EventManager

Ingegneria del Software Avanzata - Lucia Ferrari

https://github.com/dream-19/Ruby_On_Rails_Web_App

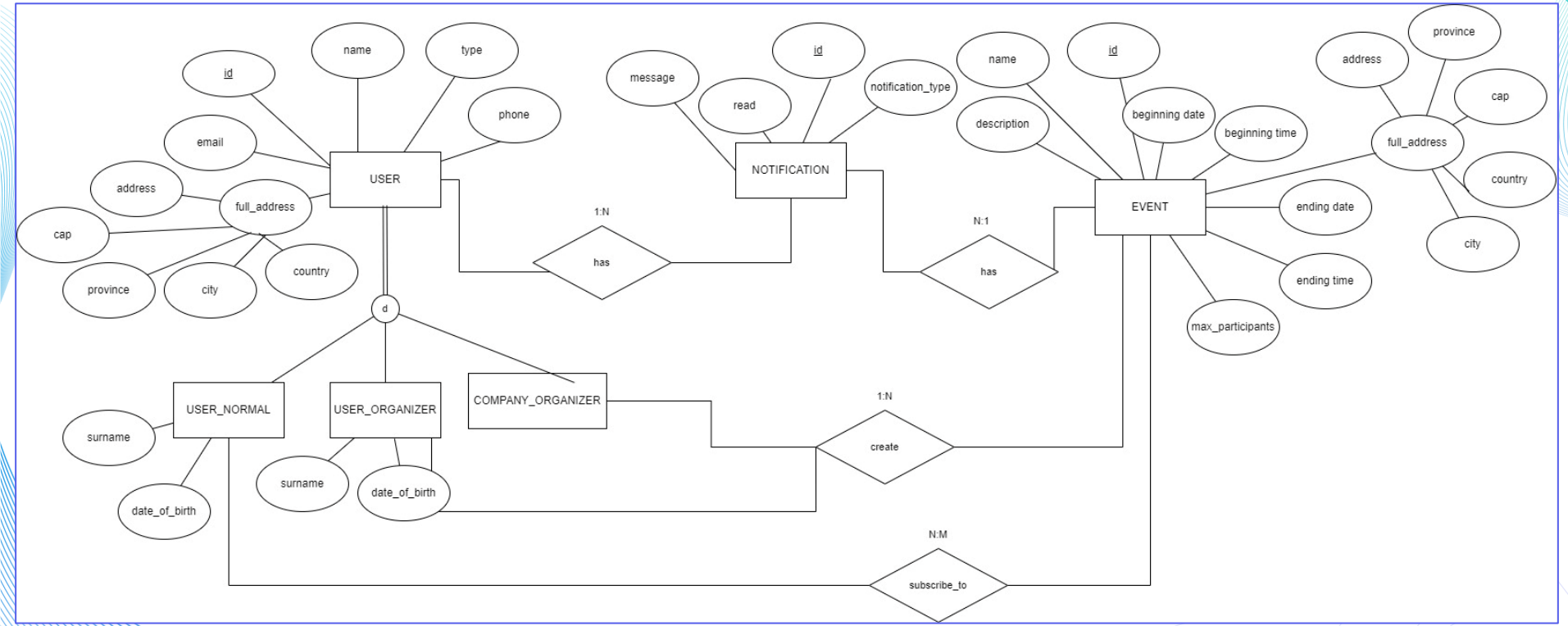
Indice

01

EventManager

EventManager è un'applicazione web che ha come scopo quello di gestire la **prenotazione ad eventi**.
Sono presenti degli utenti organizzatori che creano eventi e degli utenti normali che si iscrivono/disiscrivono ad eventi

Diagramma ER



Specifiche (Informali)

