablar largo y tendido), si no generar un ambiente en donde de forma round-robin hablemos de algo que nos guste, no veamos aplicarlo en nuestro entorno de forma habitual y queramos compartir. # Cuándo Aplican al trabajo del día a día, pero depende como se enfoque puede variar desde; organización de trabajo, de mails, gestión del tiempo, hacks de sistema operativo, de shell, rutinas de trabajo en equipo, depuración, automatización, ...
Links Pues molaría tener enlaces de estos, tipo LifeHack pero ITHacks, un @hackoftheday o algo, pero yo al menos no conozco x)

https://lifehacker.com/5743814/become-a-command-line-ninja-with-these-time-saving-shortcuts https://stedolan.github.io/jq/ https://github.com/robbyrussell/oh-my-zsh

Quién

nestor@nestorsalceda.com # 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

loaisa@gmail.com # 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

patoroco@gmail.com # 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

xabiamu@gmail.com # 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

framosval@gmail.com # 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_eYn0wWAa3r https://www.youtube.com/watch?v=xCu_T0iRvsA

Quién

rubenbp@gmail.com # 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

rafael.fernandez.gis@gmail.com # 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

eduardo.ferro.aldama@gmail.com # Qué Ultimamente estoy usando (no se si correctamente) A3 thinking de Toyota # Por qué Al entrar a Nextail (y al final de la época de TheMotion) tenia que realizar cambios, mejoras, eliminar problemas que eran dificiles de planificar y que eran de larga duración (eliminar un bottleneck en el sistema, mejorar una práctica, motivación, etc) y los A3 me ayudan a poder tener una visión clara mientras voy dando pasos pequeños. Dejando muy claro en el proceso que NO tenemos ni puta idea y que todo son experimentos. # Cuándo Cambio grande en duración / trasversal / difuso

y dificil de medir. Artefacto alineado con Lean / systerm thinking / Lean startup y con el craftmanship. # Links https://www.crisp.se/gratis-material-och-guider/a3-template https://en.wikipedia.org/wiki/A3_problem_solving https://www.toolshero.com/problem-solving/a3-thinking/

Quién

felixvs@gmail.com # 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

jimenamartine@gmail.com # 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

imazjm@gmail.com # 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

pablo@540deg.com # 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

jerolba@gmail.com # Qué JFleet: Construyendo el mapeador de BD más rápido del mundo # Por qué Cuando como resultado de tus procesos tienes que persistir millones de registros en BD, tu ORM o herramienta de persistencia de base de datos se convierte en un cuello de botella. JFleet te permite escribir un stream/collection muy grande de Objetos en MySQL o PostgreSQL usando la técnica de persistencia más rápida que proporciona cada uno, usando sólo las anotaciones de JPA para mapear tus entidades a tablas (sin usar ninguna implementación JPA) ¿Por qué podría interesar hablar de la herramienta? Por que la he hecho yo :D https://github.com/jerolba/jfleet https://github.com/jerolba/jfleet-benchmark # Cuándo Cuando tienes un montón de datos que escribir que en una tabla (a partir de millares de registros ya empieza a notarse la diferencia) Y es para Java :D # Links Por ahora sólo tengo publicado esto: https://github.com/jerolba/jfleet https://github.com/jerolba/jfleet-benchmark Quiero escribir un post para hablar de la herramienta y darla a conocer, pero por ahora estoy trabajando en depurarla y documentarla.

Quién

agustin.herranz@gmail.com # 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

uthopiko@gmail.com # 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

usue@usitdevelopment.com # 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

rodri.rgm@gmail.com # 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

ggalmazor@gmail.com # Qué Chugen Krüngen # Por qué chungenizar los krungens # Cuándo All the time # Links hinkli

Quién

s.ortegamunoz@gmail.com # 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

ladymm@gmail.com # 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

salazar3pa@gmail.com # 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.jimenez.mateo@gmail.com # 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

kinisoftware@gmail.com # 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

nhpatt@gmail.com # 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 :)

/experiencias/0.html

/experiencias/1.html

/experiencias/2.html

/experiencias/3.html

/experiencias/4.html

/experiencias/5.html

/experiencias/6.html

/experiencias/7.html

/experiencias/8.html

/experiencias/9.html

/experiencias/10.html

/experiencias/11.html

/experiencias/12.html

/experiencias/13.html

/experiencias/14.html

/experiencias/15.html

/experiencias/16.html

/experiencias/17.html

/experiencias/18.html

/experiencias/19.html

/experiencias/20.html

/experiencias/21.html

/experiencias/22.html

/experiencias/23.html

/experiencias/24.html

/experiencias/25.html

/experiencias/26.html

/experiencias/27.html

/experiencias/28.html

/experiencias/29.html

/experiencias/30.html

/experiencias/31.html

/experiencias/32.html

/experiencias/33.html

/experiencias/34.html

/experiencias/35.html

/experiencias/36.html

/experiencias/37.html

/experiencias/38.html

/experiencias/39.html

/experiencias/40.html

/experiencias/41.html

/experiencias/42.html

/experiencias/43.html

/experiencias/44.html

/experiencias/45.html