

# Taller de Ingenieria de Software

## Actividad 1

D'Autilio Joel, Rojo Jonathan, Rossi Pablo

13 de octubre de 2022

Link del repositorio en GitHub:

<https://github.com/Jota98daut/Prode-AYDS2022-DRR>

En esta actividad, el rol de Scrum Master lo toma Joel.

## 1. Tareas

### 1.1. Agregar test integrador de crear apuesta

Corroborar que al crear una apuesta válida se impacta correctamente la base de datos. Para agregar este tipo de test hay que asegurarse de que la base de datos se encuentre limpia a la hora de ejecutar cada test.

- Tiempo destinado: 2 hs
- Responsable: Jonathan Rojo

### Implementación

Los testeos se agregan a los archivos en el directorio spec, para ser ejecutados por la herramienta rspec.

`spec/bet_spec.rb`

```
describe 'Bet' do
  describe 'when a bet is submitted' do
    describe 'and it is valid' do
      it 'should add it correctly' do
        player = Player.create(...)
        team_A = Team.create(name: 'teamA')
        team_B = Team.create(name: 'teamB')
        tournament = Tournament.create(name: 'copita')
        stage = Stage.create(name: 'stageA',
                              penalties: true, tournament: tournament)
        match = Match.new(draw: false, home: team_A, away:
```

```

        team_B, stage: stage)
    expect(match.save).to eq(true)
  end
end
end
end

```

## 1.2. Agregar test de registrar usuario

Corroborar que al registrar un usuario ocurra lo siguiente:

- Si el usuario existe, cancelar
- Si no hay contraseña, cancelar
- Si las contraseñas no coincide, cancelar
- Si el usuario no existe y las contraseñas coinciden, que impacte la base de datos
- Tiempo destinado: 2 hs
- Responsable: Pablo Rossi

### Implementación

spec/user\_spec.rb

```

describe 'User' do
  describe 'When an user is submitted passing three args' do
    describe 'and it is valid' do
      it 'should add it correctly' do
        u = User.create(...)
        expect(User.find_by(id: u.id).nil?).to eq(false)
      end
    end
  end

  describe 'and the passwords dont match' do
    it 'shouldn\'t be added' do
      u = User.create(username: 'jlennon', password: 'yesterda',
                      password_confirmation: 'yesterday')
      expect(User.find_by(id: u.id).nil?).to eq(true)
    end
  end

  describe 'and the username already exists' do
    it 'shouldn\'t be added' do
      u = User.create(username: 'jlennon', password: 'yesterday',
                      password_confirmation: 'yesterday')
    end
  end
end

```

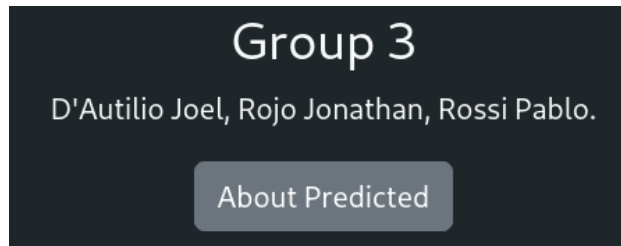


Figura 1: Botón agregado al footer

```
w = User.new(username: 'jlennon', password: 'heyjude',  
              password_confirmation: 'heyjude')  
expect(w.save).to eq(false)  
end  
end  
end  
end
```

### 1.3. Agregar información

Agregar una página con información sobre cómo jugar. En esta explicar que es Predicted, como se juega, la diferencia entre las apuestas a un partido eliminatorio y un partido común. También se podría aclarar como es el sistema de puntos.

- Tiempo destinado: 2 hs
- Responsable: Pablo Rossi
- Dependencia: Modularizar el server en distintos helpers

Los resultados se ven en las figuras 1, 2 y 3.

### 1.4. Investigar sobre flashes e implementarlos en la creación del usuario

En caso de error al crear un usuario, informar el error en la página mediante el uso de flashes. En un principio, decidimos agregar flashes en páginas públicas como '/sign up' o '/login'. Se puede pensar como una actividad a futuro, agregar flashes en todo el sistema.

- Tiempo destinado: 3 hs
- Responsable: Joel D'Autilio
- Dependencia: Modularizar el servidor en distintos helpers

El resultado se ve en la figura 4

## About Predicted

Predicted is a tournament betting site for all kinds of sports. In the lobby you can see a list of all tournaments loaded so far.

By choosing a tournament, you will be redirected to a page where you can see your current bets if you have ever bet on the chosen tournament. On this page you have the chance to add a new bet as well as you can see the accumulated points in the header of the page.

When adding a new bet you can appreciate that the matches are ordered by stages.

In the event that a stage does not allow definition by penalties, you will have the possibility to choose either the winning team or a tie.

Home	Away	Draw	Date	Time	
Qatar ●	Belgium ●	●	24-11-2022	07:00	<a href="#">Predict</a>

Otherwise, you will not only have the opportunity to choose the winner of the match but also choose whether it won on penalties or not.

Home	Away	Date	Time	Draw/Penalties	
Qatar ●	Belgium ●	22-12-2022	16:00	<input type="checkbox"/>	<a href="#">Predict</a>

Figura 2: Página con información sobre el juego (parte 1)

Once the match results have been uploaded, the points earned will be reflected in the appropriate tournament section. The point system is as follows:

Common Matches	
* Guess the winner	+1
* Hit the tie	+1
* Other case	+0
Knockout Matches	
* Guess the winner + Penalty forecast	+2
* Guess the winner only	+1
* Guess the penalty forecast only	+1
* Other case	+0

Figura 3: Página con información sobre el juego (parte 2)

## Log in

Username

Password

user doesn't exist

Log in

[Don't have an account yet?](#)

Figura 4: Error al ingresar un usuario inexistente

## 1.5. Modularizar el server en distintos helpers

Distribuir la lógica de las rutas, que se encuentra en `server.rb`, en varios archivos helpers.