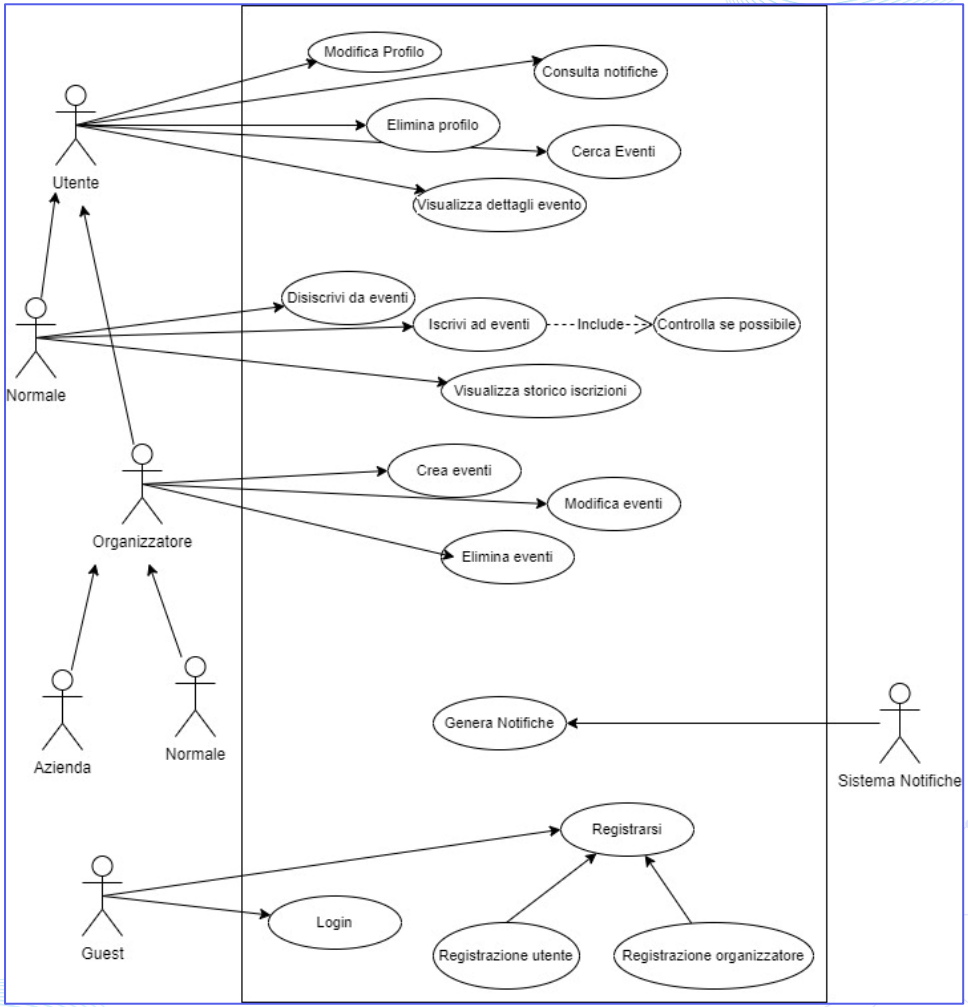
Un **organizzatore** può:
- Registrarsi (come utente o come azienda)
- Creare/modificare ed eliminare eventi specificando una massima capacità
- Visualizzare gli eventi creati
- Consultare e rimuovere le iscrizioni ai propri eventi

Un **utente normale** può:
- Registrarsi
- Iscriversi/disiscriversi da eventi
- Visualizzare gli eventi a cui si è iscritto
- Non possono iscriversi ad eventi i cui periodi di svolgimento si sovrappongono
- Non è possibile iscriversi ad eventi passati

Tutti gli utenti possono:
- Modificare il proprio profilo o eliminarlo
- Ricercare e visualizzare eventi tramite vari filtri
- Consultare le proprie notifiche e segnarle come lette

Il **sistema di notifiche**:
- Crea notifiche per utenti organizzatori e normali, in base alle azioni svolte (creazione/modifica eventi, registrazione ad eventi, raggiunta la massima capacità di un evento...)

UML – Diagramma dei casi d'uso



Tecnologie e Strumenti Usati

- **Ruby**
- **Ruby on Rails**
- **Javascript e Jquery**
- **Bootstrap 5**

Backend & Frontend

- **Git e Github**
- **Docker e docker-compose**

Generali

- **Gemfile**

Gestore dipendenze

- **MySQL**

DBMS

Tecnologie e Strumenti Usati

Gemfile: è un file che permette di gestire tutte le dipendenze di un progetto in ruby on rails

```
group :development, :test do
  gem "debug", platforms: %i[ mri windows ]
  gem 'factory_bot_rails'
  gem 'faker'
  gem 'rspec-rails'
  gem 'capybara'
  gem 'selenium-webdriver'
  gem 'webdrivers'
  gem 'dotenv-rails'
  gem 'simplecov', require: false
  gem 'rails-controller-testing'
end
```

Una volta elencate le gemme necessarie (le librerie) e le version richieste, è sufficiente eseguire:

- **Bundle install**
- **Bundle update**

Per installarle e aggiornarle automaticamente

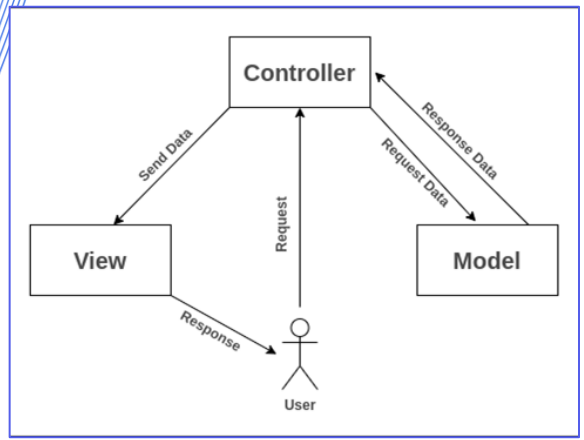
Gemme principali utilizzate:

- **Devise** -> autenticazione
- **Mysql2** -> connessione mysql
- **Bootstrap**
- **Jquery-rails**
- **Factory_bot_rails** -> dati d'esempio
- **Faker** -> dati finti
- **Rspec-rails** -> testing
- **Capybara** -> testing
- **Simplecov** -> testing coverage

02

Implementazione

L’applicazione è implementata seguendo il pattern del **model-view-controller**



Sono presenti 4 modelli principali:
- User (con le 3 sottoclassi)
- Event
- Subscription
- Notification

```ruby
class Notification < ApplicationRecord
  belongs_to :user
  belongs_to :event, optional: true

  validates :message, presence: true
  validates :notification_type, presence: true

  # number of read notifications of an user
  def self.read
    where(read: true).count
  end

  #number of unread notifications of an user
  def self.unread
    where(read: false).count
  end
end
```

Esempio: modello delle notifiche

> L’approccio seguito è quello del **fat model**

> L’**autenticazione** è completamente gestita dalla gemma devise

Testing

Generazione dei dati di testing tramite le gemme: **FactoryBot** e **Faker**

Testing effettivo con le gemme: **RSpec** e **Capybara**

Testing coverage con le gemme: **SimpleCov**

```
You, 4 seconds ago | 1 author (You)
FactoryBot.define do
    factory :notification do
        message { Faker::Lorem.sentence }
        notification_type { "info" }
        read { false }
        event { nil }
        association :user
    end
end       You, last month • testing models
```

Creazione delle notifiche con factory bot e faker

Testing

Unit Testing

Su modelli e controller per intero

Test di integrazione

Sulle funzionalità principali dell'applicazione

Test Coverage

98% con 275 test

```
(base) lucia@LAPTOP-OVTQBK55:~/EventManager$ bundle exec rspec
.....................................................................................................
.....................................................................

Finished in 47.43 seconds (files took 6.68 seconds to load)
275 examples, 0 failures

Coverage report generated for RSpec to /home/lucia/EventManager/coverage. 1957 / 1997 LOC (98.0%) covered.
```

Unit Testing

Sui modelli è stata testata in particolare tutta la **validazione**

```ruby
#Testing per event
RSpec.describe Event, type: :model do
  #Test for the event attribute validations
  describe "Event Attribute Validations" do
    let(:event) { build(:event) }
      You, last month • testing models
    it "is valid with valid attributes" do
      expect(event).to be_valid
    end

    it "is invalid without a name" do
      event.name = nil
      expect(event).not_to be_valid
      expect(event.errors[:name]).to include("can't be blank")
    end

    it "is invalid without a beginning time" do
      event.beginning_time = nil
      expect(event).not_to be_valid
      expect(event.errors[:beginning_time]).to include("can't be blank")
    end
```

Parte del testing sul modello ‘Event’

```ruby
it 'user can\'t subscribe to an event that is already full' do
  user = create(:user_normal)
  user2 = create(:user_normal)
  event = create(:event, max_participants: 1)
  create(:subscription, user: user2, event: event)
  subscription = build(:subscription, user: user, event: event)

  expect(subscription.valid?).to be false
  expect(subscription.errors[:base]).to include("The event is already full")
end

it 'user organizer can\'t subscribe to an event if they are an organizer' do
  user = create(:user_organizer)
  event = create(:event)
  subscription = build(:subscription, user: user, event: event)

  expect(subscription.valid?).to be false
  expect(subscription.errors[:base]).to include("You are an organizer and cannot subscribe to events")
end
```

Parte del testing sul modello ‘Subscription’

