Documentazione

Link github: https://github.com/MicheleAmoroso/SportsBook

User Stories:

1. As a registered user that is not logged I want to login with email or username

2. As an unregistered user I want to sign up using Facebook account so that I can become a registered user

3. As an uregistered user i want to sing up using my email.

4. As a registered user I want to set an avatar so that I can show an image representing me

5. As a registered user i want to be able to edit my account so that i can change info about me

6. As a registered user I want to have setting so that I can delete my account

7. As a registered user I want to have setting so that I can change my password

8. As a registered user I want to have setting so that I can recover my password

9. As a player i want to have setting so that i can book a sports ground.

10. As a player i want to have setting so that i can cancel my reservation.

11. As a player I want to leave a comment on the sports ground's page.

12. As a player I want to rate the sports ground.

13. As a player I want to delete a comment on the sports ground's page.

14. As a registered user I want to add to my favorites list my favorites sports ground.

15. As a registered user I want to delete from my favorites list a sports ground.

16. As a registered user i want to be able to search the sports ground by name.

17. As a registered user i want to be able to search the sports ground by price.

18. As an proprietor i want to be able to delete a ground.

19. As an proprietor i want to be able to edit a ground.

20. As an proprietor i want to be able to add a ground's book.

21. As an <u>proprietor</u> i want to be able to remove a ground's book.

22. As an <u>admin</u> i want to be able to delete a comment.

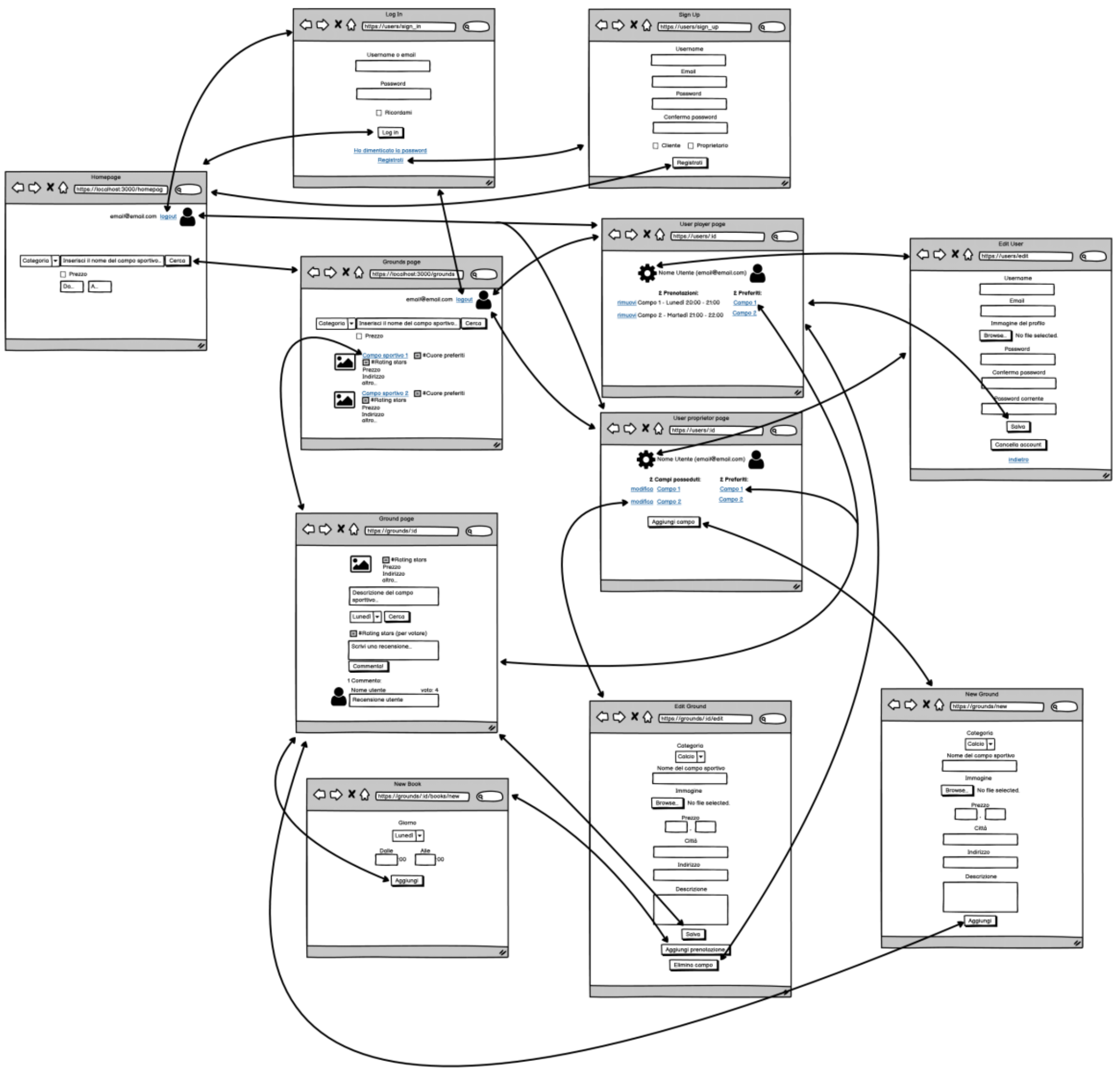
23. As an <u>admin</u> i want to be able to delete a reservation.

