

# RubyTrick

## Producto

### Grupo 3 - Proyecto de ingeniería de software

Instituto de computación  
Facultad de ingeniería  
Universidad de la República

2 de diciembre de 2014



# ¿Qué es RubyTrick?

RubyTrick es:

- Un aplicación web. ([rubytrick.herokuapp.com](http://rubytrick.herokuapp.com)).
- Un juego de fútbol multijugador masivo en línea.
- Una herramienta educativa para el aprendizaje de programación (puntualmente en el lenguaje Ruby).

Surge inspirado en **RubyGoal**, desarrollada por **WyeWorks** y presentada en RubyConf 2014.

RubyGoal consiste en un simulador 2D de un partido de fútbol, donde los usuarios pueden escribir la estrategia de cada equipo en lenguaje Ruby.

La idea de **Moove-It** es llevar un juego similar a un nivel más masivo a través de la web. La manifestación de esta idea es RubyTrick.

Las principales funcionalidades requeridas son:

- Un simulador de partidos a partir de estrategias escritas en Ruby.
- Una especificación de cómo escribir las estrategias.
- Una interfaz web para que los usuarios puedan jugar.
- Un sistema de información para el manejo de la actividad de los usuarios (partidos jugados, estrategias escritas e información adicional).

Los usuarios deben poder jugar partidos contra otros usuarios utilizando sus estrategias.

Extendiendo la temática del juego:

- Creación de campeonatos.
- Manejo del equipo (mejorar habilidades de los jugadores, comprar y vender jugadores, paga de salarios).

Adicionalmente:

- Relacionamiento entre usuarios (amistad, chat).
- Apuestas.
- Ranking.
- Integración con redes sociales (Twitter, Facebook).
- Panel de administración.

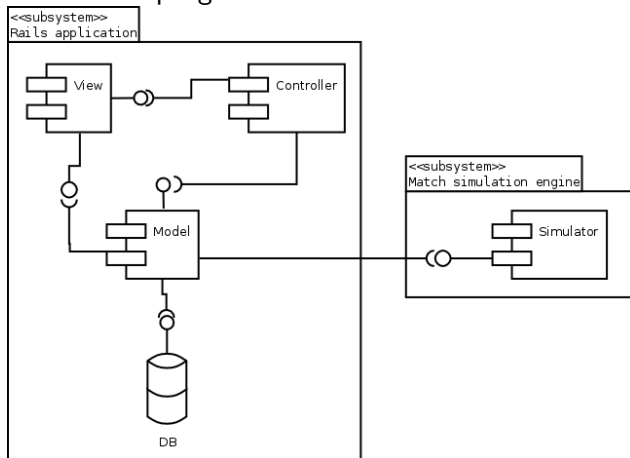
# Requisitos no funcionales

No se determinaron requisitos no funcionales.

Sí se habló de que era un objetivo “retener a los usuarios.”

Se desarrolló el objetivo en historias y en diseño de UI.

## Vista de despliegue:



- Servidor central en *Ruby On Rails*.
- Por lo tanto, sigue el patrón arquitectónico *Model-View-Controller*
- y el patrón *ActiveRecord* para el modelo.
- Simulador de partidos como módulo separado, escrito en Ruby puro.



El diseño de clases del modelo se realizó colectivamente.

Se realizó incrementalmente, por *sprint*.

Al inicio de cada *sprint* una versión inicial del diagrama de clases fue presentada por el arquitecto.

En caso de ser necesario, los desarrolladores realizaban modificaciones.

Originalmente se pensó un diseño basándose en los criterios GRASP.

Se hacía uso de una capa de controladores de casos de uso, procurando separar la capa lógica (modelo) de la presentación (controladores).

Esto se revirtió a pedido de los especialistas técnicos, a favor de trabajar de la manera sugerida por el *framework*.

Los controladores pasaron a tener más relevancia en la lógica.

- Modelo y controladores: desarrollados heredando de las clases del *framework* (en Ruby).
- Vista: escritas en HAML y JavaScript.
- Hojas de estilo en SCSS (Sass).

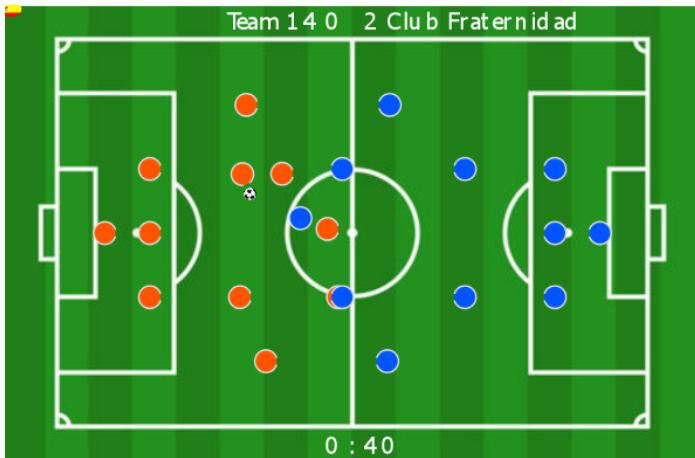
# Simulación de partidos

Simulador completamente escrito en Ruby.

Gema ejecutable. Ejecuta fuera del contexto del servidor.

- Entrada: dos estrategias escritas en Ruby y dos equipos (conjuntos de jugadores).
- Salida: Objeto JSON representación del partido.

# Simulación de partidos



# Simulación de partidos

La cancha se modela como una cuadrícula de  $5 \times 5$ .

Cada jugador tiene una posición fijada por la estrategia.

El jugador más cercano a la pelota de cada equipo se desplaza hacia ella.

Cuando llega, la pasa al compañero de adelante más cercano.

Si es el último jugador patea al arco.

Determinan la posición de cada jugador en cada instante.

Deben escribirse en Ruby, implementando dos métodos:

- `get_initial_formation`: formación al iniciar el partido.
- `get_formation`: determina la formación en cada instante.  
Recibe como parámetros el marcador y las posiciones actuales de ambos equipos.

Ambos deben devolver una matriz de  $5 \times 5$  representando la posición de cada jugador en la cancha.

En caso contrario el equipo pierde.

Se desarrollaron y aceptaron todas las historias recabadas.  
La última iteración se utilizó para corregir según el *feedback* recibido y los *bugs* reportados hasta el momento.  
Sin embargo, se detectaron nuevos *bugs* que quedaron pendientes.  
Se puede acceder a la aplicación en [rubytrick.herokuapp.com](http://rubytrick.herokuapp.com).



Algunos aspectos positivos de la aplicación resultante:

- Diseño simple.
- Reusabilidad del simulador.
- Estrategias en Ruby (no DSL).

Algunas debilidades:

- Seguridad.
- Robustez.
- UI.

Posibles mejoras:

- Nuevas modalidades de juego.
- Mayor realismo en la simulación.