AUTENTICAZIONE CON RUBY + RAILS

Marco Console
console@diag.uniroma1.it

NOTA PRELIMINARE

- Il seguente materiale didattico è basato su un insieme di lucidi prepato dal **Prof. Leonardo Querzoni.**

- Ringrazio il **Prof. Leonardo Querzoni** per avermi concesso di usare il materiale da lui prodotto.

PREMESSA

Per comprendere a fondo i meccanismi di base della gestione delle identità in rails sarebbe opportuno costruire una soluzione da zero:

- https://www.railstutorial.org/book/sign_up

Questa esercitazione parte dal tag: **`base-autenticazione`**

- https://gitlab.com/console.marco/spoiled-potatoes.git

AUTENTICAZIONE BASE

GEMME PER L'AUTENTICAZIONE

Per implementare meccanismi di autenticazione base è sensato utilizzare uno dei tanti moduli esistenti:

- **Devise**
- Authlogic
- Sorcery
- Clearance
- Doorkeeper
- Monban
- Letmein
- …

FUNZIONALITÀ

Devise mette a disposizione una lunga serie di funzionalità per la gestione degli utenti in una web application:

- Gestisce utenze locali memorizzate su un database in modo sicuro
- Permette l'autenticazione attraverso l'uso di token esterni
- Gestisce l'autenticazione attraverso OAuth
- Permette di richiedere via e-mail la conferma della creazione di un account
- Gestisce il recupero delle password attraverso una procedura sicura
- Permette la completa gestione dell'account da parte dell'utente
- Gestisce sessioni persistenti attraverso cookies
- Traccia le operazione svolte dagli utenti
- Gestisce il timeout delle sessioni scadute
- Controlla la validità dei dati utilizzati dall'utente
- Permette il lockout temporaneo delle utenze

AUTENTICAZIONE BASE

Installazione:

- Aggiungere a Gemfile

```
gem 'devise'
```

- Eseguire bundler

```
bundle install
```

A questo punto Devise dovrebbe essere installato insieme ad una serie di moduli di supporto

- Assicurarsi di utilizzare una versione sufficientemente recente di ruby

AUTENTICAZIONE BASE

Setup:

- Per prima cosa inizializziamo **`devise`** per la nostra applicazione

```
rails generate devise:install
```

- Il comando genera il file config/initializers/devise.rb
- Contiene un gran numero di parametri di configurazione che controllano il funzionamento di devise

```
config.mailer_sender = 'my-email@gmail.com'
config.password_length = 8..128
```

- È utile controllarne il contenuto per completare il setup correttamente
- **NOTA**: è necessario riavviare il server dopo aver modificato questo file

AUTENTICAZIONE BASE

Setup:

- Generiamo un modello che permetta di gestire le identità digitali:

```
rails generate devise User
```

- Questo comando genera app/models/user.rb

```
class User < ApplicationRecord
 # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable and :omniauthable
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :validatable,
end
```

- Genera anche delle nuove routes

```
devise_for :users
```

AUTENTICAZIONE BASE

Setup:

- Viene generato anche un migration script:

```
class DeviseCreateUsers < ActiveRecord::Migration
  def change
    create_table :users do |t|
      ## Database authenticatable
      t.string :email,              null: false, default: ""
      t.string :encrypted_password, null: false, default: ""
      ## Recoverable
      t.string   :reset_password_token
      t.datetime :reset_password_sent_at
      ## Rememberable
      t.datetime :remember_created_at
      ## Trackable
      t.integer  :sign_in_count, default: 0, null: false
      t.datetime :current_sign_in_at
      t.datetime :last_sign_in_at
      t.string   :current_sign_in_ip
      t.string   :last_sign_in_ip
```

AUTENTICAZIONE BASE

ATTENZIONE: se l'applicazione è stata sviluppata per versioni di Rails precedenti la 5 potrebbe mancare la definizione di ApplicationRecord, necessaria per far funzionare Devise

- È sufficiente aggiungere un modello app/models/application_record.rb

```
# app/models/application_record.rb
class ApplicationRecord < ActiveRecord::Base
  self.abstract_class = true
end
```