24. As an <u>admin</u> I want to be able to add a sports ground.

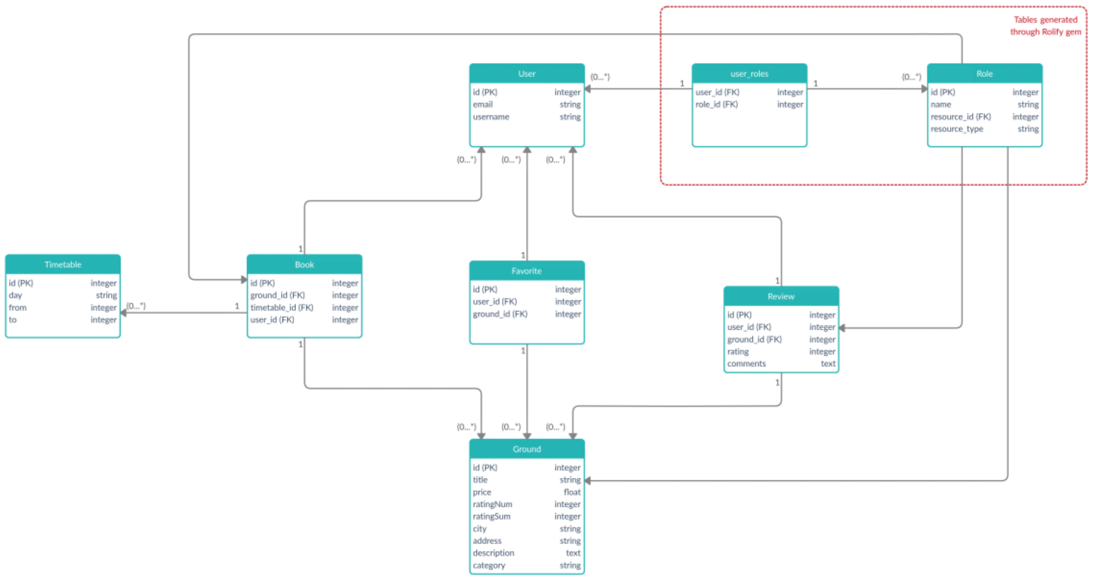
25. As an <u>admin</u> I want to be able to remove a sports ground.

26. As an <u>admin</u> I want to be able to edit the sports ground's info.

Mockup:



Schema DB (UML):



Struttura controllo degli accessi (ruoli):

	Ground	Review	Favorite	Timetable	Book	User
Admin	CRUD	CRUD	CRUD	CRUD	CRUD	CRUD
Proprietor	CRUD	R	CRD	CRD	CRD	U
Player	R	CRD	CRD	R	RU	U

C = Create

R = Read

U = Update

D = Delete

*** Un Proprietor può effettuare un'operazione di Delete su Timetable, Book e Ground solo se ne è il creatore => ruolo "Owner"**

*** Un Player può effettuare un'operazione di Delete su Review solo se ne è scrittore => ruolo "Writer"**

*** Un Player può effettuare un'operazione di Update (disdendo la prenotazione) su Book solo se ne è il prenotatore => ruolo "Booker"**

