01-list

En este módulo trabajaremos con una aplicación más completa, vamos a simular un portal inmobiliario. En este ejemplo vamos a implementar un listado de propiedades que recuperaremos de servidor.

Pasos a realizar

Comenzaremos por la página principal `property-list.html`. Ésta es la página del listado de propiedades, donde vamos a simular llamadas a servidor y recuperar dicha lista.

- Vamos a crear el fichero principal `property-list.js`:

./src/pages/property-list/property-list.js

```
console.log('property-list page');
```

- Y referenciarlo en el html:

./src/pages/property-list/property-list.html

```
...
    <footer>
      <img class="footer_logo" src="../../core/content/img/logo_header.svg" />
    </footer>
+   <script src="property-list.js"></script>
  </body>
</html>
```

- Vamos a recuperar los datos de servidor:

./src/pages/property-list/property-list.api.js

```
import Axios from 'axios';

const url = `${process.env.BASE_API_URL}/properties`;

export const getPropertyList = () => Axios.get(url).then(({ data }) => data);
```

- Este método lo vamos a utilizar en cuanto se carga el fichero `property-list`, pero primero pensemos en qué necesitamos pintar:

```
interface Property {
  id: string;
  title: string;
  rooms: string; // 3 habitaciones
  squareMeter: string; // 136m2
  notes: string; // Truncate 240 chars
  price: string; // 120.000 €
  image: string; // image base64
}
```

- Por tanto, vamos a necesitar un `mapper` para transformar la entidad del servidor a la entidad de cliente:

./src/pages/property-list/property-list.mappers.js

```
export const mapPropertyListApiToVm = propertyList =>
  Array.isArray(propertyList)
    ? propertyList.map(property => mapPropertyApiToVm(property))
    : [];

const mapPropertyApiToVm = property => ({
  id: property.id,
  title: property.title,
  rooms: `${property.rooms} ${getRoomWord(property.rooms)}`,
  squareMeter: `${property.squareMeter}m2`,
  notes: `${property.notes.substring(0, 240)}...`,
  price: `${property.price.toLocaleString()} €`,
  image: Array.isArray(property.images) ? property.images[0] : '',
});

const getRoomWord = rooms => (rooms > 1 ? 'habitaciones' : 'habitación');
```

- Vamos a utilizar amos métodos para poder pintar la lista:

./src/pages/property-list/property-list.js

```
- console.log('property-list page');
+ import { getPropertyList } from './property-list.api';
+ import { mapPropertyListApiToVm } from './property-list.mappers';
+ import { addPropertyRows } from './property-list.helpers';

+ getPropertyList().then(propertyList => {
+   const vmPropertyList = mapPropertyListApiToVm(propertyList);
+   addPropertyRows(vmPropertyList);
+ });
```

- Continuamos ahora con el filtro, ya que se tratan de elementos `select` de HTML, necesitamos cargar las opciones disponibles de cada uno. Aquí podríamos distinguir entre dos tipos de opciones:
    - Las definidas servidor
    - Las definidas en cliente
- Vamos a cargar las de servidor:

./src/pages/property-list/property-list.api.js

```
import Axios from 'axios';

- const url = `${process.env.BASE_API_URL}/properties`;
+ const propertyListUrl = `${process.env.BASE_API_URL}/properties`;

- export const getPropertyList = () => Axios.get(url).then(({ data }) => data);
+ export const getPropertyList = () => Axios.get(propertyListUrl).then(({ data })
=> data);

+ const saleTypeListUrl = `${process.env.BASE_API_URL}/saleTypes`;

+ export const getSaleTypeList = () =>
+   Axios.get(saleTypeListUrl).then(({ data }) => data);

+ const provinceListUrl = `${process.env.BASE_API_URL}/provinces`;

+ export const getProvinceList = () =>
+   Axios.get(provinceListUrl).then(({ data }) => data);
```

- Utilizamos los dos métodos, esperar a que todas las promesas se resuelvan:

./src/pages/property-list/property-list.js

```
import {
  getPropertyList,
+ getSaleTypeList,
+ getProvinceList,
} from './property-list.api';
import { mapPropertyListApiToVm } from './property-list.mappers';
- import { addPropertyRows } from './property-list.helpers';
+ import { addPropertyRows, setOptions } from './property-list.helpers';

- getPropertyList().then(propertyList => {
+ Promise.all([
+   getPropertyList(),
+   getSaleTypeList(),
+   getProvinceList(),
+ ]).then(([propertyList, saleTypeList, provinceList]) => {
+   loadPropertyList(propertyList);
```

```
+   setOptions(saleTypeList, 'select-sale-type', '¿Qué venta?');
+   setOptions(provinceList, 'select-province', '¿Dónde?');
+ });

+ const loadPropertyList = propertyList => {
    const vmPropertyList = mapPropertyListApiToVm(propertyList);
    addPropertyRows(vmPropertyList);
- });
+ };
```

- Las demás colecciones tienen más sentido tenerlas en cliente, ya que son simplemente valores lo que se necesitan. Las anteriores son entidades que vamos a referenciar en la entidad del inmueble:

> NOTA: Recordad que las opciones la hemos modelado con id y name.

./src/pages/property-list/property-list.constants.js

```
export const roomOptions = [
  {
    id: '1',
    name: '+1',
  },
  {
    id: '2',
    name: '+2',
  },
  {
    id: '3',
    name: '+3',
  },
  {
    id: '4',
    name: '+4',
  },
  {
    id: '5',
    name: '+5',
  },
];

export const bathroomOptions = [
  {
    id: '1',
    name: '+1',
  },
  {
    id: '2',
    name: '+2',
  },
  {
    id: '3',
```

```
    name: '+3',
  },
  {
    id: '4',
    name: '+4',
  },
  {
    id: '5',
    name: '+5',
  },
];

export const minPriceOptions = [
  {
    id: '300',
    name: '300 €',
  },
  {
    id: '600',
    name: '600 €',
  },
  {
    id: '900',
    name: '900 €',
  },
  {
    id: '1200',
    name: '1.200 €',
  },
  {
    id: '1600',
    name: '1.600 €',
  },
  {
    id: '2000',
    name: '2.000 €',
  },
  {
    id: '2500',
    name: '2.500 €',
  },
];

export const maxPriceOptions = [
  {
    id: '100000',
    name: '100.000 €',
  },
  {
    id: '120000',
    name: '120.000 €',
  },
  {
    id: '140000',
```

```
    name: '140.000 €',
  },
  {
    id: '180000',
    name: '180.000 €',
  },
  {
    id: '200000',
    name: '200.000 €',
  },
  {
    id: '300000',
    name: '300.000 €',
  },
  {
    id: '500000',
    name: '500.000 €',
  },
];
```

- Lo utilizamos:

./src/pages/property-list/property-list.js

```
...
+ import {
+   roomOptions,
+   bathroomOptions,
+   minPriceOptions,
+   maxPriceOptions,
+ } from './property-list.constants';

Promise.all([getPropertyList(), getSaleTypeList(), getProvinceList()]).then(
  ([propertyList, saleTypeList, provinceList]) => {
    loadPropertyList(propertyList);
    setOptions(saleTypeList, 'select-sale-type', '¿Qué venta?');
    setOptions(provinceList, 'select-province', '¿Dónde?');
+   setOptions(roomOptions, 'select-room', '¿Habitaciones?');
+   setOptions(bathroomOptions, 'select-bathroom', '¿Cuartos de baño?');
+   setOptions(minPriceOptions, 'select-min-price', 'Min (EUR)');
+   setOptions(maxPriceOptions, 'select-max-price', 'Max (EUR)');
  }
);
...
```

- Ya que tenemos cargados los maestros, vamos a recoger el valor seleccionado por el usuario:

./src/pages/property-list/property-list.js

```
...
+ import { onUpdateField, onSubmitForm } from '../../common/helpers';

...

const loadPropertyList = propertyList => {
  const vmPropertyList = mapPropertyListApiToVm(propertyList);
  addPropertyRows(vmPropertyList);
};

+ let filter = {
+   saleTypeId: '',
+   provinceId: '',
+   minRooms: '',
+   minBathrooms: '',
+   minPrice: '',
+   maxPrice: '',
+ };

+ onUpdateField('select-sale-type', event => {
+   const value = event.target.value;
+   filter = {
+     ...filter,
+     saleTypeId: value,
+   };
+ });

+ onUpdateField('select-province', event => {
+   const value = event.target.value;
+   filter = {
+     ...filter,
+     provinceId: value,
+   };
+ });

+ onUpdateField('select-room', event => {
+   const value = event.target.value;
+   filter = {
+     ...filter,
+     minRooms: value,
+   };
+ });

+ onUpdateField('select-bathroom', event => {
+   const value = event.target.value;
+   filter = {
+     ...filter,
+     minBathrooms: value,
+   };
+ });

+ onUpdateField('select-min-price', event => {
+   const value = event.target.value;
```

```
+   filter = {
+     ...filter,
+     minPrice: value,
+   };
+ });

+ onUpdateField('select-max-price', event => {
+   const value = event.target.value;
+   filter = {
+     ...filter,
+     maxPrice: value,
+   };
+ });

+ onSubmitForm('search-button', () => {
+   console.log({ filter });
+ });
```

- Para aplicar el filtro, en este caso aplica usar un filtro de servidor (tambien se puede filtrar en cliente). En nuestro caso, estamos usando la libreía `json-server`, para tener un servidor de mock. El filter de esta librería funciona pasándole los parametros del filtro por query params. Vamos a crearnos un método para mapear nuestro modelo a ese filtro. En el caso de implementar un servidor real, puede que sea diferente y por tanto tendríamos que actualizar el siguiente método:

./src/pages/property-list/property-list.mappers.js

```
...
const getRoomWord = rooms => (rooms > 1 ? 'habitaciones' : 'habitación');

+ export const mapFilterToQueryParams = filter => {
+   let queryParams = '';

+   if (filter.saleTypeId) {
+     queryParams = `${queryParams}saleTypeIds_like=${filter.saleTypeId}&`;
+   }

+   if (filter.provinceId) {
+     queryParams = `${queryParams}provinceId=${filter.provinceId}&`;
+   }

+   if (filter.minRooms) {
+     queryParams = `${queryParams}rooms_gte=${filter.minRooms}&`;
+   }

+   if (filter.minBathrooms) {
+     queryParams = `${queryParams}bathrooms_gte=${filter.minBathrooms}&`;
+   }

+   if (filter.minPrice) {
+     queryParams = `${queryParams}price_gte=${filter.minPrice}&`;
+   }
```

```
+   if (filter.maxPrice) {
+     queryParams = `${queryParams}price_lte=${filter.maxPrice}&`;
+   }

+   return queryParams.slice(0, -1);
+ };
```

- Vamos a utilizar el mapper:

./src/pages/property-list/property-list.js

```
...
import {
  mapPropertyListApiToVm,
+ mapFilterToQueryParams,
} from './property-list.mappers';
import {
  addPropertyRows,
  setOptions,
+ clearPropertyRows,
} from './property-list.helpers';
...

onSubmitForm('search-button', () => {
- console.log({ filter });
+ const queryParams = mapFilterToQueryParams(filter);
+ clearPropertyRows();

+ getPropertyList(queryParams).then(propertyList => {
+   loadPropertyList(propertyList);
+ });
});
```

- Por último actualizamos la `api`:

./src/pages/property-list/property-list.api.js

```
import Axios from 'axios';

- const propertyListUrl = `${process.env.BASE_API_URL}/properties`;
+ const propertyListUrl = `${process.env.BASE_API_URL}/properties?`;

- export const getPropertyList = () =>
-   Axios.get(propertyListUrl).then(({ data }) => data);
+ export const getPropertyList = queryParams =>
+   Axios.get(`${propertyListUrl}${queryParams}`).then(({ data }) => data);
```

```
...
```