Vue.js (Routing)

Interface Pessoa-Máquina - 25/26 - LEI / UM

Hugo Pacheco
hpacheco@di.uminho.pt

Routing

- Uma funcionalidade essencial em backends web
  - Servir um conjunto de páginas web que constituem um site
    - ✚ Organização hierárquica por URL = diretorias num sistema de ficheiros
  - Cada URL é uma página independente
    - ➖ Páginas não partilham estado (ao nível do front-end)
    - ➖ Reload do browser quando se navega entre páginas
  - Várias funcionalidades adicionais
    - ✚ Páginas criadas dinamicamente, e.g. PHP
    - ✚ Controlo de acessos
    - ✚ …

SPAs ⇒ “MPAs”

- Até agora, SPAs
  - A aplicação é convertida numa só página HTML + CSS + JS
  - Um só URL = um só entry point
  - Separação de conceitos / componentes apenas interna à framework, não visível para o utilizador
- Vue permite construir “MPAs”
  - Preservam as características de SPAs
  - Organizar conteúdo da aplicação por URLs acessíveis externamente
  - Algumas vantagens na gestão de estado da página

Vue Router

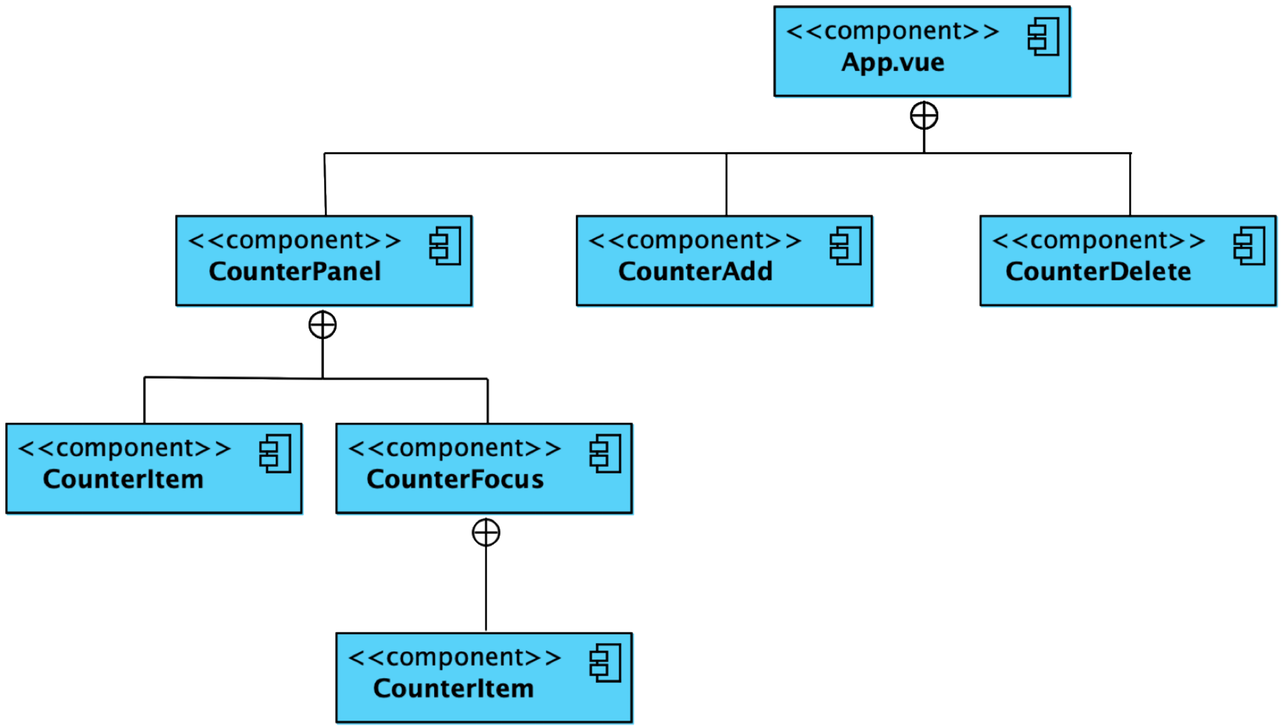
- Biblioteca oficial para routing em Vue

```
npm install vue-router
```

- Permite navegação por URL dentro de uma SPA
- Routing simulado, gera na mesma apenas uma página HTML + CSS + JS
  - Compatível com browser navigation ⇒ experiência para o utilizador como se fosse uma MPA real
- Navegação mais imersiva dentro da aplicação:
  - Programmatic Navigation
  - Scroll Behaviour Control
  - Lazy loading
  - …

Exemplo (Counters)

- Antes de routing, um exemplo de components
  - Muita comunicação entre components para preservar estado com navegação entre tabs



Setup

- No ficheiro `main.js`
  - Acrescentar uma nova dependência

```
import { createRouter, createWebHistory } from ‘vue-router’;
```

- Definir o router

```
const router = createRouter({
    history: createWebHistory(),
    routes: […]
});
```

Utilizar suporte nativo do browser para histórico de navegação

- Ao montar a aplicação, acrescentar o router

```
app.use(router);
```

Exemplo (Counters + Tab Routes)

- No ficheiro `main.js`, criar routes para os 3 tabs

```
{ path: '/panel', component: CounterPanel }
```

  - Uma route tem um `path` (sufixo do URL dentro da página) e aponta pra um `component`

- No ficheiro `App.vue`

  - Substituir `button` de cada tab por `router-link`

```
<router-link to=“/panel">Painel</router-link>
```

