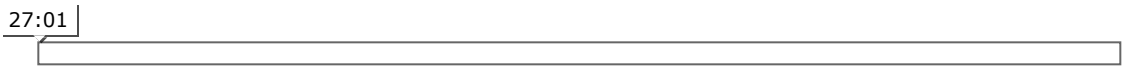
Watch: 💣 **Redefiniendo el testing en frontend**



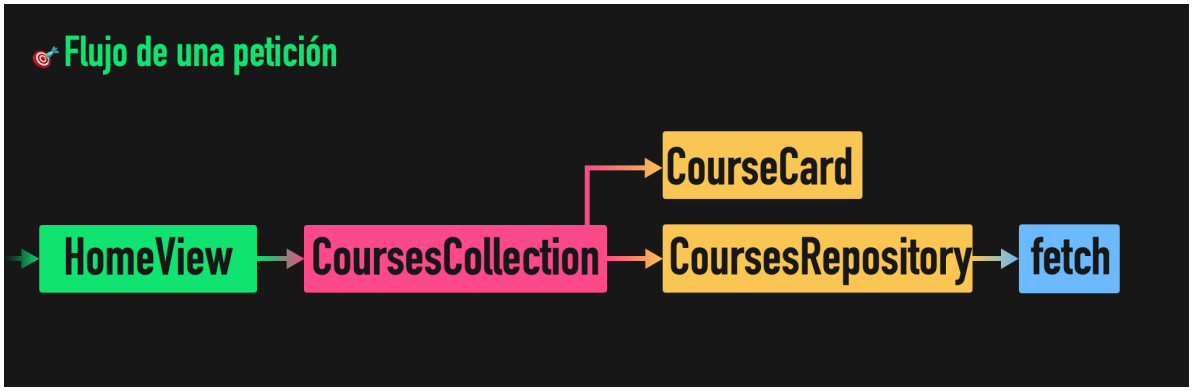
Puedes encontrar el código de este vídeo en GitHub.

Tradicionalmente se plantean las capas de test usando la pirámide:

- En la base tenemos los tests unitarios, que dan menos cobertura pero se ejecutan muy rápido.
- En segundo lugar, tenemos los tests de integración. tienen más cobertura pero pasan más lentamente ya que integramos nuestro código con una parte externa.
- Finalmente, en la parte superior tenemos los tests de aceptación o end to end, que cubren toda la aplicación pero se ejecutan muy lentamente ya que tenemos que levantar todas las partes de nuestra aplicación.

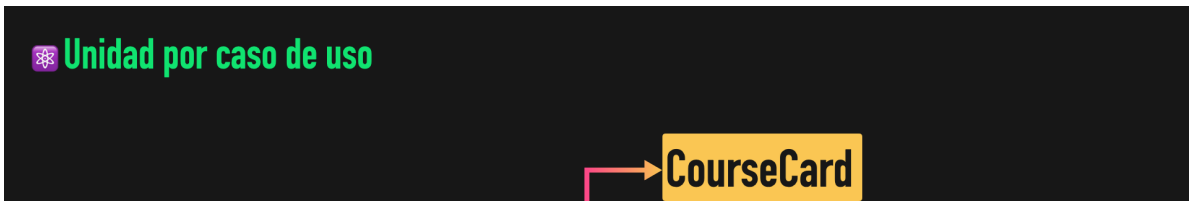
Kent C. Dodds plantea cambiar la pirámide por el trofeo de testing, según el cual la mayor parte de nuestro testing se concentra en integración. Pero, ¿qué es un test unitario y qué es un test de integración?

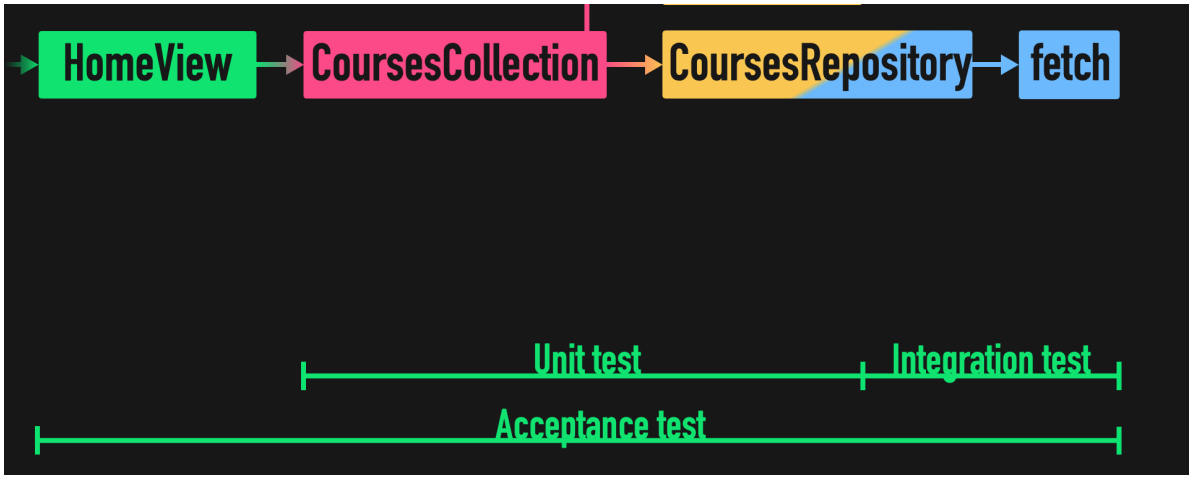
Este es el flujo de nuestra aplicación:



Si testeamos por unidad, es decir, consideramos un test unitario el que cubre solo una de las partes y dobla lo demás, vemos que estos tests nos dan poco valor.

Lo que consideramos como unitario es el caso de uso, es decir, todo lo relacionado con los cursos: CourseCollection, sin doblar sus dependencias: CourseCard y CoursesRepository. Pero como nuestra aplicación necesita conectarse con una API externa, cubrir esta parte implicaría que nuestro test unitario sería muy lento, especialmente teniendo en cuenta que es a este nivel donde testeamos todas las posibilidades de nuestra lógica de negocio, delegando a niveles superiores testear lo que consideraríamos *happy path*. Los tres niveles nos quedarían entonces distribuidos de la siguiente forma:





Así vemos que la diferencia entre la pirámide de tests y el trofeo que plantea Kent C. Dodds es más una diferencia de lo que consideramos test unitario, y al final estamos alineados en que la mayoría de nuestros tests tienen que cubrir un caso de uso.

Links relacionados:

- Artículo de Kent C. Dodds: Write tests. Not too many. Mostly integration.