Guesswhich!

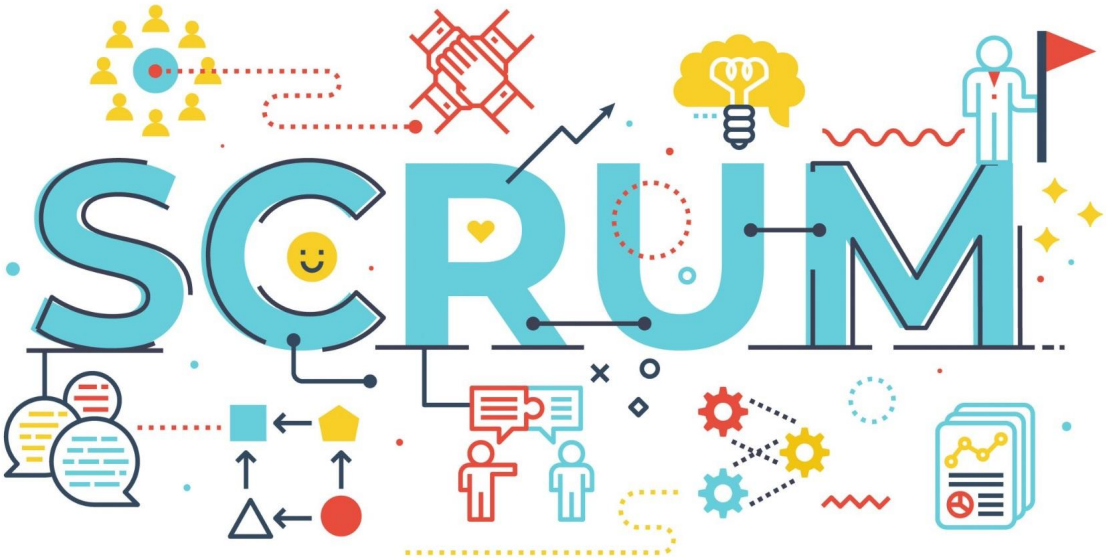
Aprende sobre la historia del mundo jugando.

Idea

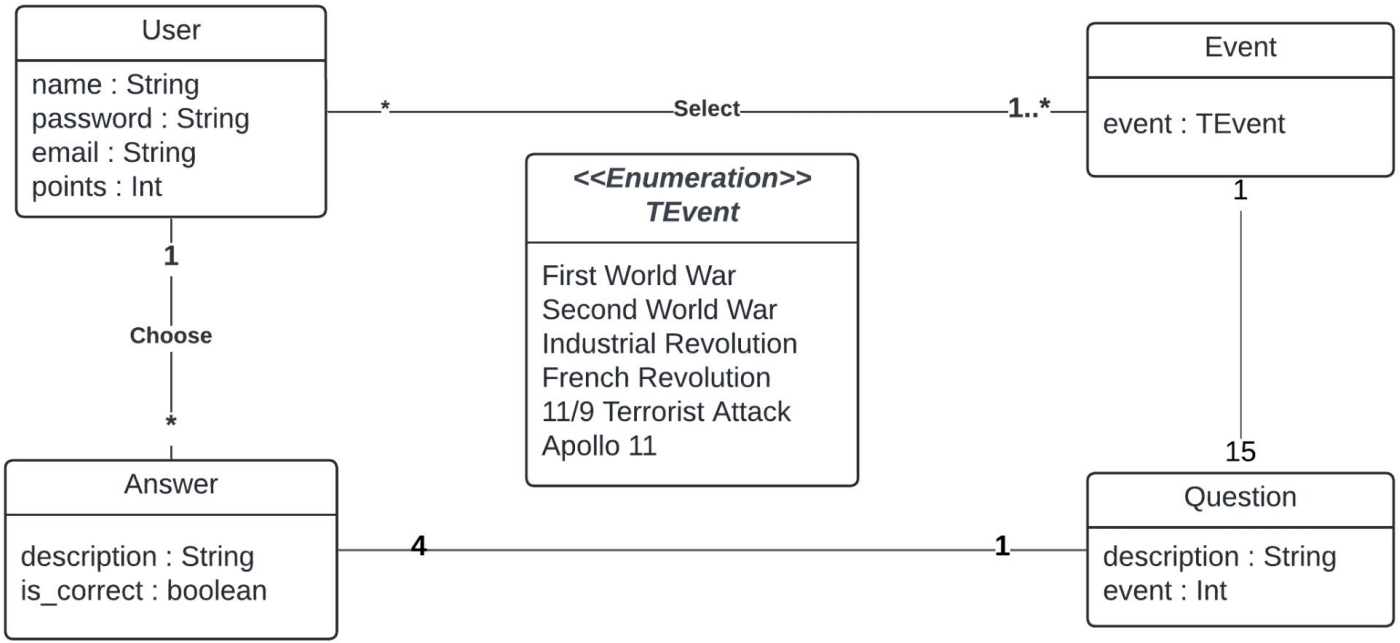
Guesswhich! es un juego de trivia diseñado para mejorar el conocimiento sobre historia de los usuarios.