    - Custom link element para routers, navegação não recarrega página

    - Como anchor tag HTML `a`, aponta para um `path` em vez de para um URL

Exemplo (Counters + Tab Routes)

- No ficheiro `App.vue`
  - Substituir if-then-else de components no template HTML por apenas um `router-view`
    - Como template slots para routes
  - Navegar programaticamente para “Painel” quando se apagam contadores no “Apagar”

```
import { useRouter } from
'vue-router';
const router = useRouter();
router.push('/panel');
```

- No HTML: `router-link` é compilado para HTML link `a`
- No CSS

`button` ⇒ `a`
`.active` ⇒ `.router-link-active`

Route properties

- Dentro de um component, podemos aceder às propriedades da route atual

```
import { useRoute } from 'vue-router';
const route = useRoute();
```

- Algumas propriedades:

```
route = { path : “/users/:id”, … }

route.fullPath                “/users/42?tab=posts#details”
route.path                    “/users/42”
route.params                  {id : “42” }
route.name                    shortname dado em Vue
route.query                   {tab : “posts” }
route.hash                    “#details”
```

Dynamic routes

- Podemos definir uma route com parâmetros (`:var`)

```
{ path: '/panel/:counterID', component: CounterFocus }
```

- Podemos passar parâmetros como props ao component

```
props: true
```

- Podemos definir uma função que transforma parâmetros em `props`

```
props: (route) =>
({ contador: { id: route.params.counterID, help: 'no help',
total: 0 } })
```

? Carregar um `CounterFocus` sem estar dentro de um `CounterPanel`?

! Estamos a criar um contador “falso”

Exemplo (Counters + Nested Routes)

- A solução mais idiomática para suportar `path` para `CounterFocus` é utilizar nested routes

```
{ path: '/panel'
, component: CounterPanel
, children: [
  { path: ‘:counterID', component: CounterFocus, props: true, }
]}
```

- Nested router view
  - No `CounterPanel`: CounterFocus ⇒ `router-view`
- “Focar” um contador redireciona para `/panel/:counterID`
- “Fechar” um contador redireciona para `/panel`

Exemplo (Counters + Nested Routes)

- ❗ Mas um `CounterFocus` recebe uma prop `contador` com valor `{ id : …, help : …, total : … }`, e o router só lhe fornece o `counterID`
- 💡 Queremos que o `CounterPanel` defina a prop `contador` do `CounterFocus`
  - A partir do seu estado e da route do componente

"Pattern matching" sobre a route

Template element para components

```
<router-view v-slot="{ Component, route }">
  <component
    :is="Component"
    :contador="makeContador(route.params.counterID)"
  />
</router-view>
```

Exemplo (Counters + Multiple Components)

- Podemos utilizar query parameters
  - No URL: `/panel/Remates?help=text`
  - No component `CounterFocus`: `route.query.help`

- Templates podem ter múltiplas `slot` / `router-view`
  - E.g., acrescentar um novo `CounterReset` ao `CounterPanel`

```
components: { default: CounterPanel, reset: CounterReset }

<main>
  <router-view></router-view>
  <router-view name="reset"></router-view>
</main>
```

named components, with default

default slot

named slot

Exemplo (Counters + NotFound)

- Podemos utilizar navigation guards para alterar routes
  - Antes de resolver qualquer route

    ```
    router.beforeEach((to,from,next) => { … })
    ```

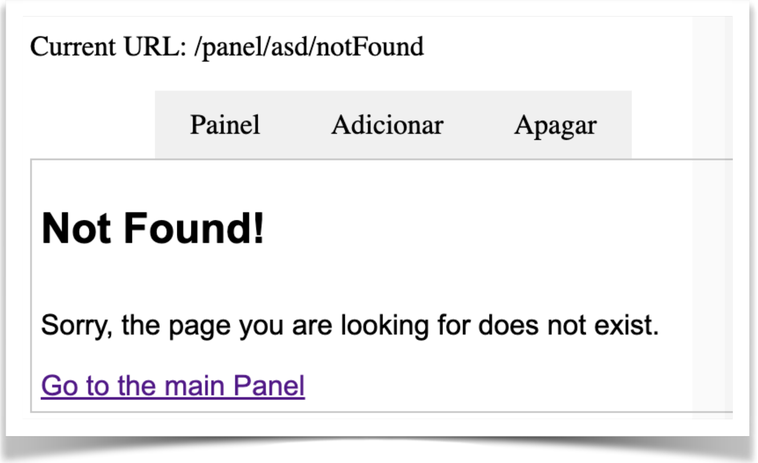
  - Depois de resolver qualquer route

    ```
    router.afterEach((to,from,failure) => { … })
    ```

  - Antes de resolver uma route em partilhar, declarado nas suas propriedades

    ```
    beforeEnter : (to,from,next) => { … }
    ```

- Podemos utilizar expressões regulares em paths
  - E.g., catch all routes wildcard, como ’ `/:notFound(.*)` ’ no fim da route list



Exemplo (Counters + NotFound)

- ! Routes são declaradas no ficheiro `main.js` antes dos components
- ? E se a função que executa aquando de uma navigation guard precisar de ler o estado do modelo, definido num `Component.vue`?

```
beforeEnter : (to,from,next) => { … }
```

  - ! Por enquanto vamos utilizar um **store pattern** ⇒ hack!

    1. Declarar `ref` necessária para as routes num ficheiro JS à parte
    2. Importar no `main.js` (para as routes)
    3. Importar no `Component.vue` (onde estava declarado)