Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2143 - Ingeniería de Software 2024-2

# Tarea Introducción a *Rails*

### Objetivos

- Aprender a utilizar el *framework Ruby on Rails*.
- Entender e implementar el concepto de *API*.

### Fecha de Entrega

30 de Agosto del 2024, 23:59

## Gamers review API

En esta tarea usted tendrá que crear una API para registrar, crear, actualizar, y borrar videojuegos, reviews y jugadores que pertenecen al mundo gamer. Además, su API debe permitir algunas consultas relacionando distintos modelos.

**Tips:**

- Realizar la tarea con tiempo.
- Crear el proyecto con: `rails new GamersApi --database=postgresql --api`
- Manejar los DELETE que afecten a las reviews mediante eliminado en cascada.
- Cuando se indica que cierto modelo debe tener cierto atributo, se espera que su solución verifique que esto ocurra y no permita la presencia de vacíos o nulos

# Modelos mínimos

En esta sección se detallarán los modelos mínimos que debe manejar la *API*, junto a sus atributos. Todo lo señalado a continuación, deberá ser implementado utilizando los **mismos nombres**, ya que de esta forma los atributos en los JSON de respuesta tendrán los nombres que se pide para cada endpoint. En caso de no utilizar los nombres mencionados, la corrección fallará, ya que utilizamos automatización de pruebas para revisar los modelos.

## Model Game

- **Atributos:**
  - `name` de tipo `string`
  - `calification` de tipo `float`
  - `description` de tipo `string`
- **Asociaciones:** Un juego puede tener varias reviews.
- **Consideraciones:**
  - Para simplificar los sistemas de búsqueda, los nombres de juegos tendrán un `_` para simbolizar los espacios.
  - El atributo `calification` sera un float que ira del `1.0` al `10.0`.
  - Debe tener el juego un nombre, calificacion y descripcion

## Model Review

- **Atributos:**
  - `title` de tipo `string`
  - `description` de tipo `string`
  - `calification` de tipo `float`
  - `player_id` de tipo `references`
  - `game_id` de tipo `references`
- **Asociaciones:** Una review pertenece exactamente a solo un juego y pertenece también a exactamente un jugador.
- **Consideraciones:**
  - Es importante recalcar que un review debe tener un titulo, descripcion, calificacion, id de jugador e id de juego.

## Model Player

- **Atributos:**
  - `name` de tipo `string`
  - `preference` de tipo `string`
  - `phone` de tipo `string`

  - `favourite_game_id` de tipo `references`

- **Asociaciones:** Cada jugador puede poseer multiples review y solo un juego favorito.
- **Aclaración:** En caso de que el juego favorito sea borrado este debe quedar como `null` en el modelo de `player`.
- **Consideraciones:**
  - Es importante recalcar que un jugador debe tener un nombre, preferencia y numero. Pero puede no tener un juego favorito

## Rutas solicitadas

Las rutas solicitadas se dividen en 3 partes, donde cada una contiene los endpoints respectivos. Cada parte se evaluara usando tests que verificaran el funcionamiento correcto de los distintos endpoints. El puntaje final sera calculado en base a la proporción de pruebas aprobadas en cada parte.

### Parte 1 (Easy): 2 ptos

1. `GET /games`

   Debe retornar todos los juegos que se encuentren en la base de datos.

   Ejemplo `response body`:

```
[
    {
        "id": 4,
        "name": "Elden_ring",
        "calification": 5.4,
        "description": "Elden Ring combina la jugabilidad desafiante y
            gratificante de FromSoftware con una narrativa profunda y un mundo
            expansivo, creando una experiencia inmersiva y unica." ,
        "created_at": "2024-03-07T14:03:40.827Z",
        "updated_at": "2024-03-07T14:03:40.827Z"
    },
    {
        "id": 5,
        "name": "fifa_2011",
        "calification": 3.0,
        "description": "Lanzado en 2010, es un videojuego de futbol desarrollado
            por EA Sports. Forma parte de la serie anual de videojuegos de la FIFA
             y esta disponible en multiples plataformas, incluidas PlayStation "3"
            , Xbox "360", PC, y mas."
        "created_at": "2024-03-07T14:03:41.827Z",
        "updated_at": "2024-03-07T14:03:41.827Z"
    }
]
```

2. `POST /games`

   Deberá crear un nuevo juego en base a los atributos enviados.