AUTENTICAZIONE BASE

Setup:

- Bisogna quindi aggiornare lo schema del DB

```
rake db:migrate
```

- Infine è necessario configurare il mailer (in config/environments/development.rb)

```
config.action_mailer.default_url_options = { host:
'localhost', port: 3000 }
```

A questo punto devise è correttamente configurato per gestire le utenze. Mancano due punti:

- Le interfacce di gestione per l’autenticazione (sign in, sign out, sign up, etc.)
- Come forzare l’autenticazione quando viene richiesta una pagina

AUTENTICAZIONE BASE

Setup:

- Generiamo un set di viste:

```
rails generate devise:views users
```

- Questo definisce viste per:
  - Registrare un nuovo utente
  - Gestione della conferma ricevuta via e-mail
  - Reset della password
  - Login
  - Logout
- Le viste, posizionate in app/views/users, possono essere customizzate per adattarle al sito web.

GENERAZIONE VISTE

- Se utilizziamo solo un modello per l'autenticazione, possiamo definire le viste relative all'autenticazione con il seguente commando:

```
rails generate devise:views
```

- Le viste generate in questo modo saranno posizionate nella cartella app/views/devise

AUTENTICAZIONE BASE

Setup:

- Per gestire l’identità degli utenti all’interno dell’applicazione devise ci fornisce una serie di metodi di supporto:
  - authenticate_user!
  - current_user
  - user_signed_in?
  - sign_in(@user)
  - sign_out(@user)
  - user_session

AUTENTICAZIONE BASE

Setup:

- Il metodo authenticate_user! può essere utilizzato in un controller per forzare l'autenticazione dell'utente nella sessione corrente prima che il metodo richiamato sul controller venga eseguito:

```
class MoviesController < ApplicationController

   before_action :authenticate_user!
[…]
```

- Questo garantisce che l'utente sia autenticato prima di permettergli l'accesso alle risorse da proteggere
  - Se l'utente non è autenticato viene reindirizzato alla pagina di sign in
  - Una volta che l'utente si autentica, viene reindirizzato alla route root

AUTENTICAZIONE BASE

Setup:

- Il metodo current_user può essere utilizzato per accedere ad informazioni legate all'identità dell'utente.
- Ad esempio potremmo voler mostrare in tutte le viste il fatto che l'utente è autenticato (codice in app/views/layouts/application.html.erb):

```
<% if user_signed_in? %>
Logged in as <%= current_user.email%> [<%= link_to 'Logout',
destroy_user_session_path, :method => :delete%>]
<% end %>
```

IMPLEMENTAZIONE

- Una possibile implementazione è contenuta nel branch **master** con tag **`base-autorizzazione`**.
  - https://gitlab.com/console.marco/spoiled-potatoes.git

OAUTH

AUTENTICAZIONE OAUTH

Devise gestisce l'autenticazione via OAuth attraverso un apposito modulo.

Setup:

- Aggiungiamo il modulo al Gemfile:

```
gem 'omniauth-facebook'
gem 'omniauth', '~>1.9.1'
```

- Diversi servizi esterni sono accessibili con diversi moduli
- Ricordarsi di eseguire bundle install

```
bundle install
```

AUTENTICAZIONE OAUTH

Setup:

- È quindi necessario modificare il DB per il modello User per aggiungere i campi necessari alla corretta gestione delle identità OAuth:

```
rails g migration AddOmniauthToUsers provider:string uid:string
rake db:migrate
```

- Dobbiamo quindi definire il provider in config/initializers/devise.rb

```
config.omniauth :facebook, '<APP-ID>', '<APP-SECRET>'
```

- I due parametri sono forniti da Facebook quando viene creata una applicazione.

