Juin 2022 -- Devise Authentication

Ce *how to* ne serait pas nécessaire s'il s'agissait d'un cas basique d'authentification. La page github de `devise` suffirait. Cependant, j'ai besoin pour mes apps, non seulement d'identifier un compte mais aussi de l'associer à la structure (entreprise) qui a souscrit à mon produit. Il est alors nécessaire de surcharger certaines fonctionnalités de `devise` .

L'installation

```
bundle add devise
rails g devise:install
```

Ensuite suivre les instructions données à la fin de la procédure d'installation.

Modèle associé

Ensuite, il convient de créer le modèle associé à `devise : account` .

> Après réflexion, je pourrais me laisser aller à utiliser `user` (comme tout le monde), en réservant `account` pour l'entité qui souscrit (rental place). En effet, il n'est pas illogique de dire qu'il y a un compte (account) rental place avec plusieurs usagers (users). Autrement dit, account serait une « entrée » de la base de données partagée par plusieurs users.

```
rails g devise account
```

Éditer la migration associée et décommenter les lignes adéquates de façon à personnaliser le modéle. En paralllèle, éditer le fichier `config/initializers/devise.rb` .

Il est également possible d'ajouter tous les champs dont on a besoin : `username` , `pseudo` ainsi que d'éventielles *foreign_keys*. J'associe chaque `account` à une `rental_place` (entité qui souscrit à mon app). J'ai donc ajouté dans la migration qui crée `account` une ligne :

```
t.references :rental_place, foreign_key: { on_delete: :cascade }
```

Bien entendu, à la migration, le modéle `rental_place` doit être créé avant.

```
rails db:migrate
```

La première ligne de `routes.rb` doit référencer les routes associées à l'authentification :

```
devise_for :accounts
```

Enfin, compléter les modèles :

```
# models/account.rb
class Account < ApplicationRecord
  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable, :trackable and :omniauthable
  devise :database_authenticatable, :registrable, :recoverable,
         :rememberable, :validatable, :lockable, :timeoutable,
         #:confirmable
  belongs_to :rental_place, inverse_of: :accounts
end

# models/rental_place.rb
class RentalPlace < ApplicationRecord
  has_many :accounts, dependent: :destroy, inverse_of: :rental_place
  (...)
end
```

Customization

Je décris ici comment (après de longues heures de recherche) je suis parvenu à mettre en place l'authentification (*sign_in* dans le langage `devise`) et la souscription (*sign_up*) avec création de l'entité souscrivant associée.

L'idée : lorsque qu'un internaute souscrit à mon app, il définit différentes caractéristiques propres à l'entité pour laquelle il souscrit. Il devient ainsi l'administrateur de cette souscription. Il pourra, à l'intérieur de l'app, créer d'autres comptes associés avec des *droits* (éventuellement) différents de *admin*; par exemple *manager* ou encore *employee*.

Il va falloir :

- surcharger[1] le controller `registration` propre à devise.

```
rails g devise:controllers accounts -c registrations
```

```
# accounts/registrations_controller.rb
class Accounts::RegistrationsController < Devise::RegistrationsController
  before_action :configure_sign_up_params, only: [:create]

  (...)

  # override_build_resource method (in use in new and create methods)
  def build_resource( params = nil )
    super( params )
    resource.build_rental_place( RentalPlace.defaults )
  end

  (...)
  # If you have extra params to permit, append them to the sanitizer.
  def configure_sign_up_params
    params[:account].merge!( position: 'admin' )

    devise_parameter_sanitizer.permit(
      :sign_up,
      keys: [
        :email,
        :password,
        :password_confirmation,
        :position,
        :rental_place_id
      ]
    )
  end

  (...)
end
```

- compléter la définition de la route associée à `devise` pour qu'elle pointe vers le nouveau controller.

```
devise_for :accounts,
:controllers => {
  :registrations => "accounts/registrations"
}
```

- enfin, générer les vues de façon à les personnaliser (je choisis de les mettre dans `views/devise` qui est un fall-back plutôt que de les partager entre `account` (registration) et `devise`).

```
rails g devise:views
```

  Un layout (dans `views/layout` nommé `devise.html.erb` peut être défini pour contenir par défaut toutes les vues devise...
- si on veut ajouter des méthodes à un controller `devise`, il suffit de définir les routes supplémentaires (attention, je n'ai pas encore testé) :

- first way :

```
devise_scope :accounts do
  get '..', to: 'sessions#..'
end
```

- second way :

```
devise_scope :sessions do
  get '..', to: 'accounts/sessions#..
end
```

---

[1] : *et décommenter ce qui est nécessaire.*