TP – Ruby on Rails

FERTILLE Eliott

31/12/2023

—

Développer des composants d'accès aux données

—

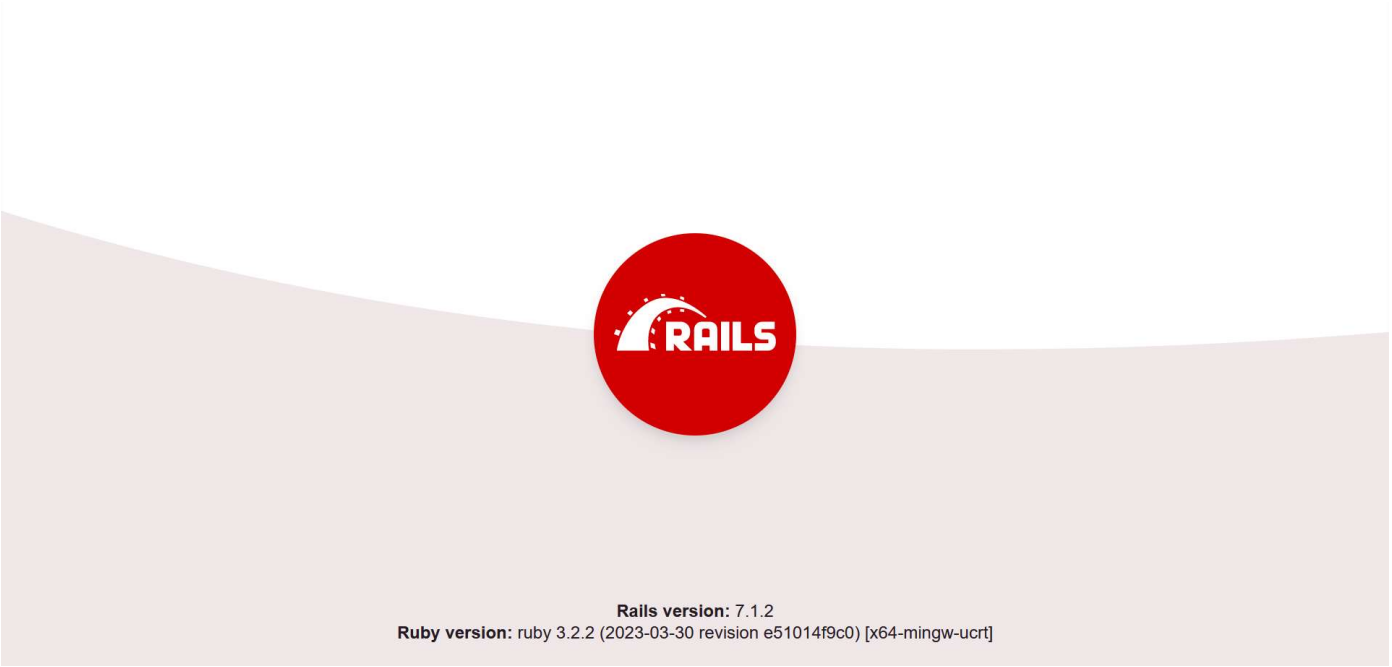
Mr. BENTIFRAOUINE

Création de l'application Ruby

J'ai créé mon app Ruby en lançant les commandes suivantes :

- rails new LeCarnetDeVoyage
- cd LeCarnetDeVoyage
- bundle install
- gem update
- bundle
- rails server

De ce fait, mon projet se nomme 'LeCarnetDeVoyage' et toutes les dépendances liées au Framework Ruby sont bien installées et à jour.
La commande 'rails server' me permet de lancer mon projet une première fois pour voir si tout va bien :



Configuration de la base

Maintenant pour créer mes premiers models User et Trip, j'utilise un outil de Ruby : Scaffold

Grâce à ça, je peux créer mes models simplement avec les commandes suivantes :

- rails g scaffold User nom:string email:string
- rails g scaffold Trip destination:string description:text dateDebut:date dateFin:date

Après avoir créé mes models, je migre ma base avec la commande 'rails db:migrate' pour mettre à jour le schéma de la base.

