Proyecto

Genaro Salomone
Cristian Herrera

SRS

Funcionalidades principales del sistema

-1) Gestión de usuario (Username/Email - Password / Google / Facebook).

-2) Selección de dificultad (Principiante - Experto).

-3) Diferentes tipos de preguntas (Choice - Autocompletado - True/False).

-4) Después de una trivia, proporcionar información sobre si la respuesta fue correcta (o no), y ademas, mostrar la respuesta correcta.

-5) Bonus de puntos si se responde de velozmente una cierta cantidad de preguntas.

-6) Selección de preguntas aleatorio de un banco de preguntas.

-7) Si se elige dificultad principiante, se permitirá un sistema de ayudas/ pistas con información adicional de la pregunta (incluso saltear una pregunta)

-8) Ranking de usuarios.

-9) Timer para preguntas: cada pregunta tendrá un timer con determinado tiempo para responder según su dificultad.

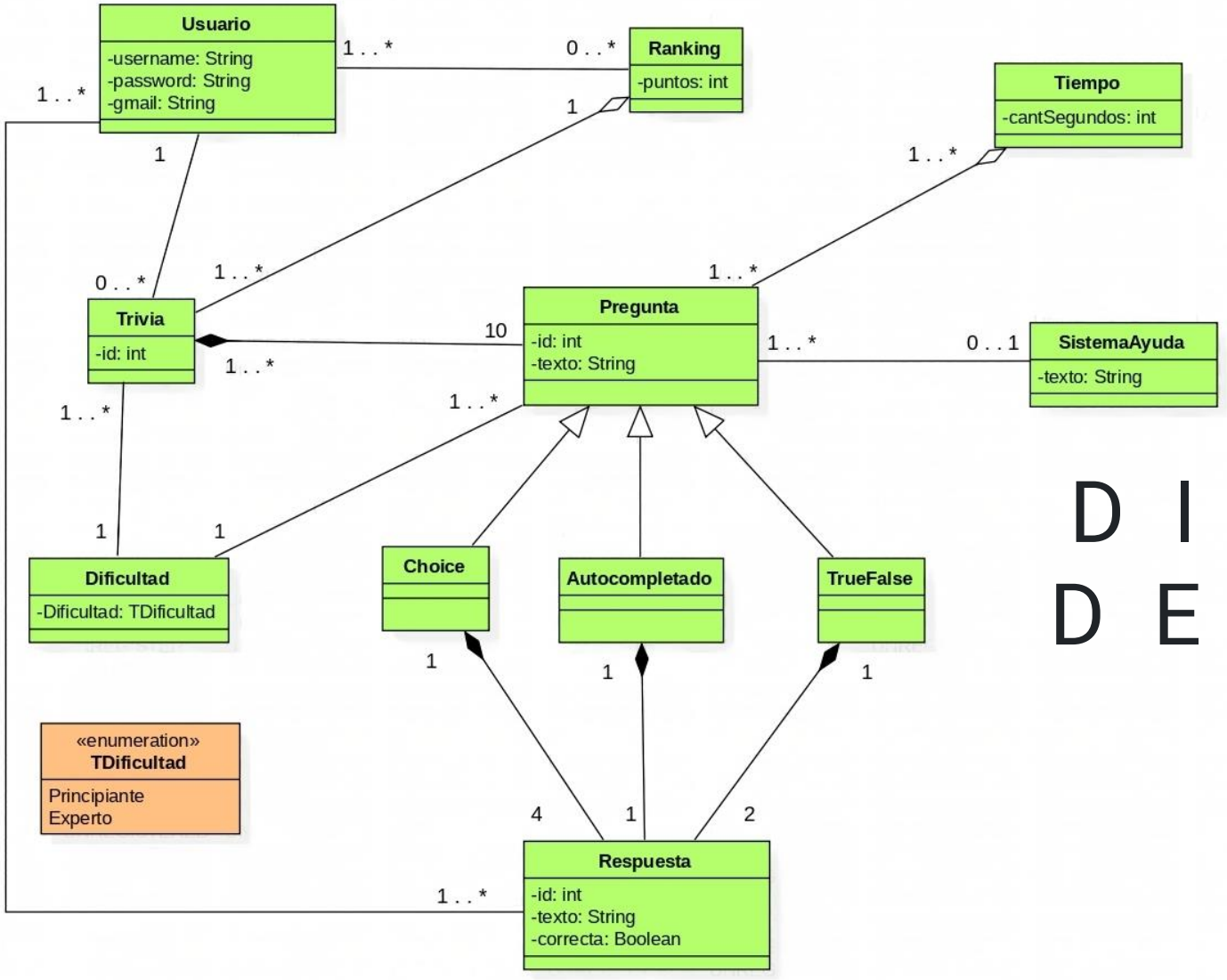
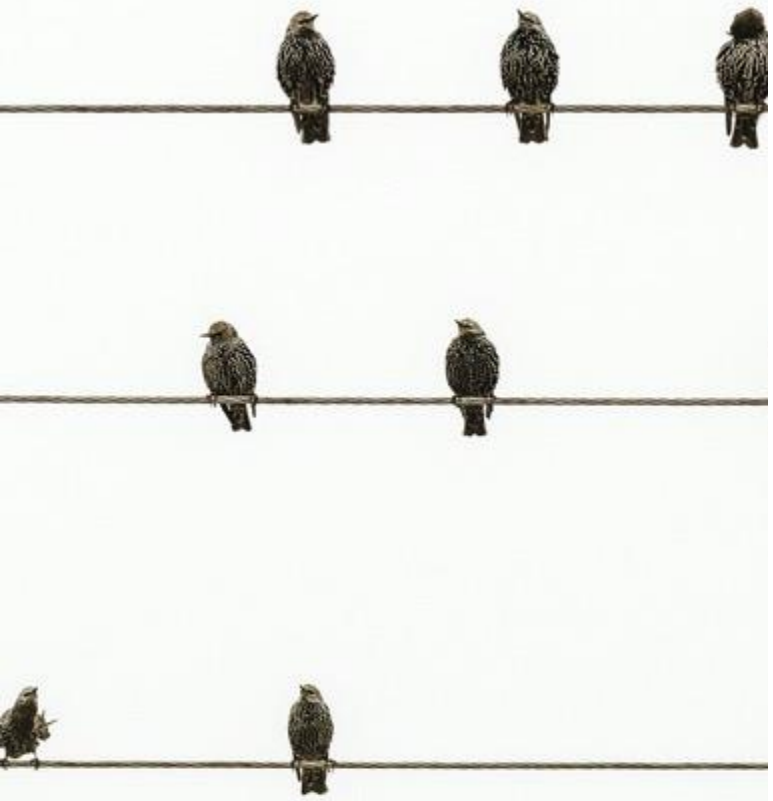