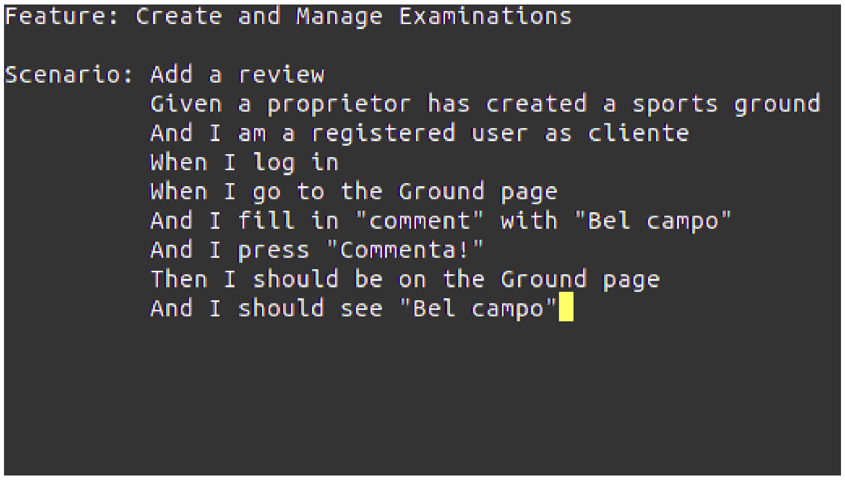
Piano dei test

Sono stati effettuati dei test su alcune funzionalità dell'applicazione web. In particolare ci siamo serviti di due strumenti:

- **Cucumber**, con cui abbiamo effettuato i test di integrazione grazie anche ai metodi offerti dalla gemma **Capybara.**
- **RSpec**, con cui abbiamo effettuati gli unit test su alcuni models presenti all'interno dell'applicazione e su alcuni controllers.

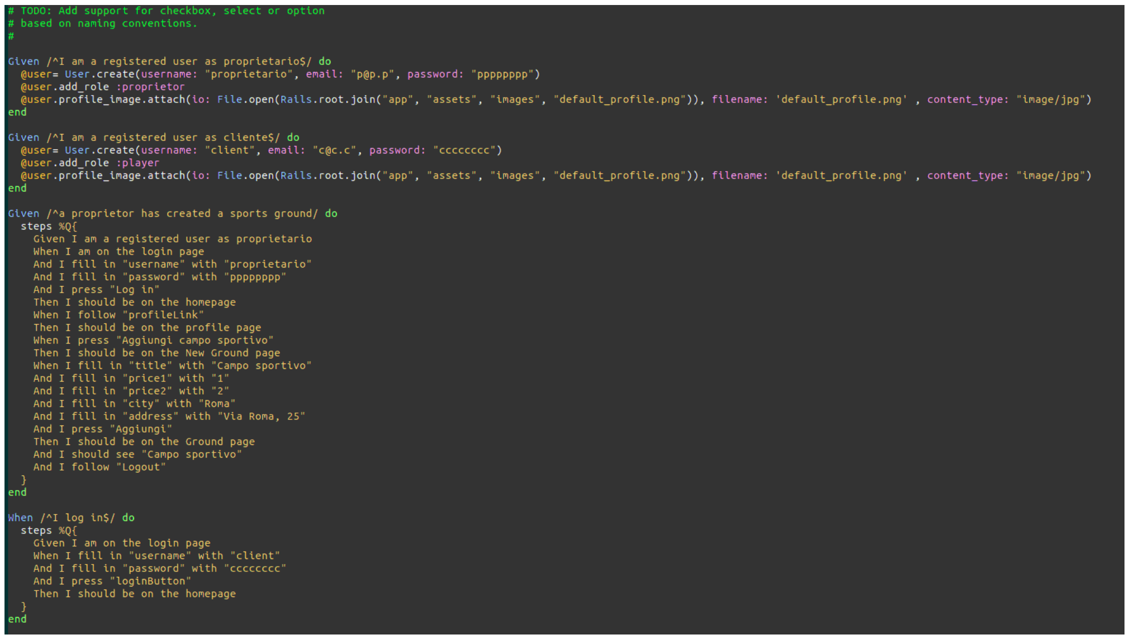
Esempi:

User story (AddReview)