   Ejemplo `request body`:

```
{
    "game": {
        "name": "Call_of_Duty:_Black_Ops_2",
        "calification": 10.0,
        "description": "Call of Duty: Black Ops II es conocido por su innovadora
            historia con multiples finales, su robusto multijugador y su popular
            modo Zombis, lo que lo convierte en una de las entregas mas destacadas
             de la serie Call of Duty."
    }
```

```
}
```

Ejemplo `response body`:

```
{
    "id": 6,
    "name": "Call_of_Duty:_Black_Ops_2",
    "calification": 10.0,
    "description": "Call of Duty: Black Ops II es conocido por su innovadora
        historia con multiples finales, su robusto multijugador y su popular modo
        Zombis, lo que lo convierte en una de las entregas mas destacadas de la
        serie Call of Duty.",
    "created_at": "2024-03-07T14:05:21.827Z",
    "updated_at": "2024-03-07T14:05:21.827Z"
}
```

3. `GET /games/:id`

   Debe retornar al juego cuya `id` se entrega en la url.

   Ejemplo url: `http://localhost:3000/games/5`

   Ejemplo `response body`:

```
    {
        "id": 5,
        "name": "fifa_2011",
        "calification": 3.0,
        "description": "Lanzado en 2010, es un videojuego de futbol desarrollado
            por EA Sports. Forma parte de la serie anual de videojuegos de la FIFA
             y esta disponible en multiples plataformas, incluidas PlayStation "3"
            , Xbox "360", PC, y mas."
        "created_at": "2024-03-07T14:03:41.827Z",
        "updated_at": "2024-03-07T14:03:41.827Z"
    }
```

4. `DELETE /games/:id`

   Deberá eliminar el juego cuya `id` coincida con la entregada en la url.

   Ejemplo url: http://localhost:3000/games/5

   Ejemplo response body:

```
{}
```

5. `DELETE /games`

   Deberá eliminar todos los juegos.

   NOTA: Este será el primer endpoint que se ejecutara en todos los tests.

   Ejemplo url: http://localhost:3000/games

   Ejemplo response body:

```
[]
```

## Parte 2 (Medium): 3 ptos

1. `POST /players/:game_id`

   Deberá crear un nuevo jugador cuyo juego favorito sea el id entregado `:game_id`.

   Ejemplo url: http://localhost:3000/players/4

   Ejemplo request body:

```
{
    "player": {
        "name": "Benja1234",
        "preference": "indie",
        "phone": "+56966782343"
    }
}
```

   Ejemplo response body:

```
{
    "id": 1,
    "name": "Benja1234",
    "preference": "indie",
    "phone": "+56966782343",
    "favourite_game_id": 4,
    "created_at": "2023-08-17T21:31:17.521Z",
    "updated_at": "2023-08-17T21:31:17.521Z"
}
```

2. `DELETE /players`

   Deberá eliminar todos los jugadores.

   NOTA: Este será el segundo endpoint que se ejecutara en todos los tests de tipo medium y hard.

   Ejemplo url: http://localhost:3000/players

   Ejemplo response body:

```
[]
```

3. `GET /players/:player_id`

   Deberá entregar el jugador de id player_id.

   Ejemplo url: http://localhost:3000/players/1

   Ejemplo response body:

```
[
    {
    "id": 1,
    "name": "Benja1234",
    "preference": "indie",
    "phone": "+56966782343",
    "favourite_game_id": 4,
    "created_at": "2023-08-17T21:31:17.521Z",
```

```
    "updated_at": "2023-08-17T21:31:17.521Z"
}
]
```

4. `GET /games/:game_id/reviews`

   Deberá entregar las reseñas del juego solicitado.