- Tiempo destinado: 3 hs
- Responsable: Jonathan Rojo (se dividirán los helpers entre los 3 integrantes)

### Implementación

Se deben especificar en `server.rb` qué helpers se utilizarán

`server.rb`

```
class App < Sinatra::Application
  helpers SessionHelper
  helpers LobbyHelper
  helpers SportHelper
  helpers TournamentHelper
  helpers BetHelper
  helpers StageHelper
  helpers MatchHelper
  helpers TeamHelper
  helpers RankingHelper

  ...
end
```

Al recibir una request a alguna ruta, se usa el respectivo helper

`server.rb`

```
...
get '/tournaments' do
  get_tournaments
end
...
```

La lógica se encuentra en otro archivo

`helpers/tournament_helper.rb`

```
module TournamentHelper
  def get_tournaments
    @tournaments = Tournament.all
    erb : 'tournaments/index'
  end
end
```

```
...  
end
```

## 1.6. Evitar la creación de partidos entre dos equipos iguales

Averiguar cómo agregar una validación al modelo de match para no permitir un partido entre el mismo equipo, e implementarlo.

- Tiempo destinado: 20 min
- Responsable: Joel D'Autilio

### Implementación

Se agrega la validación en el modelo de Match

models/match.rb

```
class Match < ActiveRecord::Base  
  has_many :bets  
  belongs_to :home, class_name: 'Team'  
  belongs_to :away, class_name: 'Team'  
  belongs_to :winner, class_name: 'Team'  
  belongs_to :stage  
  
  validates :home, presence: true  
  validates :away, presence: true  
  validates :stage, presence: true  
  validates :home, comparison: { other_than: :away }  
  ...  
end
```

## 1.7. Agregar estilo a los flashes y agregar flash en la creación de un partido

Una vez agregados los flashes, buscaremos darle un formato para que tenga una mejor visualización. Además, con el objetivo de hacer el sistema un poco más amigable, se agregaran flashes en la sección 'match/new'

- Tiempo destinado: 45 min
- Responsable: Joel D'Autilio
- Dependencia: Investigar sobre flashes e implementarlos en la creación de un usuario

### Resultado

El resultado se ve en la figura 5

### Add match

Stage: 
 Date: 
 Time: 
 Home: 
 Away:

teams must be different

Figura 5: Error al agregar un partido con dos equipos iguales

## 2. Diagrama de Gantt

La planificación temporal se realizó usando la página bitrix24.es que provee funcionalidad para administrar tareas y mostrarlas en un diagrama de Gantt.

El diagrama resultante se ve en las figuras 6, 7 y 8.

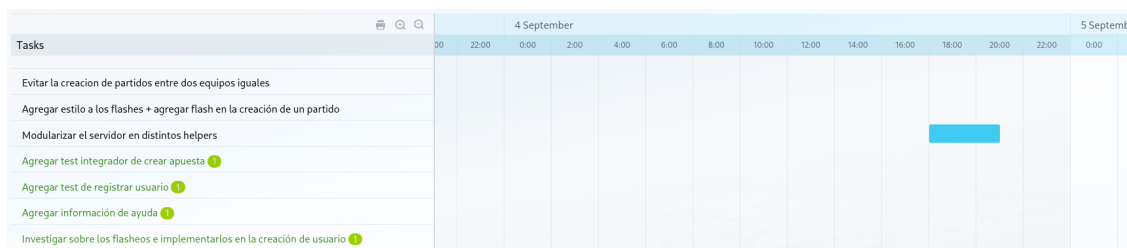


Figura 6: Diagrama de gantt (parte1)

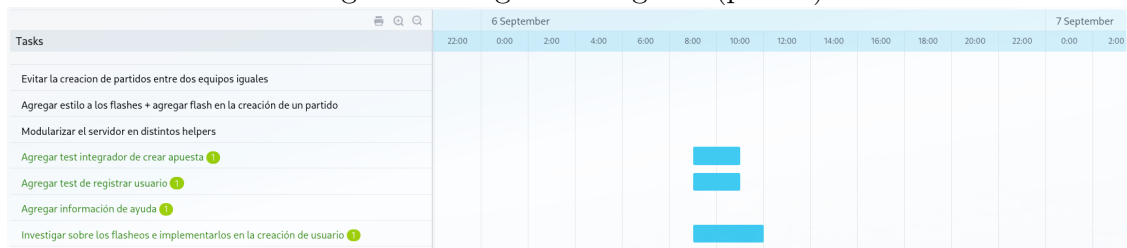


Figura 7: Diagrama de gantt (parte2)

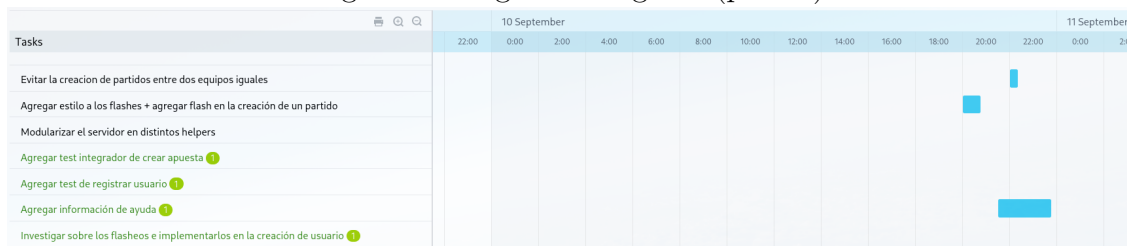


Figura 8: Diagrama de gantt (parte3)