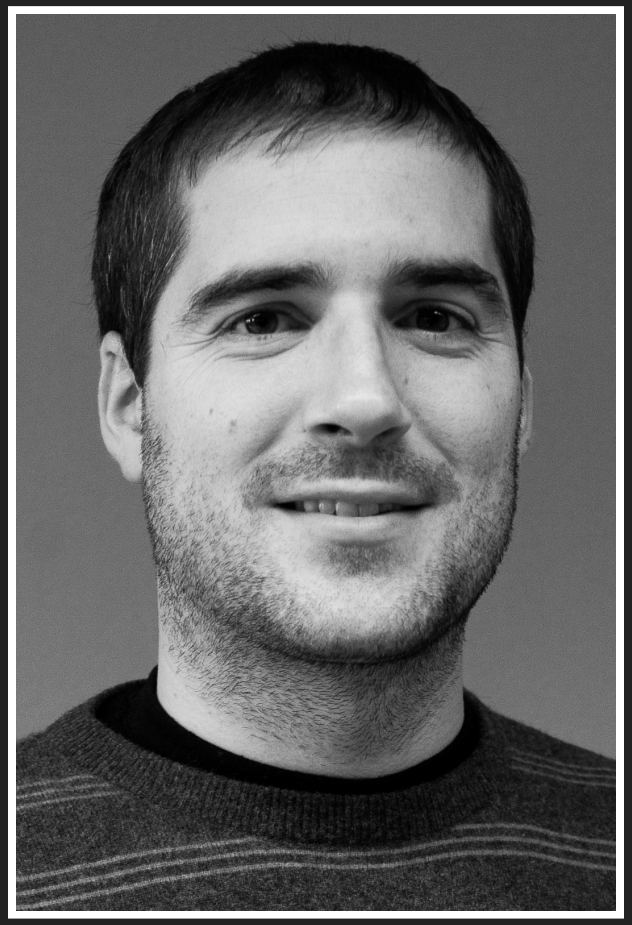
ARQUITECTURA DE DESARROLLO EN PROYECTOS DRUPAL

DRUPALCAMP SPAIN 2015

Por Ramon Vilar / @rvilar

SOBRE MI



Ramon Vilar

@rvilar

http://ymbra.com

ymbra
drupal dreamers

ÍNDICE

EL PROCESO DE DESARROLLO IDEAL

AS A DEVELOPER, I WOULD LIKE...

Como equipo de desarrollo, si hiciésemos una carta a los reyes, nos gustaría:

- Código versionado
- Despliegues automatizados
- Pruebas automatizadas
- Fácil de probar manualmente
- Que todo el equipo tenga conocimiento de todo el código
- Fácilmente mantenible

Y todo *de forma transparente a nosotros*

ESTO ES UNA UTOPÍA?

Puede parecerlo, pero no! Nosotros lo hacemos

Lo primero, y lo más importante de todo, es:

- Preparar al equipo para el cambio
- Tener complicidad y ganas por parte de todos los implicados
- No tener prisa en la adopción
- Ir paso a paso

BUENAS PRÁCTICAS INICIALES

CONOCIMIENTOS NECESARIOS PARA SEGUIR

- Conocemos y usamos Git
- Conocemos y usamos Drush
- Conocemos y usamos Features
- y ...

CUÁL ES NUESTRA FINALIDAD

- Nuestro proyecto debe poderse instalar de un entorno a otro sin requerir ninguna tarea manual
- Esto implica trabajar siempre con código
- Toda configuración y funcionalidad debe residir en el código y no en base de datos (features y código de instalación)

ESTRUCTURA DE DIRECTORIOS

Seguimos una mínima estuctura para nuestros módulos y temas:

- `/contrib`
- `/custom`

Y tenemos nuestras features dentro de un directorio `/features`. Este también lo podemos estructurar a su vez:

- `/content_types`
- `/general`
- `/entity_types`

ENCAPSULAR EL DESARROLLO

- Debemos pensar en una forma de encapsular nuestro desarrollo y hacer fácil el flujo
- Los perfiles de instalación son una herramienta Drupal nacida para ello
- Instalar el proyecto es tan fácil como instalar el perfil

MAKEFILE

- Documentación rápida de los módulos, temas y librerías usadas
- Identificación rápida de las versiones usadas y los patches aplicados
- Evita tener código duplicado en nuestro repositorio
- Asegura a nuestro cliente que no hemos modificado ningún contrib

FLUJOS DE DESARROLLO EN LAS DISTINTAS FASES DE UN PROYECTO

FASES DE UN PROYECTO

- Desarrollo sin intervención de agentes externos: flujo de instalación
- Desarrollo mientras el cliente interactua con la aplicación: flujo de actualización

FLUJO DE INSTALACIÓN

- Si sólo los desarrolladores estan trabajando en el proyecto, entonces podemos borrar y manipular la base de datos sin problemas
- Para asegurarnos que toda la configuración está en código, debemos instalar nuestra aplicación a menudo
- Desplegar nuestra aplicación en un nuevo entorno es algo tan sencillo cómo:

```
drush site-install <profile>
```

FLUJO DE ACTUALIZACIÓN

- En cuanto el cliente empieza a añadir contenido al sitio, ya no podemos seguir con un flujo de instalación (machacaríamos la base de datos)
- Debemos pasar a un flujo de actualización usando features como hasta ahora pero pasando a usar `hook_update_N()` cuando se requiera
- Para hacer nuevos despliegues, hay que actualizarla, y por tanto hacer:

```
drush updatedb -y
```

```
drush features-revert-all -y
```

EL DESARROLLO

DIVIDE Y VENCERÁS

- Trabajar con metodologías ágiles es un *win*
- Dividimos cada história de usuario en tareas
- Una tarea equivale a un commit (cuanto más unitario, más fácil de revisar)

QUÉ EXPORTAR Y QUE NO

- Debemos ir con cuidado con aquello que exportamos
- Sólo debemos exportar aquellas cosas que son própiamente configuración que no dependa del momento o del entorno (por eso nace State API en Drupal 8)
- Especial cuidado con las variables: debemos ser conscientes de cuando usar `strongarm` o cuando hacer `variable_set` en nuestra función de instalación

QUÉ EXPORTAR

- Campos
- Tipos de contenido
- Taxonomías
- Menús
- Vistas
- Panels (siempre y cuando no puedan ser modificados por el editor)
- Variables (sólo las de configuración)
- ...

QUÉ NO EXPORTAR

- Enlaces de menú
- Roles
- Permisos
- Variables (aquellas que dependan del entorno o sean modificables por el editor)
- Contenido (nodos, términos, etc.)

CUIDADO AL CREAR VISTAS

- En D7, si creamos vistas con algunos filtros estos pasaran a depender de identificadores en vez de nombres máquina
- Filtrar por término: se exporta el `tid`. Filtra mejor por nombre o por machine_name
- Filtro o control de acceso por rol: se exporta el `rid`. Usar el nombre para el filtro o un permiso especial para el control de acceso

ROLES Y PERMISOS

- Si exportamos los permisos y más tarde, el cliente los modifica, pasaremos a tener nuestras features sobreescritas
- Lo mejor es exportar los roles y luego crear los permisos en la función de instalación de nuestro perfil

TRADUCCIONES DE CADENAS

- Debemos usar siempre cadenas en inglés
- La suite `i18n` permite traducir y exportar fácilmente las traducciones
- Tendremos un directorio `/translations` en nuestro perfil de instalación con las traducciones de las cadenas necesarias
- Nuestro proceso de instalación importará todas las traducciones una vez instalados todos los módulos y features

EL TESTING

FACILITAR LAS PRUEBAS

- En un entorno ágil, cómo desarrolladores, debemos facilitar un entorno de pruebas a nuestro PO, que a su vez, deberá poder hacer demos a cliente
- Si en nuestro flujo de desarrollo instalamos en cada despliegue, cómo podemos proveer de contenido de forma fácil para que se puedan hacer las pruebas?
- Para ello nace migrate_default_content

