



Universidad del Bío-Bío  
Facultad de Ciencias Empresariales  
Dept. de Sistemas de Información

## Desarrollo de una aplicación web para la comunidad de Re-Volt America

Proyecto de título para optar al título de Ingeniero de Ejecución en  
Computación e Informática

**Alumno**  
José Benavente

**Profesora Guía**  
Alejandra Segura Navarrete

A handwritten signature in black ink that reads "Alejandra Segura". The signature is fluid and cursive, with "Alejandra" on the top line and "Segura" on the bottom line.

Lunes 18 de diciembre, 2023

# Índice

Dedicatoria . . . . .	7
Agradecimientos . . . . .	8
Resumen . . . . .	9
Introducción . . . . .	10
<b>1. Estudio del Problema</b>	<b>11</b>
1.1. Definiciones, Siglas y Abreviaciones . . . . .	11
1.2. Historia de Re-Volt y sus Comunidades . . . . .	12
1.3. La Comunidad de Re-Volt America . . . . .	14
1.4. Contexto del Problema . . . . .	15
1.4.1. Re-Volt y Partidas en Línea . . . . .	15
1.4.1.1. Autos y Categorías . . . . .	16
1.4.1.2. Sala de Espera . . . . .	17
1.4.1.3. Carreras en Línea . . . . .	17
1.4.2. Re-Volt: OpenGL y los Session Logs . . . . .	19
1.4.3. Funcionamiento Interno de Re-Volt America . . . . .	20
1.4.3.1. Sistema de Temporadas, Rankings y Sesiones . . . . .	20
1.4.3.2. Paquete de Contenido de RVA . . . . .	21
1.4.3.3. Sistema de Puntuación . . . . .	21
1.4.3.4. Cálculo de Resultados . . . . .	23
1.4.3.5. Multiplicadores por Auto . . . . .	25
1.5. Sistema Actual . . . . .	26
1.6. Problemática Actual . . . . .	26
1.6.1. Diagrama de la Situación en la Actualidad . . . . .	27
1.7. Propuesta de solución . . . . .	28
1.8. Soluciones similares disponibles . . . . .	28
1.8.1. Aplicación para Cálculo de Puntos de Re-Volt I/O . . . . .	28
1.9. Justificación del Problema . . . . .	30
<b>2. Proyecto</b>	<b>31</b>
2.1. Objetivo General del Proyecto . . . . .	31
2.2. Objetivos Específicos del Proyecto . . . . .	31
2.3. Metodología de Desarrollo . . . . .	31
2.4. Técnicas y Notaciones . . . . .	33
2.5. Estándares de Documentación . . . . .	33
2.6. Software, Frameworks y Lenguajes Utilizados . . . . .	33

<b>3. Factibilidad</b>	<b>35</b>
3.1. Factibilidad Técnica . . . . .	35
3.1.1. Conocimientos de los Usuarios . . . . .	35
3.1.2. Disponibilidad Profesional . . . . .	35
3.1.3. Despliegue y Servidor . . . . .	36
3.2. Factibilidad Operativa . . . . .	36
3.3. Factibilidad Económica . . . . .	37
3.3.1. Tablas de Costos . . . . .	37
3.3.2. Flujo de Caja . . . . .	38
3.3.2.1. Contexto e Indicadores Económicos . . . . .	38
3.3.2.2. Puesta en Marcha . . . . .	38
3.3.2.3. Cálculo del Valor Actual Neto . . . . .	39
3.4. Conclusión de Factibilidad . . . . .	40
<b>4. Arquitectura de Re-Volt America</b>	<b>41</b>
4.1. Organización de GitHub . . . . .	41
4.2. Repositorios del Paquete de Contenido de RVA . . . . .	43
4.3. Repositorio de RVA-Data . . . . .	46
4.4. Servidor de Distribución . . . . .	47
4.5. Despliegue e Integración Continua . . . . .	48
<b>5. Requerimientos del Software</b>	<b>50</b>
5.1. Límites . . . . .	50
5.2. Caracterización de los Usuarios . . . . .	50
5.3. Objetivo General del Software . . . . .	51
5.3.1. Objetivos Específicos del Software . . . . .	51
5.4. Requerimientos Funcionales del Software . . . . .	52
5.5. Requerimientos No Funcionales del Software . . . . .	54
5.6. Interfaces Internas de Salida . . . . .	58
5.7. Interfaces Externas de Salida . . . . .	59
<b>6. Análisis Funcional</b>	<b>60</b>
6.1. Actores . . . . .	60
6.2. Casos de Uso . . . . .	62
6.2.1. Diagramas de Casos de Uso . . . . .	62
6.2.2. Especificación de los Casos de Uso . . . . .	67
6.2.3. Detalle de los Casos de Uso . . . . .	69
6.3. Modelo de Datos . . . . .	78
6.4. Esquema de la Base de Datos . . . . .	82
6.5. Diseño de Interfaz . . . . .	93
6.5.1. Paleta de Colores y Tipografía . . . . .	96
6.6. Diseño de Arquitectura . . . . .	101
6.7. Estructura del Código . . . . .	102
6.7.1. Estándares de Codificación . . . . .	103
6.7.2. Caché con Redis . . . . .	103