03 Unit Testing

Property based testing per controllare che un utente non si possa iscrivere ad eventi sovrapposti ad altri in cui è già iscritto

```ruby
def random_dates()
  beginning = rand(Date.today - 2.year..Date.today + 2.year)
  ending =  beginning > Date.today ? rand(beginning..beginning + 2.year) : rand(Date.tomorrow..Date.tomorrow + 2.year)
  [beginning,ending]
end

# Generate a random beginning date and ending date for and overlapping event
# at least 1 date is overlapping (es: beginning_date is inside the range of the old event,
# ending_date is outside the range of the old event)
def generate_event_dates(start_rand, end_range)
  case rand(0..3)
    # 1 ) both beginning and ending date are inside the range of the old event
    when 0
      date1 = rand(start_rand..(end_range-1.day))
      date2 =  date1 > Date.today ? rand(date1..(end_range-1.day)) : rand(Date.tomorrow..end_range)
    # 2 ) beginning date is inside the range of the old event, ending date is outside the range of the old event
    when 1
      date1 = rand(start_rand..(end_range-1.day))
      date2 = rand (end_range..end_range + 2.year)
    # 3) beginning date is outside the range of the old event, ending date is inside the range of the old event
    when 2
      date1 = rand(start_rand - 2.years..start_rand)
      if start_rand < Date.today
        date2 = rand(Date.today..end_range)
      else
        date2 = rand((start_rand+1.day)..end_range)
      end
    # 4) the new event contains the old event
    when 3
      date1 = rand(start_rand - 2.years..start_rand)
      date2 = rand(end_range..end_range + 2.years)
    end
    [date1, date2]
end
```

Generazione di intervalli non validi

```ruby
it "(PBT): new event overlaps with a subscribed event (can't subscribe)" do
  user = create(:user_normal)
  dates = random_dates() # Randomize the dates

  # subcribed event
  overlapping_event = create(:event,
    beginning_date: dates.first,
    ending_date: dates.last,
  )
  create(:subscription, user: user, event: overlapping_event)

  # test 500 cases
  500.times do
    new_event_dates = generate_event_dates(dates.first, dates.last)
    new_event_beginning = new_event_dates.first
    new_event_ending = new_event_dates.last
    new_event = build(:event, beginning_date: new_event_beginning.to_date, ending_date: new_event_ending.to_date)
    user.reload  # Reload to ensure the user's subscriptions are up-to-date

    # Try to subscribe to the new event
    new_subscription = build(:subscription, user: user, event: new_event)

    # Assertions
    expect(new_subscription.valid?).to be_falsey
    expect(new_subscription.errors[:base]).to include("You are already subscribed to an event that overlaps with
  end
end
```

Testing che deve sempre fallire con 500 intervalli sovrapposti random

03

Unit Testing

Testing sui metodi principali dei controller

Testing del metodo #index controllando la ricerca per eventi con parametri ‘on going’ e ‘not full’

Mocking dei metodi per restituire gli eventi corretti tramite allow

```
context "with on_going parameter" do
  before do
    allow(Event).to receive(:ongoing).and_return([ongoing_event])
  end

  it "returns only ongoing events" do
    get :index, params: { on_going: true }
    expect(assigns(:events)).to eq([ongoing_event])
  end
end

context "with not_full parameter" do
  before do
    allow(Event).to receive(:notfull).and_return([not_full_event])
  end

  it "returns only not full events" do
    get :index, params: { not_full: true }
    expect(assigns(:events)).to eq([not_full_event])
  end
end
```

03

Integration Testing

Testing delle azioni principali da eseguire nell'interfaccia

```
scenario "User logs in successfully" do
  visit new_user_session_path

  fill_in "Email", with: user.email
  fill_in "Password", with: user.password
  click_button "Log in"

  expect(page).to have_text("Signed in successfully.")
```

Test: login corretto dell'utente

Test: utente non può modificare il profilo se non inserisce la password corretta

```
scenario "User fails to edit profile with wrong credentials" do
  login_as(user, scope: :user)

  visit edit_user_registration_path

  fill_in "Email", with: "newemail@edit.it"
  fill_in "current-password", with: "wrongpassword"
  click_button "Update"

  expect(page).to have_text("Current password is invalid")
end
```

03

Integration Testing

Testing delle azioni principali da eseguire nell'interfaccia

```
scenario "Organizer deletes event successfully" do
  visit event_path(event)

  click_link "Delete" #trigger modal
  id_modal = '#deleteEvent' + event.id.to_s
  within(id_modal) do
    click_link 'Confirm Deletion'
  end

  expect(page).to have_content("Event was successfully destroyed.")
end
```

Test: organizzatore elimina un evento (conferma tramite modale)

Test: organizzatore modifica il nome dell'evento

```
scenario "Organizer edits an event successfully" do
  visit edit_event_path(event)

  fill_in "Name", with: "Updated Tech Conference"
  click_button "Save Changes"

  expect(page).to have_content("Event was successfully updated.")
  expect(page).to have_content("Updated Tech Conference")
end
```