- **Descripción del juego:** El juego permite a los usuarios elegir un evento histórico sobre el cual desean aprender.
- **Proceso de aprendizaje:** Una vez seleccionado el evento, el usuario estudia un breve texto informativo proporcionado por el sistema.
- **Interacción:** El usuario responde una serie de preguntas relacionadas con el evento para evaluar su comprensión.
- **Flexibilidad y expansión:** El sistema está diseñado para ser flexible, permitiendo la fácil adición de nuevos eventos históricos. Actualmente, hay 6 eventos disponibles para jugar.
- **Objetivo educativo:** El objetivo principal es proporcionar una manera divertida e interactiva de aprender sobre la historia del mundo.
- **Experiencia de usuario:** La información y las preguntas se presentan en una interfaz amigable y fácil de usar, con elementos interactivos para mantener el interés del usuario.
- **Público objetivo:** El juego está dirigido a estudiantes y entusiastas de la historia, aunque puede ser disfrutado por cualquier persona interesada en aprender más sobre eventos históricos.

Metodología de desarrollo



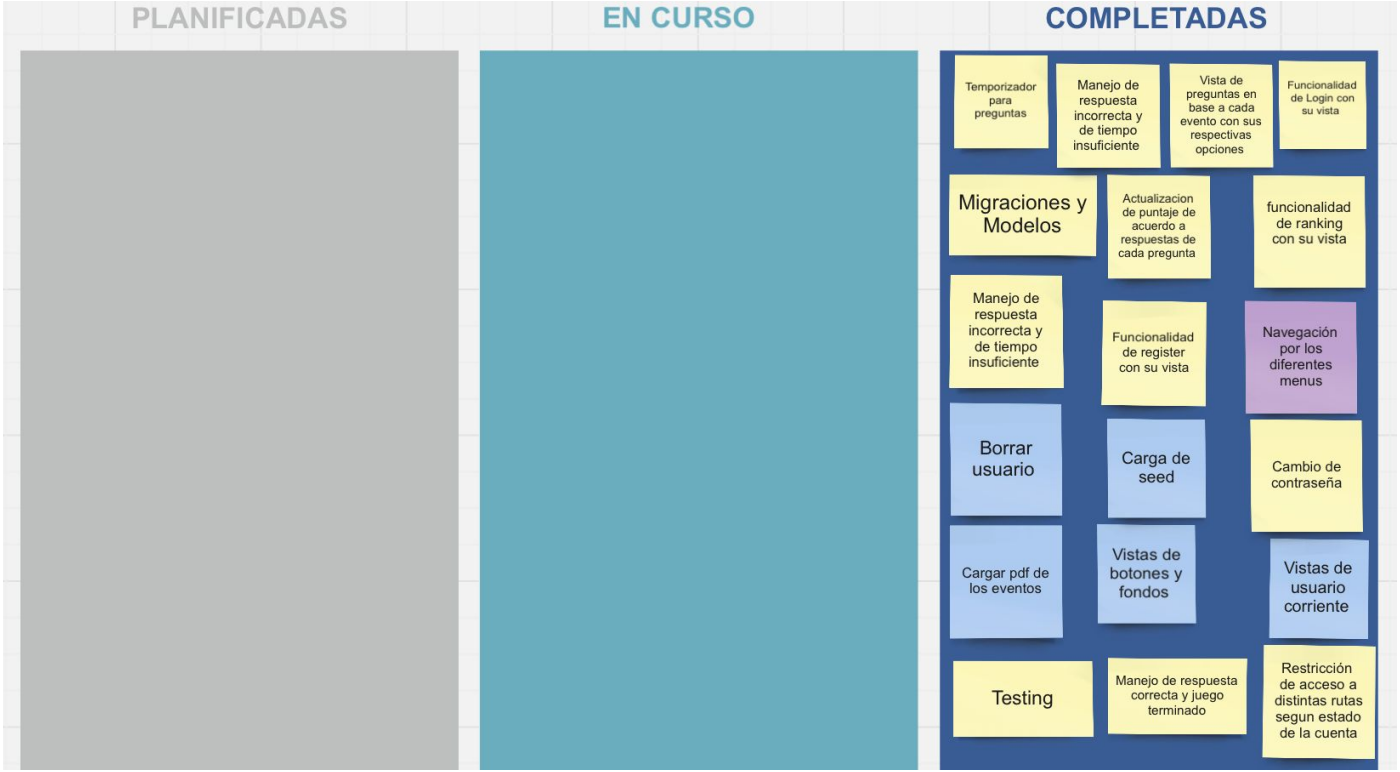
UML Class Diagram



Functional Requirements

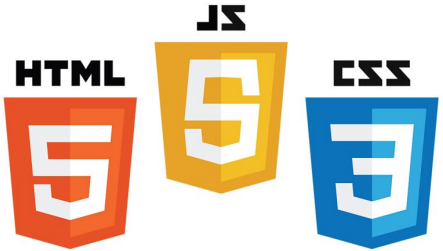
- El sistema debe permitir a los usuarios ingresar un nombre de usuario y una contraseña para autenticarse.
- El sistema debe permitir a los usuarios restablecer su contraseña.
- El sistema debe ser capaz de manejar diferentes eventos históricos.
- El sistema debe ser capaz de actualizar el estado de juego en tiempo real de acuerdo a las acciones.
- El sistema debe ser capaz de navegar por las diferentes interfaces y capaz de seleccionar cualquier opción presente.