DIAGRAMA DE DISEÑO

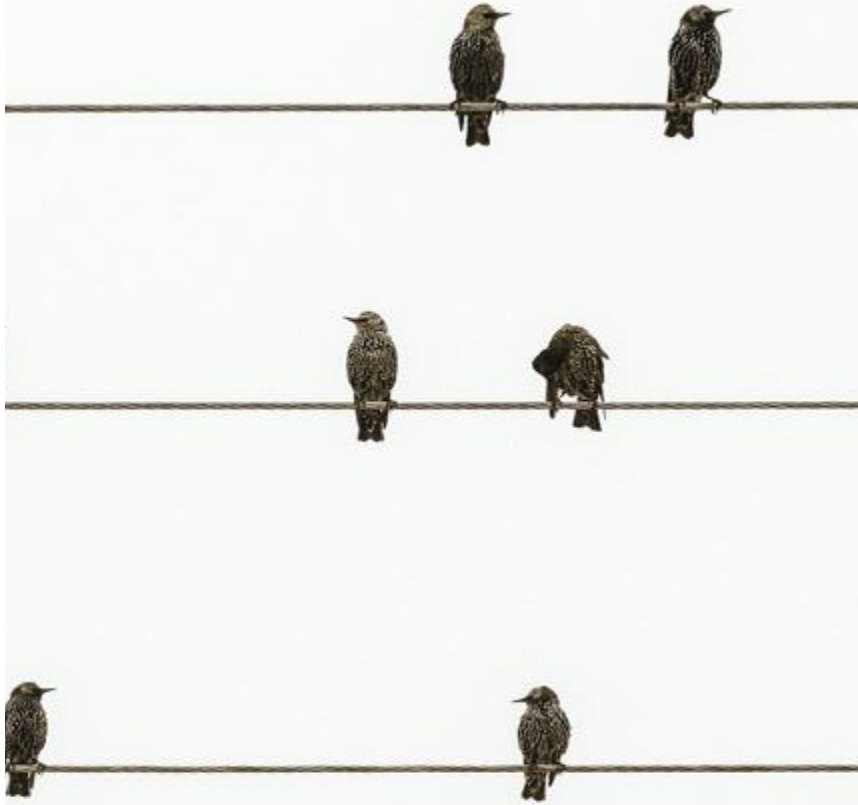
Solo las preguntas con dificultad principiante tendrán el sistema de ayudas.

A traves del tiempo transcurrido por cada pregunta, se calcula el puntaje para el ranking de usuarios.

Las preguntas con dificultad principiante tienen un tiempo de 15 segundos y las preguntas con dificultad experto tienen un tiempo de 10 segundos.



Funcionalidades y Logica



Creando una trivia

```
if difficulty_level == "beginner"
  choice_count = rand(3..6)
  true_false_count = rand(3..4)
  remaining_count = 10 - choice_count - true_false_count

  autocomplete_count = [remaining_count, 0].max

  choice_questions = difficulty.questions.where(type: 'Choice').order("RANDOM()").limit(choice_count)
  true_false_questions = difficulty.questions.where(type: 'True_False').order("RANDOM()").limit(true_false_count)
  autocomplete_questions = difficulty.questions.where(type: 'Autocomplete').order("RANDOM()").limit(autocomplete_count)

  questions = choice_questions.to_a + true_false_questions.to_a + autocomplete_questions.to_a
  shuffled_questions = questions.shuffle
  trivia.questions.concat(shuffled_questions)
```

- La cantidad de autocomplete quedan determinadas en función de las demás
- Se concatenan y se mezclan las preguntas (shuffle)

Obteniendo las respuestas

```ruby
if selected_answer.nil? && !question.is_a?(Autocomplete)
  session[:answered_questions] << index
  redirect "/question/#{index+1}"
elsif session[:answered_questions].include?(index)
  redirect '/error?code=answered'
else
  # Crear una nueva fila en la tabla QuestionAnswer con los IDs de la pregunta y la respuesta seleccionada
  session[:answered_questions] << index # Agregar el índice de la pregunta respondida a la lista
  question_answer = QuestionAnswer.find_or_initialize_by(question_id: question.id, trivia_id: @trivia.id)
  if !selected_answer.nil?
    question_answer.answer_id = selected_answer.id
    question_answer.save
    selected_answer.update(selected: true)
  end
```

- Si no se selecciono una pregunta, se marca como respondida y redirige a la próxima pregunta
- Si se intento volver hacia atrás para responder nuevamente, redirige a error
- Si pasan las validaciones, creamos una fila QuestionAnswer y guardamos la respuesta seleccionada.

Obteniendo las preguntas

```ruby
get '/question/:index' do
  redirect '/trivia' if @trivia.nil?  # Redirigir si no hay una trivia en sesión

  index = params[:index].to_i #convierte el parametro index en un entero y se guarda en la variable index
  question = @trivia.questions[index] # se obtiene la pregunta con su index y se almacena en question
  previous_index = index.zero? ? 0 : index - 1

  if index.zero? || session[:answered_questions].include?(previous_index)
    if question.nil? || index >= 10
      redirect '/results' # Redirigir a los resultados si no hay más preguntas
    else
      @question = question
      @answers = Answer.where(question_id: question.id)
      @time_limit_seconds = @trivia.difficulty.level == "beginner" ? 15 : 10
      @question_index = index # Inicializar @question_index con el valor de index
      @help = @trivia.difficulty.level == "beginner" ? question.help : nil
      erb :question, locals: { question: @question, trivia: @trivia, question_index: @question_index, answers: @answers
    end
  else
    redirect "/error?code=unanswered"
```

- Se redirige a error si la pregunta anterior no fue contestada (sin tener en cuenta la primer pregunta)
- Si el índice es valido, se prepara la información necesaria para mostrar la vista.