CREARE UN'APP FACEBOOK

1. Creare un'app su http://developers.facebook.com

2. Abilitare la funzionalità di autenticazione nell'app

3. Creare una Test App

4. Copiare app-id e app-secret

AUTENTICAZIONE OAUTH

Setup:

- Aggiorniamo quindi il file del modello per renderlo compatibile con l'autenticazione via OAuth (in app/models/user.rb):

```
class User < ActiveRecord::Base
  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable and :omniauthable
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :trackable, :validatable,
         :omniauthable, :omniauth_providers => [:facebook]

end
```

AUTENTICAZIONE OAUTH

Setup:

- Aggiungiamo anche due funzioni di supporto:

```
def self.from_omniauth(auth)
    where(provider: auth.provider, uid: auth.uid).first_or_create
do |user|
      user.email = auth.info.email
      user.password = Devise.friendly_token[0,20]
    end
  end

  def self.new_with_session(params, session)
    super.tap do |user|
      if data = session["devise.facebook_data"] &&
session["devise.facebook_data"]
["extra"]["raw_info"]
        user.email = data["email"] if user.email.blank?
      end
    end
  end
```

AUTENTICAZIONE OAUTH

Setup:

- Diamo un link per l'autenticazione all'utente (app/views/users/session/new.html.erb):

```
<%= link_to "Sign in with Facebook",
user_omniauth_authorize_path(:facebook) %>
```

- Questo rimanda l'utente a Facebook che al termine farà una callback sul nostro sito che va definita come route (config/routes.rb)

```
devise_for :users, :controllers => { :omniauth_callbacks =>
"users/omniauth_callbacks" }
```

AUTENTICAZIONE OAUTH

Setup:

- Definiamo quindi il controller per la gestione del callback (app/controllers/users/omniauth_callbacks_controller.rb)

```
class Users::OmniauthCallbacksController < Devise::OmniauthCallbacksController
  def facebook
    # You need to implement the method below in your model (e.g. app/models/user.rb)
    @user = User.from_omniauth(request.env["omniauth.auth"])

    if @user.persisted?
      sign_in_and_redirect @user, :event => :authentication
      set_flash_message(:notice, :success, :kind => "Facebook") else
      session["devise.facebook_data"] = request.env["omniauth.auth"]
      redirect_to new_user_registration_url
    end
  end

  def failure
    redirect_to root_path
  end
end
```

FLASH MESSAGES

Aggiungiamo il rendering dei flash message

- In app/views/layouts/application.html.erb

```
<% flash.each do |type, msg| %>
  <div>
    Notification: <%= msg %>
  </div>
<% end %>
```

IMPLEMENTAZIONE

- Una possibile implementazione è contenuta nel branch **feature_oauth-facebook**.
  - https://gitlab.com/console.marco/spoiled-potatoes.git

ESERCIZI

ESERCIZIO 1

1. Utilizzando l’applicazione che abbiamo creato, trasportare le azioni di Moviegoer a User
   - Aggiunta film e recensioni con il nome utente autenticato (email) .

ESERCIZIO 1 -- SOLUZIONE

- Modifichiamo il database

```
rails g migration UsersAndMoviegoers
```

```
class UsersAndMoviegoers < ActiveRecord::Migration[6.1]
  def change
    remove_column :reviews, :moviegoer_id
    add_column :reviews, :user_id, :reference
    drop_table :moviegoers
  end
end
```

ESERCIZIO 1 -- SOLUZIONE

- Aggiorniamo il modello User

```
has_many :reviews
```

- Aggiorniamo il modello Review

```
belongs_to :user
```

ESERCIZIO 1 -- SOLUZIONE

- Aggiungiamo una route per visualizzare le review degli utenti
  - **`config/routes.rb`**

```
 get '/users/reviews/:id', to:
'reviews#user_reviews', :as => :user_reviews
```

ESERCIZIO 1 -- SOLUZIONE

- Aggiorniamo una funzione nel controller
  - **app/controllers/reviews_controller.rb**

```
def user_reviews
    @user = User.find(params[:id])
 end
```

```
 private
    def review_params
      p = params.require(:review).permit(:body,
:score, :user)
      {:body=> p[:body], :score=>p[:score],
:user=>current_user}
    end
```