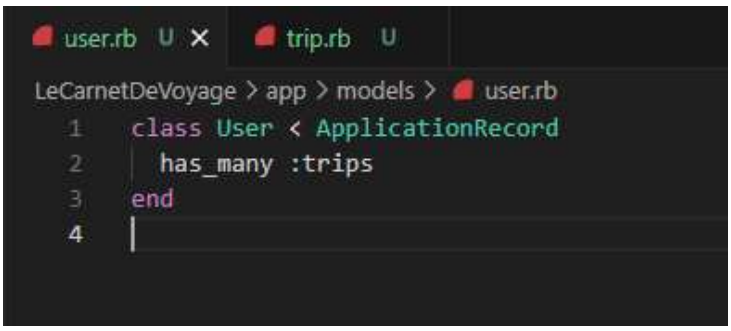
Sans Scaffold, j'aurais tout simplement créé mon model User et Trip comme suit :
- rails g model User nom:string email:string password:string
- rails g model Trip destination:string description:text dateDebut:date dateFin:date

Puis ensuite j'aurais ajouté toutes les opérations CRUD dans les controller créé au préalable avec les commandes :

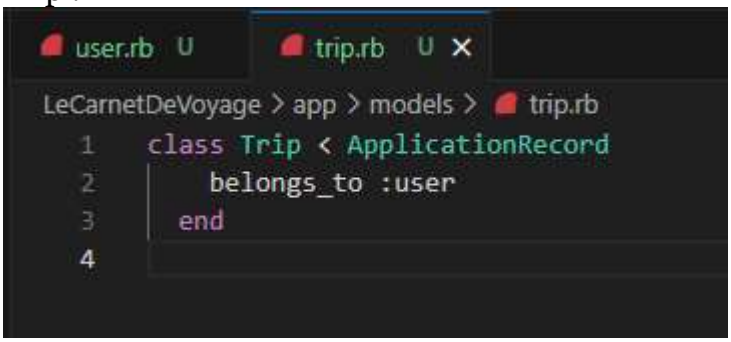
- rails g controller User
- rails g controller Trip

Pour établir la relation one-to-many, j'ai tout simplement ajouté les relations dans les models User et Trip.

User :

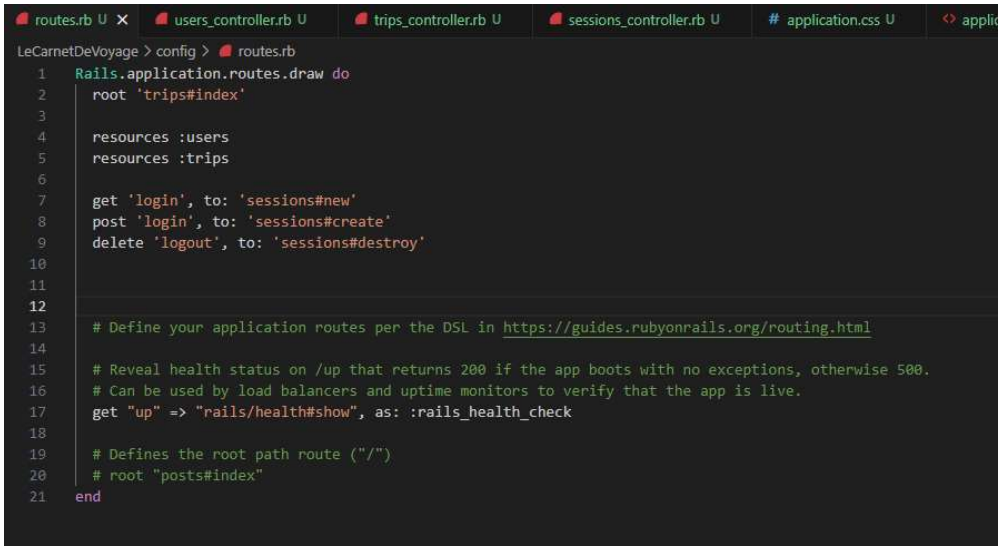


Trip :



Manipulation des données

Pour ce qui est des routes, j'ai décidé de mettre la page index des trips en route par défaut pour que le projet se lance directement sur les voyages que comporte la base et donner la possibilité d'en ajouter en se connectant.