6.7.3. Backend . . . . .	104
6.7.4. Frontend . . . . .	104
<b>7. Plan de Capacitación, Implantación y Puesta en Marcha</b>	<b>105</b>
7.0.1. Estado del Proyecto . . . . .	105
7.0.2. Implantación y Puesta en Marcha . . . . .	105
7.0.3. Plan de Capacitación . . . . .	105
<b>8. Conclusión del Proyecto</b>	<b>107</b>
<b>9. Anexos</b>	<b>109</b>
9.1. Anexo Estimación de Casos de Uso . . . . .	109
9.2. Anexos de Recopilación de Información . . . . .	110
9.3. Anexo Aspectos de Gestión de Proyectos . . . . .	111
9.3.1. Anexo Carta Gantt . . . . .	111
9.3.2. Anexo Resumen de Esfuerzo . . . . .	111
9.4. Anexos Retrospectiva del Proyecto . . . . .	112
9.4.1. Anexo Iteraciones en el Desarrollo . . . . .	112

# Índice de figuras

1.1.	Caratula de Re-Volt, 1999 . . . . .	12
1.2.	Logotipo de Re-Volt: I/O . . . . .	13
1.3.	Logotipo de Re-Volt America . . . . .	14
1.4.	Selección de nombre de usuario en RVGL . . . . .	15
1.5.	Selección de auto en RVGL . . . . .	16
1.6.	Sala de espera en RVGL . . . . .	17
1.7.	Punto de vista del jugador en carrera . . . . .	18
1.8.	Resultados de carrera en RVGL . . . . .	18
1.9.	Session Log de RVGL . . . . .	19
1.10.	Tabla de resultados de RVA . . . . .	24
1.11.	Diagrama BPMN del proceso actual de cálculo de puntos . . . . .	28
1.12.	Software de cálculo de resultados de Re-Volt: I/O . . . . .	29
4.1.	Árbol de directorios de rva_pack . . . . .	43
6.1.	Diagrama de Casos de Uso del módulo de privilegios . . . . .	62
6.2.	Diagrama de Casos de Uso del módulo de autos 1 . . . . .	63
6.3.	Diagrama de Casos de Uso del módulo de autos 2 . . . . .	63
6.4.	Diagrama de Casos de Uso del módulo de pistas 1 . . . . .	64
6.5.	Diagrama de Casos de Uso del módulo de pistas 2 . . . . .	64
6.6.	Diagrama de Casos de Uso del módulo de temporadas 1 . . . . .	65
6.7.	Diagrama de Casos de Uso del módulo de temporadas 2 . . . . .	65
6.8.	Diagrama de Casos de Uso del módulo de sesiones 1 . . . . .	66
6.9.	Diagrama de Casos de Uso del módulo de sesiones 1 . . . . .	66
6.10.	Resumen de modelo de datos . . . . .	78
6.11.	Modelo de relaciones entre usuario, perfil y estadísticas . . . . .	79
6.12.	Modelo de relación entre temporada, autos y pistas . . . . .	79
6.13.	Modelo de relaciones entre temporada, ranking y sesiones con sus componentes . . . . .	80
6.14.	Modelo de relación entre sesión, carrera y entrada de corredor . . . . .	81
6.15.	Landing page de RVA . . . . .	93
6.16.	Página de autos de RVA . . . . .	94
6.17.	Página de pistas de RVA . . . . .	94
6.18.	Página de sesión de RVA . . . . .	95
6.19.	Página de perfil de usuario de RVA . . . . .	95
6.20.	Página de estadísticas globales de RVA . . . . .	96

6.21. Logotipo de RVA . . . . .	97
6.22. Arte de Castcraft Software . . . . .	97
6.23. Tabla de resultados de sesión de RVA . . . . .	98
6.24. Tabla de ranking de RVA . . . . .	99
6.25. Paleta de colores de sesiones de RVA . . . . .	100
6.26. Diagrama de arquitectura de RVA . . . . .	101
6.27. Árbol de directorios del proyecto . . . . .	102
7.1. Video de capacitación para administradores . . . . .	106
9.1. Carta Gantt del proyecto . . . . .	111

# Índice de cuadros

# Dedicatoria

El presente proyecto está dedicado a toda la comunidad de Re-Volt, en especial a quienes forman parte del grupo de jugadores de Re-Volt America.

