02-account-list

Vamos a partir del ejemplo anterior `01-login` e implementar la página de listado de cuentas.

Pasos a realizar

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

./src/pages/account-list/account-list.js

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

- Y referenciarlo en el html:

./src/pages/account-list/account-list.html

```
...
    <footer id="footer">
      <div class="container">
        <img class="footer_logo"
src="../../core/content/img/logo_footer.svg" />
      </div>
    </footer>
+   <script src="account-list.js"></script>
  </body>
</html>
```

- Vamos a recuperar los datos de servidor:

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

```
import Axios from 'axios';

const url = `${process.env.BASE_API_URL}/account-list`;

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

- Lo utilizamos en cuanto se carga el fichero `account-list`:

./src/pages/account-list/account-list.js

```
- console.log('account-list page');
+ import { getAccountList } from './account-list.api';
+ import { addAccountRows } from './account-list.helpers';

+ getAccountList().then(accountList => {
+   addAccountRows(accountList);
+ });
```

> NOTA: en el fichero `account-list.helpers` tenemos los métodos necesarios para crear de forma dinámica todas las filas de la tabla después de que recuperemos los datos de servidor. Es decir, creamos los componentes HTML desde Javascript.

- Vemos que ya se pintan datos, pero necesitamos unas transformaciones intermedias para poder pintar los datos de una manera determinada en la vista. Por tanto necesitamos un `mapper`:

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

```
export const mapAccountListApiToVm = accountList =>
  Array.isArray(accountList)
    ? accountList.map(account => mapAccountApiToVm(account))
    : [];

const mapAccountApiToVm = account => ({
  id: account.id,
  iban: account.iban,
  name: account.name,
  balance: `${account.balance} €`,
  lastTransaction: new Date(account.lastTransaction).toLocaleDateString(),
});
```

> Si queremos transformaciones de fechas más específicas tendriamos que meternos con date-fns, moment, etc

- Utilizamos el `mapper`:

./src/pages/account-list/account-list.js

```
import { getAccountList } from './account-list.api';
import { addAccountRows } from './account-list.helpers';
+ import { mapAccountListApiToVm } from './account-list.mappers';

getAccountList().then(accountList => {
- addAccountRows(accountList);
+ const vmAccountList = mapAccountListApiToVm(accountList);
+ addAccountRows(vmAccountList);
```

```
});
```

- Ya que tenemos la navegación a cada una de las cuentas, vamos a navegar hacia `movimientos` o `transferencias` cuando seleccionemos una opcion:

./src/pages/account-list/account-list.js

```
import { getAccountList } from './account-list.api';
import { addAccountRows } from './account-list.helpers';
import { mapAccountListApiToVm } from './account-list.mappers';
+ import { onUpdateField } from '../../common/helpers';
+ import { history } from '../../core/router';

+ const setEvents = accountList => {
+   accountList.forEach(account => {
+     onUpdateField(`select-${account.id}`, event => {
+       const route = event.target.value;
+       history.push(route);
+     });
+   });
+ };

getAccountList().then(accountList => {
  const vmAccountList = mapAccountListApiToVm(accountList);
  addAccountRows(vmAccountList);
+ setEvents(vmAccountList);
});
```

> NOTA: En el fichero `account-list.helpers.js` estamos asignado el id de los componentes select.