ESERCIZIO 1 -- SOLUZIONE

- Aggiungiamo una view per il controller
  - **app/views/reviews/user_reviews.html.erb**

```
<h1> <%= @user.email %> </h1>
<hr>
<h2> Reviews </h2>
<% if @user.reviews.empty? %>
 <h3> No reviews yet! </h3>
<% end %>
<%@user.reviews.each do |review| %>
   <p>
     <bf>Body: </bf><%= review.body %> <br>
     <bf>Score: </bf><%= review.score %> <br>
    <bf>Movie: </bf><%= review.movie.title %> <br>
  </p>
```

ESERCIZIO 1 -- SOLUZIONE

- Aggiorniamo la view per Movie#show
  - **`app/views/users/show.html.erb`**

```
<bf>Author: </bf><%= link_to review.user.email,
user_reviews_path(review.user)  %> <br>
```

- Rimuoviamo tutti i riferimenti a Moviegoer
  - **`app/views/reviews/edit.html.erb`**
  - **`app/views/users/show.html.erb`**

ESERCIZIO 1 -- SOLUZIONE

1. Una possibile implementazione è contenuta nel branch **feature_users-as-moviegoers**.
   - https://gitlab.com/console.marco/spoiled-potatoes.git

ESERCIZIO 2

- Aggiungete a spoiled-potatoes il login con credenziali google
  - Partite dal codice che avete scritto **oppure**
  - Partite dal commit **`base-autorizzazione`**

- Utilizzate la gemma **`omniauth-google-oauth2`**
  - https://github.com/zquestz/omniauth-google-oauth2

- Per problemi di compatibilità usare la versione di **`omniauth`** utilizzata per l'esempio precedente

```
gem 'omniauth', '~>1.9.1'
```

ESERCIZIO 2 -- SOLUZIONE

- Creiamo un’app google
  - https://console.cloud.google.com
- Abilitiamo omniauth authentication
  - «Schermata consenso OAuth»
- Creiamo una chiave per il login
  - «Credenziali»
- Autorizziamo l’url di callback in «Credenziali»
  - http://localhost:3000/users/auth/google_oauth2/callback
- Aggiungiamo gli utenti di prova autorizzati

ESERCIZIO 2 -- SOLUZIONE

- Aggiungiamo le informazioni sulla nostra applicazione google nel file di configurazione di **devise**
  - config/initializers/devise.rb

```
config.omniauth :google_oauth2, 'CLIENT_ID', CLIENT_SECRET', {}
```

- Aggiungiamo una route apposita
  - config/routes.rb

```
devise_for :users, controllers: { omniauth_callbacks:
'users/omniauth_callbacks' }
```

- Aggiorniamo il modello
  - app/models/user.rb

```
devise :omniauthable, omniauth_providers: [:google_oauth2]
```

ESERCIZIO 2 -- SOLUZIONE

- Aggiungiamo un callback controller
  - app/controllers/users/omniauth_callbacks_controller.rb

```
class Users::OmniauthCallbacksController <
Devise::OmniauthCallbacksController
  def google_oauth2
    @user = User.from_omniauth(request.env['omniauth.auth’])

   flash[:notice] = I18n.t 'devise.omniauth_callbacks.success',
kind: 'Google’

        sign_in_and_redirect @user, event: :authentication
  end
end
```

ESERCIZIO 2 -- SOLUZIONE

- Aggiungiamo una funzionalità nel modello
  - app/models/user.rb

```
def self.from_omniauth(access_token)
      data = access_token.info
      user = User.where(email: data['email']).first
      unless user
          user = User.create( email: data['email'],
             password: Devise.friendly_token[0,20]
          )
      end
    user
end
```

ESERCIZIO 2 -- SOLUZIONE

- Diamo un link per l'autenticazione all'utente
  - app/views/users/session/new.html.erb

```
<%= link_to "Sign in with Google",
user_google_oauth2_omniauth_authorize_path %>
```

ESERCIZIO 2 -- SOLUZIONE

- Una possibile implementazione è contenuta nel branch **feature_oauth-google**.
  - https://gitlab.com/console.marco/spoiled-potatoes.git