Sin lugar a dudas, durante los últimos años, este juego pasó de ser un simple pasatiempo a convertirse en algo muy importante para mí a nivel personal. Muchas de las personas que he conocido a través de Re-Volt se han convertido, ya a estas alturas, en buenos amigos a quienes valoro muchísimo. Todos han sabido siempre apreciar mi trabajo, y por eso les estoy infinitamente agradecido. Sin ustedes, nada de lo que amo hacer tendría trascendencia alguna. De todo corazón, muchas gracias. Es por todo lo anterior que no podría dedicar este trabajo a nadie si no a ustedes.

open source ubb elizabeth carrera todos los profes bilder

# Agradecimientos

En primer lugar, se extiende un agradecimiento formal a los siguientes desarrolladores quienes, de una forma u otra, han contribuido a la base de código del presente proyecto:

- Marco Roth. Por ayudar con la migración del proyecto a "esbuild", y por resolver problemas con HAML.
- Nicolás Duque. Por probar la instalación del proyecto en plataformas de Linux, testear el proyecto en su fase beta, y por ayudar con la implementación de traducciones.
- Luigi Riccio.
- Esteban Martinez
- Henrique Gomes Britos

También se les extiende un agradecimiento a todos quienes han hecho una donación voluntaria, sin importar el monto que fuese, al desarrollador principal de este proyecto a través de GitHub Sponsors durante el desarrollo de esta tesis:

- Vicente Aguilera.
- Gabriel Carnielli.
- Dario Chaile.
- Benjamín Contreras.
- Benjamín Ferrada.
- Josafat Jiménez.
- Mateusz Kobylański.
- Jorge Matamala.
- Benjamín Mosso.
- Leandro Rodríguez.
- Juan Pablo Rosas.

# Resumen

Dentro de las comunidades del videojuego de carreras Re-Volt, 1999, Re-Volt America se ha quedado atrás en lo que refiere a su sistema interno de cálculo de resultados y estadísticas por jugador. Esta problemática genera el surgimiento de este proyecto, el cual pretende modernizar y mejorar la gestión interna de Re-Volt America, así como también busca enriquecer la experiencia de usuario como nunca antes se ha visto en la escena de las comunidades de Re-Volt.

El presente informe trata de una aplicación que busca centralizar toda la información relacionada a las sesiones multijugador del videojuego Re-Volt celebradas por la comunidad de Re-Volt America, logrando así mejorar la experiencia de usuario para los administradores encargados de mantener los registros de resultados y tablas de puntuación, además de servir como un cambio revolucionario para todos aquellos que buscan conseguir una experiencia competitiva dentro del videojuego.

# Introducción

El presente informe contiene las especificaciones técnicas correspondientes al desarrollo del proyecto de titulación de la carrera de Ingeniería de ejecución Informática titulado "Desarrollo de una aplicación web para la comunidad de Re-Volt America".

El documento se organiza en varios capítulos. Primeramente, el capítulo 1 trata del estudio del problema, que es donde se habla de la historia del videojuego Re-Volt y sus comunidades, así como también del contexto del problema al que se enfrenta Re-Volt America hoy en día. Luego, en el capítulo 2 se definen los objetivos generales y específicos del proyecto a desarrollar, además de la metodología de desarrollo elegida.

Más adelante, en el capítulo 3 de factibilidad, podrá encontrarse una descomposición de la factibilidad del proyecto en tres pilares fundamentales: factibilidad técnica, factibilidad operativa, y factibilidad económica. Este capítulo termina entregando una conclusión de factibilidad.

El capítulo 4 contiene todo lo que tiene que ver con la arquitectura de Re-Volt America a nivel técnico. Desde su lógica de negocio hasta su sistema de despliegue en la actualidad.