Les deux models User et Trip sont bien gérés par des routes REST ce qui facilite les opérations CRUD.

routes.rb U | users_controller.rb U | trips_controller.rb U | sessions_controller.rb U | application.css U | application.html.erb U

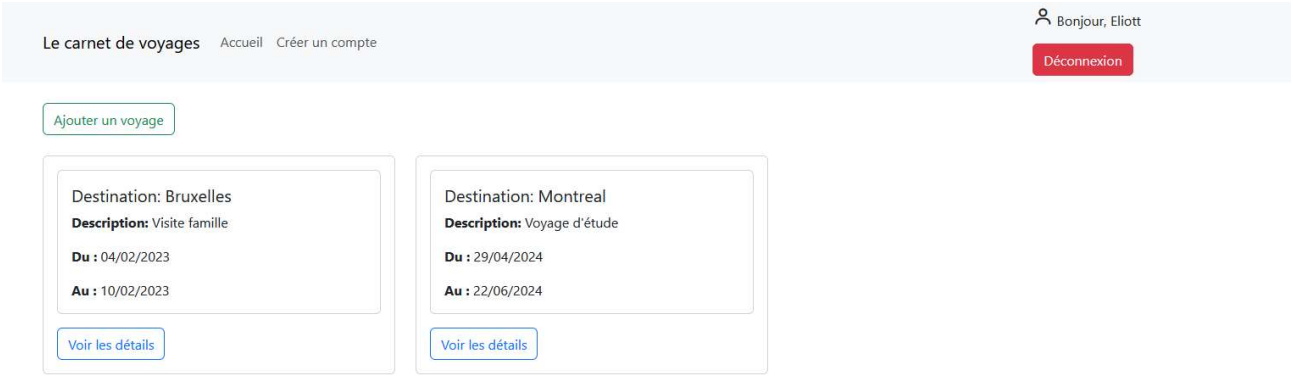
LeCarnetDeVoyage > app > controllers > trips_controller.rb

```ruby
class TripsController < ApplicationController
  # Pour effectuer toutes actions CRUD sur un Trip il faut être connecté
  before_action :logged_in_user, only: [:new, :create, :edit, :update]

  # GET /trips or /trips.json
  # Affiche tous les voyages
  def index
    @trips = Trip.all
  end

  # GET /trips/1 or /trips/1.json
  # Affiche un voyage en fonction de son id
  def show
    @trip = Trip.find(params[:id])
  end

  # GET /trips/new
  # Gestion des nouveaux voyages
  def new
    @trip = Trip.new
  end

  # GET /trips/1/edit
  def edit
  end

  # POST /trips or /trips.json
  # Création d'un voyage
  def create
    @trip = current_user.trips.build(trip_params)

    respond_to do |format|
      if @trip.save
        format.html { redirect_to trip_url(@trip), notice: "Trip was successfully created." }
        format.json { render :show, status: :created, location: @trip }
      else
        format.html { render :new, status: :unprocessable_entity }
        format.json { render json: @trip.errors, status: :unprocessable_entity }
      end
    end
  end

  # PATCH/PUT /trips/1 or /trips/1.json
  # Modification d'un voyage
  def update
    respond_to do |format|
      if @trip.update(trip_params)
        format.html { redirect_to trip_url(@trip), notice: "Trip was successfully updated." }
        format.json { render :show, status: :ok, location: @trip }
      else
        format.html { render :edit, status: :unprocessable_entity }
        format.json { render json: @trip.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /trips/1 or /trips/1.json
  # Suppression d'un voyage
  def destroy
    @trip = Trip.find_by(id: params[:id])
    if @trip
      @trip.destroy
      redirect_to trips_path, notice: 'Voyage supprimé avec succès.'
    else
      redirect_to trips_path, alert: 'Voyage introuvable.'
    end
  end

  private
    # Use callbacks to share common setup or constraints between actions.
    def set_trip
      @trip = Trip.find(params[:id])
    end

    # Only allow a list of trusted parameters through.
    # Paramètres de base d'un voyage
    def trip_params
      params.require(:trip).permit(:destination, :description, :dateDebut, :dateFin)
    end
end
```