Calculando el score

```ruby
def calculate_response_time_score(response_time, response_time_limit)
  # Asignamos una puntuación máxima de 10 puntos a una respuesta correcta
  max_score = 10

  # Si el nivel es 'beginner', restamos 1 punto por cada 4 segundos que tomó responder la pregunta
  if response_time_limit == 15
    points_to_subtract = [(response_time / 4).ceil, 3].min
  else
    # Si el nivel no es 'beginner', restamos 1 punto por cada 3 segundos que tomó responder la pregunta
    points_to_subtract = [(response_time / 3).ceil, 3].min
  end

  # Calculamos la puntuación final restando los puntos a restar de la puntuación máxima y asegurándonos de que esté dentro del rango 0 a
  final_score = max_score - points_to_subtract
  final_score.clamp(0, max_score)
end
```

- Para nivel principiante, se resta 1 punto por cada 4 segundos de respuesta.
- Para difícil, se restan 3 por cada segundo de respuesta.

Script JS: Timer

```
function startTimer(duration, display) {
  var timerContainer = document.getElementById("timer-container");
  var timer = duration, seconds;
  var triviaDifficultyLevel = "<%= @trivia.difficulty.level%>";
  timerContainer.className = "green";
  var intervalId = setInterval(function () {
    seconds = parseInt(timer % 60, 10);
    display.text(seconds);
    timer--;
    if (triviaDifficultyLevel == "beginner") {
      if (timer >= 10){
        timerContainer.className = "green";
      } else if (timer >= 5){
        timerContainer.className = "yellow";
      } else if (timer >= 0){
        timerContainer.className = "red";
      } else {
        clearInterval(intervalId);
        showTimeUpMessage();
      }
    } else if (triviaDifficultyLevel == "difficult") {
        if (timer >= 7){
          timerContainer.className = "green";
        } else if (timer >= 4){
          timerContainer.className = "yellow";
        } else if (timer >= 0){
          timerContainer.className = "red";
        } else {
          clearInterval(intervalId);
          showTimeUpMessage();
        }
    }
  }, 1000);
}
```

- En función de la dificultad, se inicia el temporizador por cada pregunta.
- Se va actualizando el tiempo en segundos.
- Si llega a 0, se llama a otro script.

Script JS: Tiempo en 0

```
function showTimeUpMessage() {
  var overlay = document.createElement("div");
  overlay.classList.add("overlay");
  var messageContainer = document.createElement("div");
  messageContainer.classList.add("message-container");
  messageContainer.innerHTML = `
    <div class="message-box">
      <p>Se terminó el tiempo, respuesta marcada como incorrecta!</p>
      <button id="entendido-btn" class="button-style" onclick="nextQuestion()">Entendido</button>
    </div>
  `;
  overlay.appendChild(messageContainer);
  document.body.appendChild(overlay);
  var entendidoBtn = document.getElementById("entendido-btn");
  entendidoBtn.addEventListener("click", function () {
    var autocompleteInput = document.getElementById("autocomplete-input");
    if (autocompleteInput) {
      autocompleteInput.value = null;
    } else {
      document.querySelector('input[name="selected_answer"]').value = null;
    }
    overlay.remove();
    nextQuestion();
  });
}
```

- Creamos 2 elementos div (uno con superposición, y otro para contener el texto)
- Mostramos por pantalla que el tiempo terminó
- Luego, se ejecuta nextQuestion()

(nextQuestion() simplemente envía el formulario)