En el capítulo 5, se definirán los límites del software, la caracterización de sus usuarios, los objetivos tanto generales como específicos, los requerimientos del software y las interfaces.

En el capítulo 6 se realiza un análisis funcional a todo el proyecto ya desarrollado, evaluando sus actores, casos de uso, modelos de datos, esquema de base de datos, diseño de interfaz, arquitectura y estructura del código.

En el capítulo 7, se habla del plan de capacitación, implantación y puesta en marcha del proyecto.

Finalmente, el capítulo 8 entrega una conclusión del proyecto, y el capítulo 9 adjunta todos los anexos relevantes.

# Capítulo 1

## Estudio del Problema

### 1.1. Definiciones, Siglas y Abreviaciones

Como aclaración inicial, **Re-Volt America** es un nombre propio de comunidad, el cual se escribe sin acentuar la letra "e" como vendría normalmente acentuada en el idioma español. El anglicismo de "America" es completamente intencional, ya que el nombre proviene del idioma inglés americano.

A continuación, se definirán algunos conceptos relevantes en el contexto del videojuego Re-Volt y de la comunidad de Re-Volt America:

- Re-Volt: El videojuego Re-Volt, publicado por Acclaim Studios en 1999.
- RV: Abreviación para "Re-Volt".
- Re-Volt: OpenGL: Reescritura moderna de Re-Volt, basada en OpenGL.
- RVGL: Abreviación de "Re-Volt: OpenGL".
- Re-Volt I/O: Comunidad europea de Re-Volt.
- Re-Volt America: La comunidad americana de Re-Volt.
- RVA: Abreviación para "Re-Volt America".
- Sesión: Serie de carreras de RVGL en línea, en donde dos o más personas compiten en carreras multijugador.
- Session Log: Archivo separado por comas, el cual contiene un registro crudo de los resultados de las carreras jugadas en una sesión de RVGL.
- Host/Anfitrión: El jugador que abre una partida online de Re-Volt para que otros jugadores puedan conectarse a él.
- Staff: Jugadores con el rol de administrador, organizador o moderador dentro de la comunidad de RVA.

## 1.2. Historia de Re-Volt y sus Comunidades

Re-Volt America, en su expresión más simple, es una comunidad de jugadores del videojuego Re-Volt, el cual fue lanzado originalmente en el año 1999 por Acclaim Studios en Londres. Re-Volt es un videojuego de carreras y simulador arcade de autos a control remoto, el cual explora una premisa en donde dichos autos compiten en carreras de radio control en ambientes como museos, supermercados, barcos, sitios de construcción, entre otros. Esto combinado con una mecánica de objetos que pueden ser recogidos por dichos autos para atacar a los competidores, obtener más velocidad, entre otras ventajas.

El arte original de la caratula del juego puede ser apreciado en la figura 1.1.



Figura 1.1: Caratula de Re-Volt, 1999

El primero de septiembre del año 2004, Acclaim Studios se declara en banca rota, y cesa permanentemente todo el desarrollo y mantenimiento que en algún momento proveyó a Re-Volt y a su comunidad. Este suceso, a lo largo de los años, dio lugar a muchas comunidades segmentadas del juego en el internet de ese entonces. Con el tiempo, nuevos sitios y proyectos comenzaron a surgir, tales como el portal web de Re-Volt Race, una página de Re-Volt que se dedicaba a organizar partidas online y mantener tablas de resultados para los jugadores, o Re-Volt: OpenGL (RVGL), una re-escritura moderna del Re-Volt original que todos conocían, ahora disponible para plataformas modernas y otros sistemas operativos además de Windows, como Linux,

MacOS, e incluso una versión para dispositivos Android.

Dentro de lo anteriormente enmarcado, aparece en el año 2015 la comunidad de Re-Volt I/O, cuyo logotipo se puede apreciar en la figura 1.2. Esta comunidad estaba formada por un grupo de jugadores de Re-Volt, principalmente europeos, quienes incursionaron por primera vez en intentar crear una plataforma estable para el videojuego y su comunidad de jugadores. Este sería un lugar en donde cualquiera que quisiera disfrutar del juego podría encontrar guías de ayuda, tutoriales, descargas y demás contenido para poder instalar y jugar Re-Volt en su computador o dispositivo móvil.