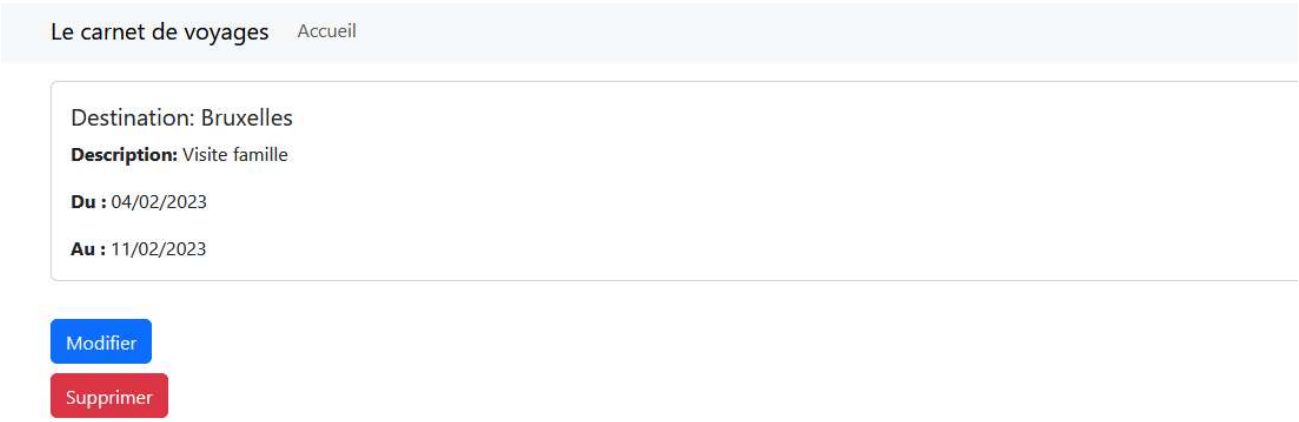
Toutes les actions CRUD ont été créées automatiquement par Scaffold mais j'ai codé moi-même la logique de chaque fonction.

J’ai créé des views assez simple juste pour faire plus propre dans l’affichage, mais toute la logique métier fonctionne très bien. J’ai notamment utilisé l’outil Bootstrap pour construire les views.