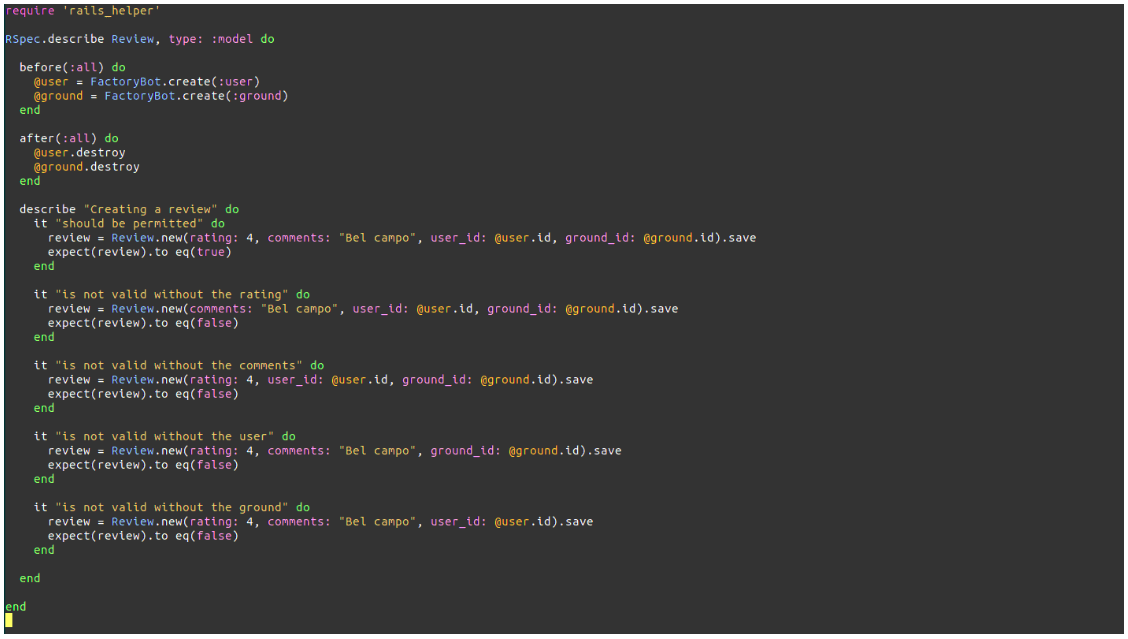
```
Feature: Create and Manage Examinations

Scenario: Add a review
          Given a proprietor has created a sports ground
          And I am a registered user as cliente
          When I log in
          When I go to the Ground page
          And I fill in "comment" with "Bel campo"
          And I press "Commenta!"
          Then I should be on the Ground page
          And I should see "Bel campo"
```

step definition (web_steps.rb)



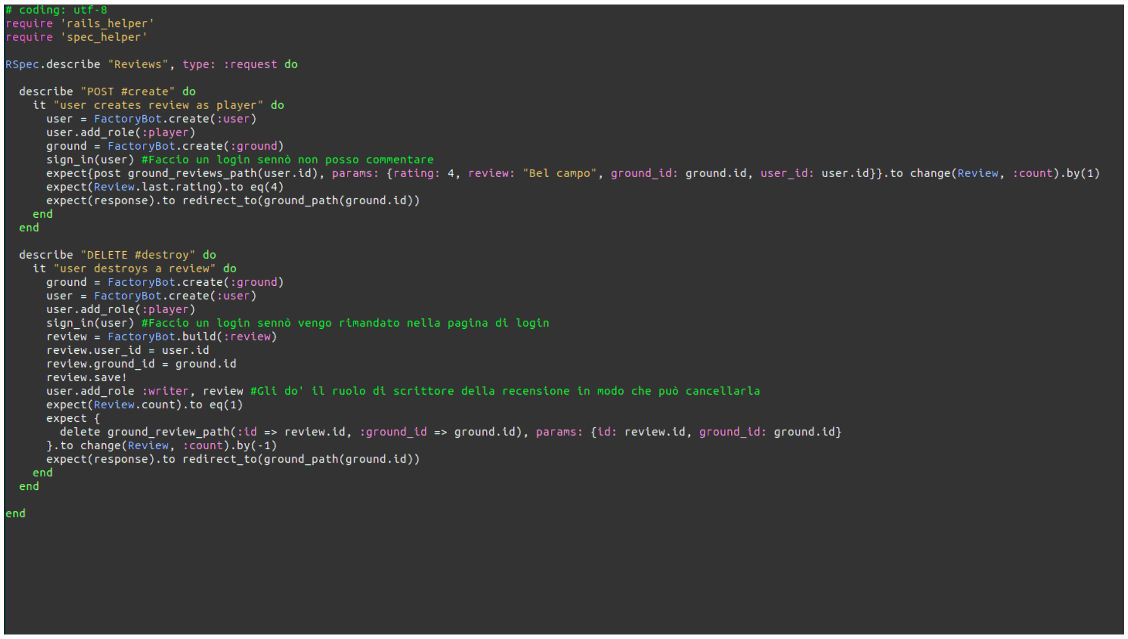
```ruby
# TODO: Add support for checkbox, select or option
# based on naming conventions.
#

Given /^I am a registered user as proprietario$/ do
  @user= User.create(username: "proprietario", email: "p@p.p", password: "pppppppp")
  @user.add_role :proprietor
  @user.profile_image.attach(io: File.open(Rails.root.join("app", "assets", "images", "default_profile.png")), filename: 'default_profile.png' , content_type: "image/jpg")
end

Given /^I am a registered user as cliente$/ do
  @user= User.create(username: "client", email: "c@c.c", password: "cccccccc")
  @user.add_role :player
  @user.profile_image.attach(io: File.open(Rails.root.join("app", "assets", "images", "default_profile.png")), filename: 'default_profile.png' , content_type: "image/jpg")
end

Given /^a proprietor has created a sports ground/ do
  steps %Q{
    Given I am a registered user as proprietario
    When I am on the login page
    And I fill in "username" with "proprietario"
    And I fill in "password" with "pppppppp"
    And I press "Log in"
    Then I should be on the homepage
    When I follow "profileLink"
    Then I should be on the profile page
    When I press "Aggiungi campo sportivo"
    Then I should be on the New Ground page
    When I fill in "title" with "Campo sportivo"
    And I fill in "price1" with "1"
    And I fill in "price2" with "2"
    And I fill in "city" with "Roma"
    And I fill in "address" with "Via Roma, 25"
    And I press "Aggiungi"
    Then I should be on the Ground page
    And I should see "Campo sportivo"
    And I follow "Logout"
  }
end

When /^I log in$/ do
  steps %Q{
    Given I am on the login page
    When I fill in "username" with "client"
    And I fill in "password" with "cccccccc"
    And I press "loginButton"
    Then I should be on the homepage
  }
end
```