Figura 1.2: Logotipo de Re-Volt: I/O

En sus inicios, Re-Volt I/O adoptó a RVGL como la distribución estándar de Re-Volt que ofrecería a sus jugadores, haciéndole ganar público y reconocimiento al proyecto publicando enlaces de descarga directos en su página web ([re-volt.io](http://re-volt.io)), además de entregar soporte y mantener hilos de discusión relacionados con RVGL y sus actualizaciones en su foro oficial ([forum.re-volt.io](http://forum.re-volt.io)).

En adición a lo anterior, RVGL no era tan sólo una versión modernizada del Re-Volt original, sino que también traía consigo el aspecto más importante que tiene Re-Volt en la actualidad, y el cual mantiene unida y activa a su comunidad en general: el modo multijugador u online. Dicho modo no sólo permitía a los jugadores correr carreras en línea, sino que, además, extendía soporte para que miembros de la comunidad pudiesen diseñar sus propios autos y pistas de manera personalizada, agrandando así, de manera casi infinita, el repertorio de contenido descargable para Re-Volt.

Re-Volt I/O adoptó un sistema en donde su administración elige ciertos autos y pistas hechos por la comunidad cada ciertos meses. De esta forma, todos estos autos y pistas, elegidos a votación, terminan juntos en un paquete de contenido de extensión para RVGL, el cual Re-Volt I/O se encarga de distribuir para que sus usuarios lo descarguen y puedan jugar en línea. De manera habitual, tener este paquete de contenido es obligatorio para poder jugar en las sesiones multijugador organizadas por Re-Volt I/O, lo cual lo convertiría en un estándar para los jugadores que quisieran incorporarse a la comunidad en toda su extensión.

Fue así como Re-Volt I/O, entre finales del 2015 y mediados del 2017, logró consolidarse y llegar a más jugadores que nunca, formando una comunidad activa de amantes del juego quienes, espontáneamente, se reunían a jugar en línea durante la semana utilizando un paquete de contenido adicional para RVGL, el cual todos debían descargar e instalar por separado para poder jugar. Eventualmente, estas partidas en línea adquirieron un horario definido con fechas y horas acordadas con antelación, para así facilitar la asistencia de los jugadores a los eventos de carreras.

En la actualidad, Re-Volt I/O sigue siendo la comunidad de Re-Volt más grande en términos de jugadores y escala, pero en si todas las comunidades de Re-Volt están unidas y se ayudan unas con otras. Después de todo, se trata de un juego nicho, en donde todos intentan hacerlo accesible y fácil de entender para quienes deseen formar parte de su comunidad.

### 1.3. La Comunidad de Re-Volt America

Si bien Re-Volt I/O fue, durante muchos años, la única comunidad grande de Re-Volt a nivel mundial, no fue mucho después de su gran auge que comenzarían a formarse los demás grupos que, a día de hoy, tienen gran relevancia en la escena multijugador de Re-Volt y que, además, cuentan con un numeroso público y gran actividad. Dentro de estas nuevas comunidades se encuentra Re-Volt America, la comunidad de Re-Volt que abarca a todos los jugadores del continente americano, especialmente de latinoamérica. El logotipo oficial de Re-Volt America, o RVA para abreviar, puede apreciarse en la figura 1.1.



Figura 1.3: Logotipo de Re-Volt America

La comunidad de Re-Volt America es concebida originalmente en el año 2017, bajo el nombre de Re-Volt Tournament. No fue hasta después de un par de años que esta sería renombrada a Re-Volt America, debido a la procedencia de sus jugadores, la cual era tanto de norte america como de sudamerica.

En el presente año 2023, Re-Volt America cuenta con una gran cantidad de jugadores activos, y con un sistema de puntuación único en la escena de Re-Volt y sus comunidades en línea. Este complejo sistema de puntuación, y su funcionamiento sostenido durante los últimos 6 años, son la base del problema que busca solucionar este proyecto de título. Con el pasar del tiempo, este sistema se ha convertido en algo muy difícil de mantener para los administradores de la comunidad, tanto a nivel logístico como técnico.

## 1.4. Contexto del Problema

### 1.4.1. Re-Volt y Partidas en Línea

Como ya se mencionó anteriormente, Re-Volt es un videojuego de carreras el cual, gracias al surgimiento de RVGL y sus comunidades impulsoras, es jugado mayoritariamente en línea. Pero, ¿a qué nos referimos con "jugar en línea"? Para poder entender este concepto, tenemos que ir a lo que es una carrera en términos conceptuales, y las implicaciones que estas conllevan dentro de un contexto competitivo.

