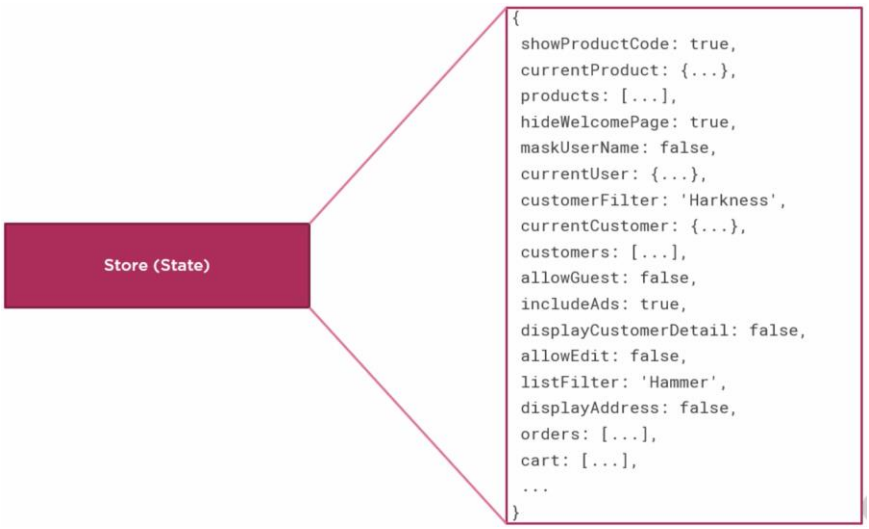
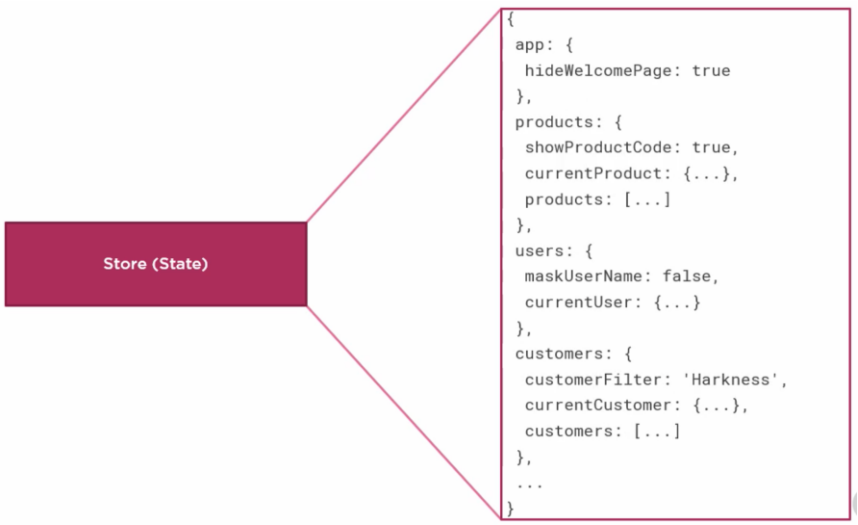
Tipado fuerte del estado – Definir Interfaces para los slice del estado

Antes hemos hablado de cómo organizar y estructurar el estado de nuestro Store. No queremos un gran objeto plano de estado:



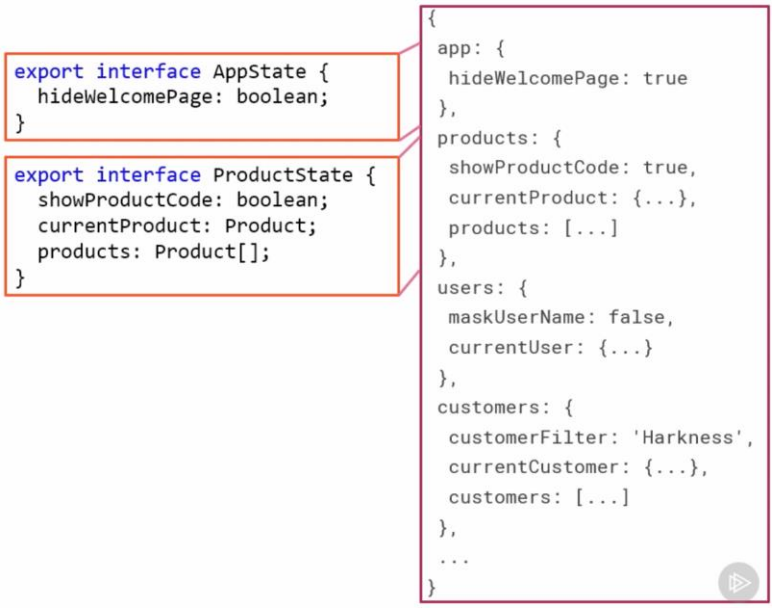
Más bien, queremos tener nuestro estado como pedazos o slices equivalentes a los features de nuestra aplicación:



¿Cómo tipeamos fuertemente este estado? Ya que estamos en TypeScript, usamos interfaces.

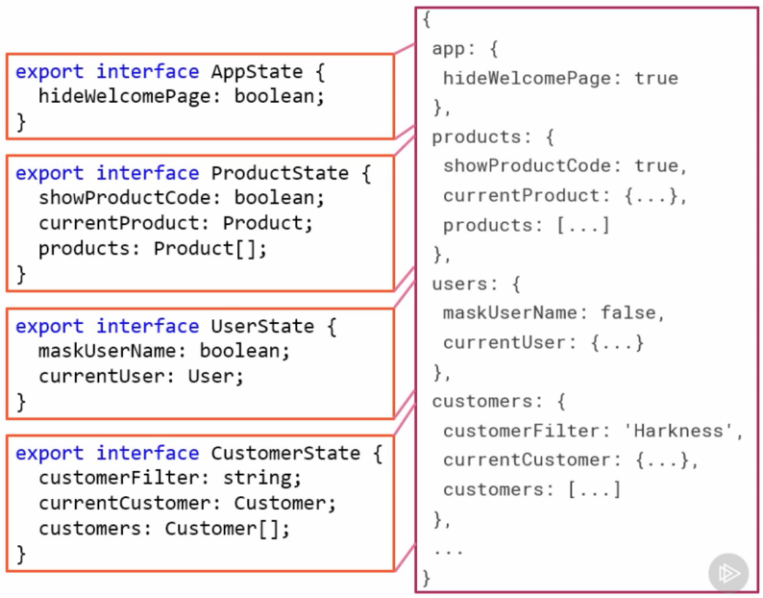
Para cada slice de estado, definimos una interfaz que describe la estructura de ese estado. Para el estado raíz de la aplicación, podemos tener una propiedad *hideWelcomePage* de tipo booleano que define si ocultar la página de bienvenida en futuros inicios de sesión:

Para el estado *products*, tenemos una propiedad *showProductCode*, una *currentProduct* y una lista de *products:*

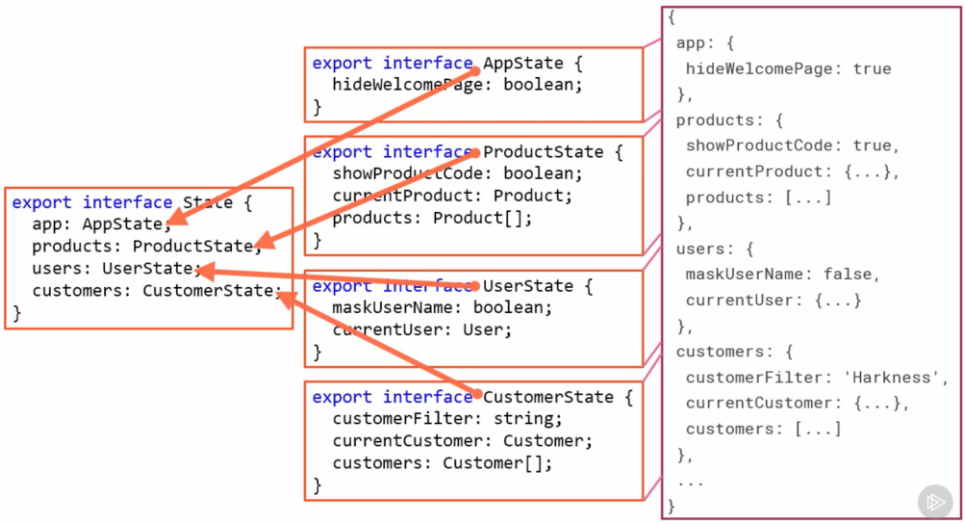


La interfaz coincide con nuestro *slice* de estado *products* y se puede utilizar para tipear fuertemente ese estado.