Model RSpec (review_spec.rb)



```ruby
require 'rails_helper'

RSpec.describe Review, type: :model do

  before(:all) do
    @user = FactoryBot.create(:user)
    @ground = FactoryBot.create(:ground)
  end

  after(:all) do
    @user.destroy
    @ground.destroy
  end

  describe "Creating a review" do
    it "should be permitted" do
      review = Review.new(rating: 4, comments: "Bel campo", user_id: @user.id, ground_id: @ground.id).save
      expect(review).to eq(true)
    end

    it "is not valid without the rating" do
      review = Review.new(comments: "Bel campo", user_id: @user.id, ground_id: @ground.id).save
      expect(review).to eq(false)
    end

    it "is not valid without the comments" do
      review = Review.new(rating: 4, user_id: @user.id, ground_id: @ground.id).save
      expect(review).to eq(false)
    end

    it "is not valid without the user" do
      review = Review.new(rating: 4, comments: "Bel campo", ground_id: @ground.id).save
      expect(review).to eq(false)
    end

    it "is not valid without the ground" do
      review = Review.new(rating: 4, comments: "Bel campo", user_id: @user.id).save
      expect(review).to eq(false)
    end

  end

end
```

Controller RSpec (reviews_request_spec.rb)



```ruby
# coding: utf-8
require 'rails_helper'
require 'spec_helper'

RSpec.describe "Reviews", type: :request do

  describe "POST #create" do
    it "user creates review as player" do
      user = FactoryBot.create(:user)
      user.add_role(:player)
      ground = FactoryBot.create(:ground)
      sign_in(user) #Faccio un login sennò non posso commentare
      expect{post ground_reviews_path(user.id), params: {rating: 4, review: "Bel campo", ground_id: ground.id, user_id: user.id}}.to change(Review, :count).by(1)
      expect(Review.last.rating).to eq(4)
      expect(response).to redirect_to(ground_path(ground.id))
    end
  end

  describe "DELETE #destroy" do
    it "user destroys a review" do
      ground = FactoryBot.create(:ground)
      user = FactoryBot.create(:user)
      user.add_role(:player)
      sign_in(user) #Faccio un login sennò vengo rimandato nella pagina di login
      review = FactoryBot.build(:review)
      review.user_id = user.id
      review.ground_id = ground.id
      review.save!
      user.add_role :writer, review #Gli do' il ruolo di scrittore della recensione in modo che può cancellarla
      expect(Review.count).to eq(1)
      expect {
        delete ground_review_path(:id => review.id, :ground_id => ground.id), params: {id: review.id, ground_id: ground.id}
      }.to change(Review, :count).by(-1)
      expect(response).to redirect_to(ground_path(ground.id))
    end
  end

end
```