   Ejemplo url: http://localhost:3000/games/5/reviews

   Ejemplo response body:

```
[
    {
        "id": 2,
        "title": "Grandes mejoras respecto a la entrega anterior",
        "description": "FIFA 11 es una entrega solida de EA Sports, con el motor "
            Personality+" que mejora la autenticidad de los jugadores y el sistema
             "Pro Passing" que agrega precision a los pases. El modo Carrera
            expandido y las mejoras graficas ofrecen una experiencia de juego
            inmersiva y realista.",
        "calificacion": 3.0,
        "player_id": 1,
        "game_id": 5,
        "created_at": "2024-03-07T14:12:04.800Z",
        "updated_at": "2024-03-07T14:12:04.800Z"
    },
    {
        "id": 3,
        "title": "no me gusto el juego",
        "description": "el juego fue lo mismo de siempre esperaba mejoras respecto
             al juego anterior, ademas que esperaba mayor enfoque en el mundial",
        "calification": 6.0
        "player_id": 5,
        "game_id": 5,
        "created_at": "2024-03-07T14:15:44.603Z",
        "updated_at": "2024-03-07T14:15:44.603Z"
    }
]
```

5. `POST /reviews`

   Deberá crear una nueva reseña, notar que esta debe estar asociado a el jugador y juego indicados.

   Ejemplo request body:

```
{
    "review":{
        "title": "odio este juego",
        "description": "no puede ser que no hayan creado algo con graficos tan
            vagos",
        "calification": 4.7,
        "player_id": 5,
```

```
        "game_id": 4
    }
}
```

Ejemplo response body:

```
{
    "id": 4,
    "title": "odio este juego",
        "description": "no puede ser que no hayan creado algo con graficos tan
            vagos",
    "calification": 2.0,
    "player_id": 5,
    "game_id": 4,
    "created_at": "2024-03-07T14:15:48.243Z",
    "updated_at": "2024-03-07T14:15:48.243Z"
}
```

6. `PATCH /reviews/:review_id`

   Deberá editar la reseña en base de los nuevos atributos entregados en el body.

   Ejemplo url: localhost:3000/reviews/4

   Ejemplo request body:

```
{
    "review": {
        "title": "No es tan malo"
    }
}
```

   Ejemplo response body:

```
{
    "id": 4,
    "title": "No es tan malo",
        "description": "no puede ser que no hayan creado algo con graficos tan
            vagos",
    "calification": 2.0
    "player_id": 5,
    "game_id": 4,
    "created_at": "2024-03-07T14:15:48.243Z",
    "updated_at": "2024-03-07T14:15:48.243Z"
}
```

7. `GET /reviews/:game`

   Deberá retornar todos las reseñas del juego cuyo nombre es el `:game` solicitado

   Ejemplo url: localhost:3000/reviews/Elden_ring

   Notar que Elden ring es el juego de id 4 en este ejemplo, por lo que se retornan aquellos reviews con game 4

   Ejemplo response body:

```
[
{
    "id": 4,
    "title": "No es tan malo",
        "description": "no puede ser que no hayan creado algo con graficos tan
            vagos",
    "calification": 2.0
    "player_id": 5,
    "game_id": 4,
    "created_at": "2024-03-07T14:15:48.243Z",
    "updated_at": "2024-03-07T14:15:48.243Z"
}
]
```

## Parte 3 (Hard): 1 pto

1. `GET /players/:player_id/game`

   Deberá entregar el juego favorito del jugador solicitado.

   Ejemplo response body: Igual a Parte 1, Ruta 3

2. `GET /games/top/:quantity`

   Deberá entregar los mejores `quantity` juegos considerando sus calificaciones.
   En caso de empate, es decir, en caso de que dos juegos tengan la misma calificación, será mejor el que tenga mayor `id`.