Creamos algo similar para cada slice adicional de estado:



A continuación, juntamos todas las interfaes para componer el estado global de la aplicación a partir cada slice de estado:



Esta nueva interface State define todo nuestro árbol de estado, haciendo referencia a cada una de las *feature interfaces*.

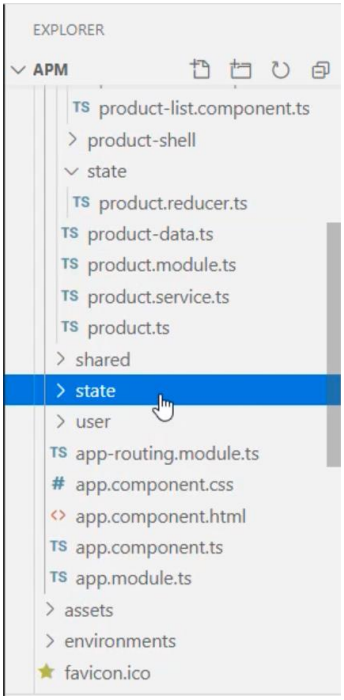
Vamos a implementarlo en nuestra aplicación.

Nuestra primera pregunta es dónde debemos definir la interfaz para nuestro estado. Anteriormente, dijimos que nuestros Reducers proporcionan una representación del estado de nuestra aplicación, por lo que tiene sentido definir nuestras *feature interfaces* de estado en sus Reducers asociados. Así que, en el archivo *product.reducer.ts*, añadiremos una interfaz para el *slice* de estado *products*:

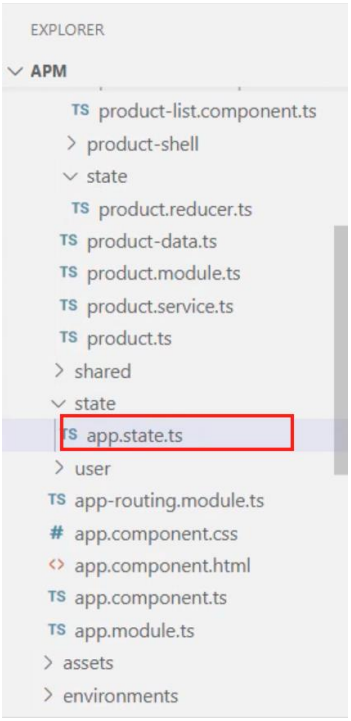
```
TS product.reducer.ts ×
1   import { createReducer, on, createAction } from '@ngrx/store';
2
3   import { Product } from '../product';
4
5   export interface ProductState {
6     showProductCode: boolean;
7     currentProduct: Product;
8     products: Product[];
9   }
10
11  export const productReducer = createReducer(
12    { showProductCode: true },
13    on(createAction('[Product] Toggle Product Code'), state => {
14      return {
15        ...state,
16        showProducCode: !state.showProductCode,
17        myFavoriteMovie: 'LOTR'
18      };
19    })
20  );
21
```

Exportamos la interfaz para poder utilizarla en cualquier parte de nuestra aplicación, y la nombramos con el nombre del feature.

A continuación, definimos la interfaz para todo el árbol de estado de la aplicación. ¿Dónde lo ponemos? Una opción es crear una carpeta directamente debajo de la carpeta app y llamarla *state*:



Luego, en esta carpeta, creamos un archivo llamado *app.state.ts*:



En este archivo, definimos la interfaz para todo el estado de nuestro Store:

Actualmente no tenemos ningún estado para nuestro componente raíz. Tenemos estado para nuestros features de product y user, así que definimos dos propiedades para esos features.

Ahora acabamos de definir la estructura completa del estado de nuestro Store. Actualmente tiene dos *feature slices*, *products* y *user*. A medida que construyamos otros features, las añadiremos aquí.

Pero, hay un problema con este código, nuestro *feature product* está implementando *lazy loadin*g. Vamos a ver lo que eso significa y por qué este código rompe nuestros límites de *lazy loading*.