MIGRATE_DEFAULT_CONTENT

- Módulo basado en Migrate para crear contenido de prueba
- API sencilla para crear migraciones de nuevos tipos de contenido, términos, menús, etc.
- Importar y eliminar este contenido por defecto es tan fácil como:

```
drush mi --group=migrate_default_content

drush mr --group=migrate_default_content
```

CONTENIDO NO POR DEFECTO

- Por requisito de algunos proyectos hay contenido que debe proveerse con la aplicación
- Ejemplos de ello pueden ser elementos de menú, términos, alguna página, etc.
- La creación de estos contenidos debe estar programado en nuestra función de instalación del perfil

INTEGRACIÓN CONTÍNUA Y ENTORNOS

INTEGRACIÓN CONTÍNUA

- El equipo debe de trabajar de forma que no sepa que hay más entornos
- Se debe montar un sistema transparente al equipo que haga el trabajo de integración y que sólo informe del resultado

ENTORNOS

Desarrollo

Este entorno se lanza a cada push al respositorio. Integra el código, pasa los tests e informa si todo ha ido bien

Test

Una vez aprobados los commits, y cada ciertas horas, se integra todo el código para que pueda ser probado por el PO

Stage

Este entorno es el accesible por el cliente. Se construye bajo demanda

HERRAMIENTAS

Capistrano

Es el software que usamos para crear las tareas de despliegue basadas en su flow. Desarrollado en Ruby y muy fácil de usar

Jenkins

La usamos para lanzar las builds en los distintios entornos. Hace llamadas a capistrano e informa a nuestras herramientas del resultado

IRC

Sí IRC. Lo usamos para tener un canal todo el equipo y es dónde las builds lanzan los resultados al finalizar cada una de ellas. Muy útil para no perder el foco.

LARGA VIDA A LAS *BUILDS*

- Al lanzar nuestras tareas de construcción en cada entorno, podemos aprovechar para lanzar otras cosas
- Nosotros lanzamos tests, sacamos estadísticas sobre coding standards, revisamos la calidad del código, etc.
- Lo hace una máquina, así que aprovechad para darle trabajo
- Pero no nos pasemos que una *build* larga puede llegar a ser un cuello de botella

OTRAS BUENAS PRÁCTICAS

TRAZABILIDAD

- Desde una tarea debo tener enlazado el commit que la resuelve para futuras revisiones
- Eso es fácil usando algunas herramientas, pero sobretodo, documentando bien
- Por ejemplo, en nuestro mensaje de commit, debemos incluir el identificador de la tarea
- Redmine y jenkins tienen plugins para integrarse entre sí y con nuestro repositorio y facilitar la nevegación entre ellos

REVISIÓN DE CÓDIGO

- Incluir revisión de código es una de las mejores prácticas que hemos añadido en nuestro proceso
- Mejora la calidad del código, el control de errores,...
- Ayuda a mejorar como desarrollador y a aprender mucho de todos
- Facilita la propiedad del código por parte de todo el equipo, no sólo por parte de la persona que lo desarrolla

AGILE

- Agile, agile y agile; no hay más
- Aunque no se pueda usar agile con el cliente, usarlo internamente facilita mucho el trabajo con el equipo de desarrollo
- Definir cada una de las funcionalidades, dividirlas, pero sobretodo discutirlas, hace que se entienda mucho más el proyecto y se puedan prevenir errores

FUTURO

- Crear un entorno por cada commit
- Máquinas virtuales
- Aplicar agile en el proceso de diseño
- ... somos mentes inquietas y nunca paramos de pensar en como mejorar

CONCLUSIONES

- Interpretar esto como un recetario, no como una norma
- Lo mejor es adoptar poco a poco
- Empezar por cambiar la forma de desarrollo, luego empezar con revisión de código y dejar para el final las peleas con la integración contínua y los entornos

DUDAS?