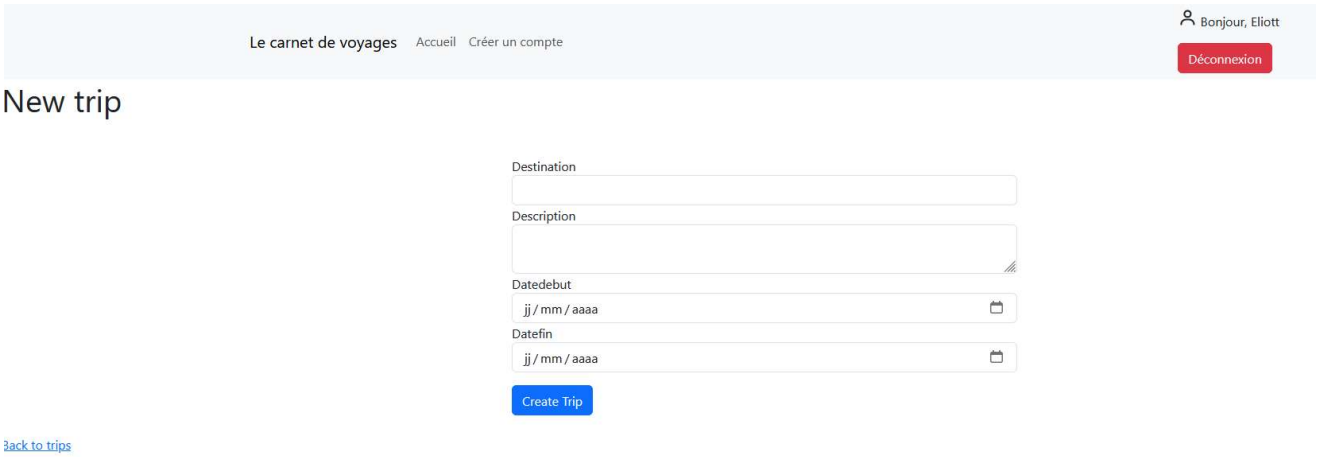
Vue globale :



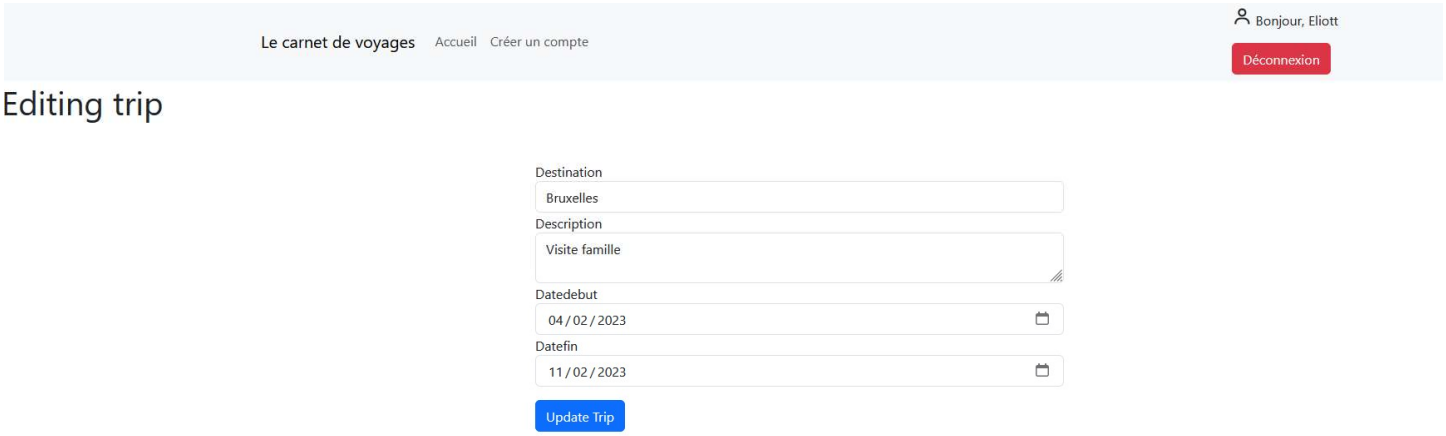
Vue précise sur un voyage :



Création d’un voyage (formulaire avec vérification de cohérence des dates) :



Modification d’un voyage :



À noter que j’ai ajouté la possibilité de s’inscrire en tant que nouvel utilisateur pour poster soi-même un voyage dans le carnet.

J’ai également ajouté des fonctionnalités :

- Un utilisateur ne peut supprimer ou modifier un voyage uniquement si c’est le sien ou qu’il est administrateur
- Seul un administrateur peut donner les droits d’admin à un autre et il peut lui retirer
- Un utilisateur peut modifier les données liées à son compte, son mot de passe lui sera alors redemandé