- El sistema debe permitir salir del juego en cualquier momento.
- El sistema debe ser capaz de borrar un usuario existente.

Product Backlog



Stack tools

Ruby



Database

Migraciones: Utilizadas para definir y gestionar cambios en la estructura de la base de datos de manera controlada.
Tablas: Las tablas creadas mediante migraciones son *user*, *question* y *answer*.
Población de Datos: *question y answer* contienen datos iniciales cargados a través de un archivo *seeds.rb.*
Facilidad de Actualización: Es sencillo agregar nuevos eventos históricos. Solo se necesitan añadir las nuevas preguntas y respuestas en *seeds.rb* y hacer mínimos cambios en el servidor.

```ruby
# seeds.rb
Question.create!([{ description: '¿En qué año comenzó la Segunda Guerra Mundial?', event: 0 }])
Answer.create!([
                  { description: '1938', question: questions[0], is_correct: false },
                  { description: '1939', question: questions[0], is_correct: true },
                  { description: '1940', question: questions[0], is_correct: false },
                  { description: '1941', question: questions[0], is_correct: false }])
```

Testing

Se realizaron tests utilizando la herramienta *RSpec* para la **detección de errores** en el funcionamiento del sistema.

Ejemplo:

```
it 'logs in with valid credentials' do
 post '/login', username: 'testuser', password: 'testuserpassword'
 expect(last_response.status).to eq(302)
 follow_redirect!
 expect(last_request.path).to eq('/menu')
 expect(last_response.body).to include('User: testuser')
end
```

```
@user = User.create(
 username: 'testuser',
 password: 'testuserpassword',
 email: 'testuser@example.com',
 names: 'Test User',
 score: 0
)
```

Demo