04

Deployment in Container

Realizzato tramite docker e docker-compose

DockerFile

```
FROM ruby:3.3

# Run update and install build-essential, apt-utils, libpq-dev and nodejs
RUN apt-get update -qq && apt-get install -y build-essential apt-utils libpq-dev nodejs

# Create a directory /myapp
WORKDIR /myapp

# Copy the Gemfile and Gemfile.lock from the current directory into the /myapp directory
RUN gem install bundler
COPY Gemfile* ./

# Install the gems
RUN bundle install

# Set the environment variable RAILS_ENV to development
ARG DEFAULT_PORT 3010
EXPOSE ${DEFAULT_PORT}
```

04

Deployment in Container

Realizzato tramite docker e docker-compose

```yaml
db_development:
  image: mysql:latest
  volumes:
    - mysql-data-dev:/var/lib/mysql
  environment:
    MYSQL_ROOT_PASSWORD: password
    MYSQL_DATABASE: EventManager_development
    MYSQL_USER: user
    MYSQL_PASSWORD: password
  ports:
    - "3307:3306" # 3307 is the host port, 3
  restart: always
```

Docker-compose:

Gestione di 3 container:

- **Database development**
- **Database testing**
- **Applicazione web**

```yaml
db_test:
  image: mysql:latest
  volumes:
    - mysql-data-test:/var/lib/mysql
  environment:
    MYSQL_ROOT_PASSWORD: password
    MYSQL_DATABASE: EventManager_test
    MYSQL_USER: user
    MYSQL_PASSWORD: password
  ports:
    - "3308:3306"
  restart: always
```

```yaml
web:
  build: .
  env_file:
    - .env.container
  command: >
    bash -c "rm -f tmp/pids/server.pid && rails db:prepare && rails db:seed && rails s -p 3000 -b '0.0.0.0'"
  volumes:
    - .:/myapp
  ports:
    - "3010:3000"
  depends_on:
    - db_development
    - db_test
  restart: always
```

05

Repository Git e Github

Il codice è stato salvato in un **repository git strutturato** ed è presente anche su **github**

dream-19 update pbt	348abc9 · 3 hours ago	125 Commits
app	integration test end	3 days ago
bin	initial	2 months ago
config	fix docker-compose	last week
db	docker compose completed	3 weeks ago
lib	initial	2 months ago
log	initial	2 months ago
public	initial	2 months ago
spec	update pbt	3 hours ago
storage	initial	2 months ago
test	notifications	last month
vendor	events index and edit	2 months ago
.dockerignore	initial	2 months ago
.env.container	fix docker-compose	last week
.gitattributes	initial	2 months ago
.gitignore	env container	2 weeks ago
.ruby-version	initial	2 months ago
Dockerfile	fix dockerfile	4 days ago
Gemfile	test event controller	last week
Gemfile.lock	integration test end	3 days ago
README.md	update pbt	3 hours ago

https://github.com/dream-19/Ruby_On_Rails_Web_App

Pipeline CI/CD tramite github action

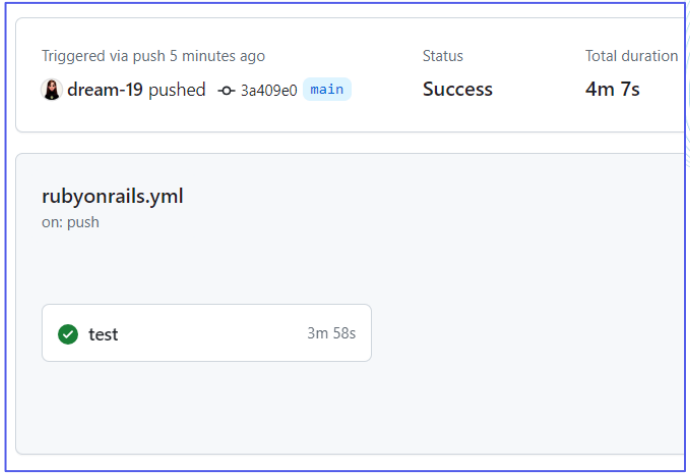
Esecuzione dei **test** → Push dell'immagine su **docker hub**

```yaml
- name: Run Tests
  env:
    RAILS_ENV: test
  run: |
    bundle exec rspec

- name: Login to DockerHub
  uses: docker/login-action@v3
  with:
      username: ${{ secrets.DOCKER_USERNAME }}
      password: ${{ secrets.DOCKER_PASSWORD }}

- name: Build and push Docker image
  uses: docker/build-push-action@v5
  with:
      context: .
      push: true
      tags: ${{ secrets.DOCKER_USERNAME }}/railsapp:lastest
```

Step principali della pipeline



FINE

Grazie per l’attenzione