   **Nota:** `:quantity` será un numero natural.

3. `GET /reviews/:review_id/player`

   Para la review del id entregado en `:review_id`, entregar el jugador asignado.
   Ejemplo url:http://localhost:3000/reviews/1/player

   Ejemplo response body:

```
{
    "id": 1,
    "name": "Benja1234",
    "preference": "indie",
    "phone": "+56966782343",
    "favourite_game_id": 4,
    "created_at": "2023-08-17T21:31:17.521Z",
    "updated_at": "2023-08-17T21:31:17.521Z"
}
```

4. `DELETE /games/low`

   Deberá entregar y luego eliminar a el juego con la menor calificación.
   En caso de empate, es decir, si existen dos juegos con la misma calificacion, será peor el que tenga menor id.

   Ejemplo response body: Igual a Parte 1, Ruta 3

5. `PATCH /games/calification_update/:game_id`

   Deberá actualizar el valor de `calification` que tenga el juego cuyo id sea :game_id tal que su nuevo valor sea el promedio de las calificaciones recibidas por ese juego en reviews.

   Ejemplo url: games/calification_update/4

   Ejemplo response body, suponiendo que el promedio de la calificación en las reviews recibidas es un 7.0:

```
{
    "id": 4,
    "title": "elden ring",
    "calification": 7.0,
    "description": "Elden Ring combina la jugabilidad desafiante y gratificante de
        FromSoftware con una narrativa profunda y un mundo expansivo, creando una
        experiencia inmersiva y unica.",
    "created_at": "2024-03-07T14:15:48.243Z",
```

```
    "updated_at": "2024-03-07T14:15:48.243Z"
}
```

## Evaluación

Para evaluar que su *API* funciona correctamente, se utilizará un script de Python para realizar consultas sobre su API. Cada test tendrá puntaje y dificultad asociado y este indicará si Pasa (1) o No Pasa (0), por su parte, cada test consistirá en varios llamados a distintos endpoints. El puntaje total corresponderá a la suma de todos los tests. Se les facilitará un script dentro de los siguientes dias con test públicos para poder testear sus tareas, para la corrección se utilizarán tests secretos que serán publicados una vez se entreguen las notas.

## Entrega

La entrega de esta tarea deberá realizarse en un repositorio de GitHub que se les facilitara, es importante que su código se encuentre en la rama *main*, de lo contrario la tarea no se revisara.
Es importante que el código de su entrega corresponda a las carpetas que conforman su proyecto de RoR sin estar estar contenidas dentro de otra carpeta del tipo GamersAPI. El no cumplir con este aspecto supondrá un descuento de 5 décimas a la nota obtenida por el alumno. De manera similar, la entrega de código ejecutable que presente errores o aspectos harcodeados en su archivo .env, usen una versión de ruby diferente a la del curso o cualquier otro motivo que dificulte la corrección de su tarea, serán penalizados con 5 décimas a su nota final por cada motivo de descuento
Es responsabilidad del alumno preocuparse de aceptar la invitación a su repositorio respectivo. No se aceptaran entregas fuera del repositorio entregado ni fuera de plazo.

#### Ruta de ejemplo

`https://github.com/IIC2143/2024-2-Tarea-UsuarioDeGithub`

## Política de integridad académica

Los alumnos de la Escuela de Ingeniería de la Pontificia Universidad Católica de Chile deben mantener un comportamiento acorde a la Declaración de Principios de la Universidad. En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un Procedimiento Sumario. Es responsabilidad de cada alumno conocer y respetar el documento sobre Integridad Académica publicado por la Dirección de Docencia de la Escuela de Ingeniería en el SIDING. Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica. Todo trabajo presentado por un alumno para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno, sin apoyo en material de terceros. Por “trabajo” se entiende en general las interrogaciones escritas, las tareas de programación u otras, los trabajos de laboratorio, los proyectos, el examen, entre otros. Si un alumno copia un trabajo, obtendrá nota final 1.1 en el curso y se solicitará a la Dirección de Pregrado de la Escuela de Ingeniería que no le permita retirar el curso de la carga académica semestral. Por “copia” se entiende incluir en el trabajo presentado como propio partes hechas por otra persona. Obviamente, está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la referencia correspondiente. Lo anterior se entiende como complemento al Reglamento del Alumno de la Pontificia Universidad Católica de Chile. Por ello, es posible pedir a la Universidad la aplicación de sanciones adicionales especificadas en dicho reglamento.