

# Composables

---

Els composables son fragments de lògica de codi reutilitzable que no tenen un component gràfic. L'objectiu es evitar definir una vegada i una altre el mateix còdi a cada pàgina, amb l'efecte de que al voler fer una millora aquesta s'aplicarà a totes les pàgines editant un únic fitxer.

## Estructura bàsica

- Els composables han d'anar dins la carpeta composables del mòdul en qüestió i en el seu defecte dins shared
- El nom del composable ha de començar per use. Ex: useExpedient, useFitxer...
- El composable ha de tenir un return amb les propietats i mètodes que exposa

Estructura bàsica:

```
export const useExemple = (parametres) => {  
  // definició de variables i mètodes  
  const propietat1 = ...  
  const propietat2 = ...  
  
  const metodel = (params) => {  
    ...  
  }  
  
  return {  
    // parametres i mètodes  
    propietat1,  
    propietat2,  
  
    metodel  
  }  
}
```

Al voler usar un composable simplement l'hem de cridar

```
const { propietat1, propietat2, metodel } = useExemple()
```

## Exemples

Anam a veure uns exemples pràctics (versió simplificada d'ells) que es troben al core de l'esquelet.

### useLoading

L'utilitat d'aquest composable es mostrar un loading quant l'element que hem de mostrar encara no s'ha carregat.

Aquest loading s'ha de mostrar si:

- S'esta fent la petició
- No tenim l'element previament carregat

```
const useLoading = (
  isFetching, // Ens indica si s'esta fent la petició al back
  model // Model reactiu
) =>{

  // Loading serà true si esta fent la petició a l'API i el model no esta
  // emplenat.
  const loading = computed(() => isFetching.value && !model.value)

  // Escoltamos els canvis a la variable loading
  watch(
    loading,
    (newVal) => {
      if(newVal){ // Si canvia a true mostrem el loading
        quasar.loading.show()
      } else { // En cas contrari l'amagam
        quasar.loading.hide()
      }
    },{
      immediate: true
    })

  return {
    loading // Podem retornar la variable loading per si la volem usar
  }
}
```

## useSelectFiltre

Aquest composabile simplifica la creació de filtres per al desplegable que son molt llargs, com el de països o municipis

Partim d'un llistat d'elements totals i volem obtenir el mètode per a filtrar i el llistat filtrat.

Encara que en el composabile del core els paràmetres son opcionals els farem obligatoris en aquest exemple.

```
export const useSelectFilter = ({
  repo, // llistat complet
  field, // Camp corresponent al label del option
  id, // Camp corresponent al value del option
  searchFiled, // Camp per el que volem filtrar
}) => {
```

```
// Carregam les dades a una nova variable per a normalitzar(majuscles,
llevar accents...) el camp de cerca
const fullData = computed(() => {
  repo.value.map(d => {
    return {
      label: d['field'] // Agafam el camp de label
      value: d['id'] // Agafam el camp de valor
      search: normalize(d['searchField']) // Normalitzam el camp a cercar
    }
  })
})

// Inicialitzam les dades filtrades amb tots els elements
const data = ref(fullData)

// La definició d'aquest mètode es correspon a la del component
const filter = (
  val, // Valor de cerca
  update // Funció per actualitzar el select
) => {
  if(val.length < 3){ // Si el text de cerca es curt retornem tots el
    valors
    update(()=> {
      data.value = fullData.value
    })
  } else {
    update(()=> {
      // normalitzam el camp de cerca
      const normalitzat = normalize(val)
      // Filtram els valors
      data.value = fullData.value.filter(
        v => v.search.indexOf(normalitzat) > -1
      )
    })
  }
}

return {
  data,
}
```