

Watch: **Cómo definir Bounded Contexts**

14:04

Cómo definir Bounded Contexts

La definición de **bounded contexts** es un aspecto crítico que nos permitirá hacer bien la definición de agregados y demás elementos de nuestro sistema. Lo más importante no es centrarse en una definición “perfecta” de los bounded contexts, sino que **puedan promocionarse con facilidad** elementos como los módulos a bounded contexts: Pasito a pasito Driven Development 🦶!

Modelando Bounded Contexts

Tal como veíamos en la lección de introducción, una cuestión fundamental es que mantendremos el **lenguaje ubicuo**, trasladándolo desde el lado de negocio al resto de la aplicación (Si, por ejemplo, nuestro equipo de negocio nos habla en términos de

Mooc, nosotros llevaremos estos términos al código de nuestra aplicación).

Siguiendo el ejemplo de la creación de la plataforma de CodelyTV, vemos múltiples elementos que participan (La Home, el Backoffice, el Mooc, el Blog...), que si tratásemos de tener todo dentro de un mismo proyecto acabaría siendo insostenible, y nos encontraríamos con entidades que tendrían que integrar la lógica de todos los elementos.

Siguiendo DDD, la separación lógica que estableceríamos sería a nivel de bounded contexts (Mooc, Backoffice del Mooc, Blog), dentro de los cuales se estructurarían los módulos

Módulos

Los módulos son elementos que tienen sentido dentro de un bounded context. Se suele seguir la regla de 1 agregado por módulo y llamar al módulo con el nombre de dicho agregado. Por ejemplo...

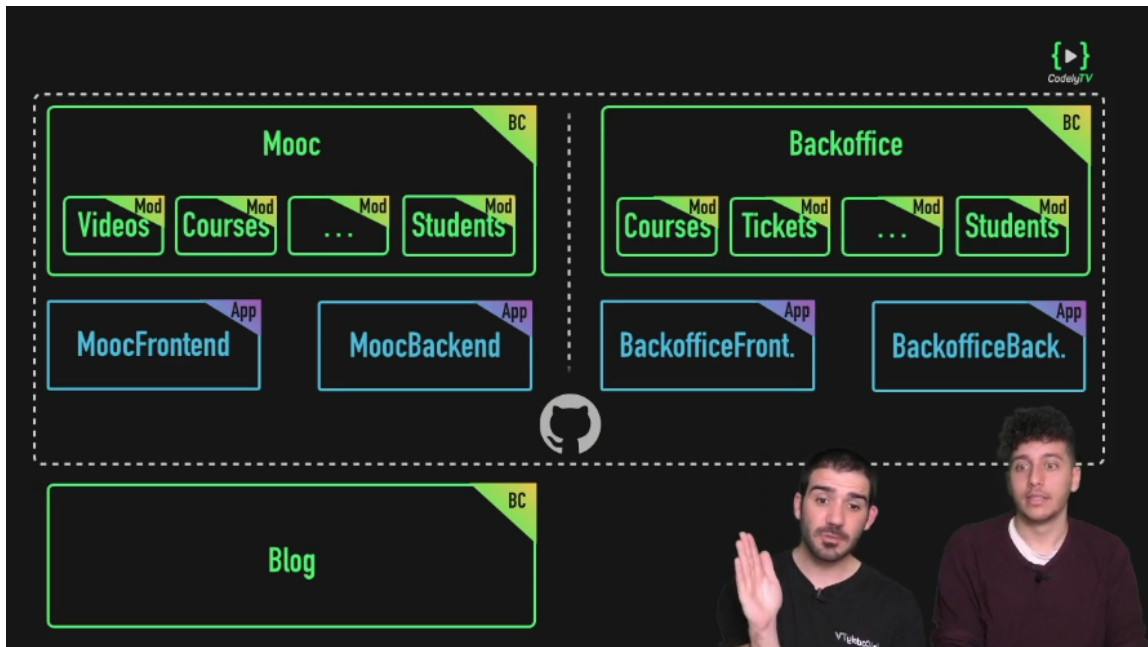
- Mooc: Módulos => Videos, Cursos, Estudiantes, ...
- Backoffice: Módulos => Cursos, Tickets, Estudiantes, ...

Vemos cómo Los módulos no tienen por qué coincidir entre nuestros diferentes bounded contexts (Un módulo Tickets tendrá sentido en el backoffice, pero no en el mooc o el blog)

Aplicaciones

Las aplicaciones vendrían a definir el “cómo y donde queremos desplegar” (Web, dispositivos móviles, etc), por lo que no nos interesa integrarlo dentro de un bounded context

Ojo!👁👁 Si tenemos nuestras aplicaciones en el mismo repo que los bounded contexts, nuestras aplicaciones web podrán comunicarse con estos bounded contexts sin necesidad de salir al exterior del servidor (No hará falta establecer comunicaciones HTTP, TCP, ...) mejorando enormemente la latencia en nuestra aplicación 🏃



Alternativas ↔

Apps dentro de los Bounded Contexts

En este caso, el propio bounded context contiene a la aplicación

- Ventajas: El código está autocontenido y mas recogido
- Inconvenientes: Puede ser engorroso si nuestra aplicación tiene que llamar no solo a la api de su bounded context sino a otras externas (tendríamos una aplicación dentro de un bounded context que tiene que salir para después volver de nuevo)

Módulos dentro de las Apps




En este caso, tendríamos un frontend y backend que contendrían a los módulos.

- Ventajas: El código está agrupado a nivel de conceptos
- Inconvenientes: Estaríamos confundiendo los módulos propiamente dichos con una agrupación de código por conceptos


Repositorios de código del curso

Aquí tienes los repositorios a los que haremos referencia durante el curso. Están en continuo desarrollo y les vamos añadiendo funcionalidades poco a poco.

¡Stars en GitHub y Pull Requests bienvenidas!

-  [PHP](#) (el más completo y con más ejemplos)
-  [Scala](#)
-  [Java](#)

¿Alguna duda?

Nos vemos en el siguiente video  Diferencias entre Bounded Contexts, Subdomains, Modules, y Shared Kernel

Recuerda que puedes abrir una nueva discusión aquí abajo si tienes cualquier duda o te gustaría sugerirnos alguna mejora para esta lección. 