Para poder jugar en línea, cada jugador debe elegir un nombre de usuario, el cual puede incluso variar de partida en partida. Esto se hace una vez que ingresa al juego y avanza en el menú hasta llegar al selector de nombre de usuario en forma de neumático. El nombre que el jugador ingrese aquí será el nombre de usuario con el que se identificará a la hora de ser ingresado a los resultados de cada carrera en la que participe. El selector de nombre de usuario se puede apreciar a continuación en la figura 1.4.



Figura 1.4: Selección de nombre de usuario en RVGL

#### 1.4.1.1. Autos y Categorías

En Re-Volt, existen diferentes tipos de autos que pueden ser elegidos por el jugador. En el juego original, estos autos se clasifican en categorías según su velocidad máxima, aceleración, peso, y desempeño general en pista. Las categorías originales, ordenadas desde los autos más lentos, hasta los más rápidos, son las siguientes:

- Rookie (Novato).
- Amateur (Amateur).
- Advanced (Avanzado).
- Semi-Pro (Semi-Profesional).
- Pro (Profesional).
- Clockwork.

Además de las categorías originales, también existen categorías especiales que han surgido a partir del contenido creado por la comunidad de Re-Volt y RVGL. Estas categorías son las siguientes:

- Super-Pro

La categoría o "Rating" de cada auto puede apreciarse desde el menú del juego, tal como se puede ver en la figura figura 1.5.



Figura 1.5: Selección de auto en RVGL

De manera habitual, el anfitrión define una categoría de auto con la cual se jugará la sesión. De esta forma, todos los jugadores que se conecten deberán utilizar autos de la categoría definida.

Teniendo en cuenta lo anterior, las sesiones suelen ser nombradas a partir de su categoría asociada. Por ejemplo, si para una sesión se decide jugar autos de la categoría "Pro", es normal que esta sea titulada "Pros Session", o "Pro Races" al momento de ser anunciada.

#### 1.4.1.2. Sala de Espera

Una vez que el usuario elige su auto, este es llevado a la sala de espera, la cual viene representada por la figura figura 1.6



Figura 1.6: Sala de espera en RVGL

Una vez llegados a este punto, sólo se debe esperar a que el anfitrión de la sesión de inicio a las carreras.

#### 1.4.1.3. Carreras en Línea

Normalmente, en las partidas multijugador, suelen jugarse muchas carreras de manera consecutiva. A estas series de partidas en línea se les conoce como "sesiones". Cada sesión de Re-Volt consiste en una cantidad predefinida de carreras en pistas determinadas y con cierta clase de autos. Estas determinaciones las realiza el anfitrión de la partida, quien las comunica públicamente de manera oportuna para que todos aquellos que deseen participar tengan en cuenta todas las características de la sesión que van a jugar.

Las siguientes dos ilustraciones presentan las instancias clave dentro del juego. En la figura 1.7, se puede apreciar la perspectiva del jugador al momento de jugar Re-Volt. Luego, en la figura 1.8, se puede ver la tabla de resultados que se muestra por pantalla a medida que los corredores finalizan la carrera.



Figura 1.7: Punto de vista del jugador en carrera



Figura 1.8: Resultados de carrera en RVGL

### 1.4.2. Re-Volt: OpenGL y los Session Logs

Para ayudar a llevar una cuenta fiable de todas las carreras jugadas, RVGL introduce una funcionalidad que permite a los jugadores obtener un registro escrito de los resultados de cada carrera. Este registro viene en forma de un archivo separado por comas, el cual es conocido por la comunidad como Session Log. Este es el archivo que utilizan los organizadores y administradores de RVA para calcular los resultados oficiales de las sesiones.

En la figura 1.9, presentada a continuación, puede apreciarse un extracto del Session Log de una sesión cualquiera, el cual ha sido importado desde Microsoft Excel para una visualización más clara. Este extracto representa una sola de las carreras jugadas en la sesión. Entiéndase que, inmediatamente después de la última línea, vendría escrita la carrera siguiente, y así sucesivamente con todas las demás.

	A	B	C	D	E	F	G
1	Version	RVGL 23.1030a1	P2P	Client			
2	Session	12/02/2023 16:05	INS	Arcade	3	TRUE	
3	Results	School's Out! 1 R		11			
4	#	Player	Car	Time	BestLap	Finished	Cheating
5	1	B	Genghis Kar	0.090741	0.032639	TRUE	FALSE
6	2	ROD	Naranja Turbo	0.096991	0.036088	TRUE	FALSE
7	3	KILABARUS	HSF-1	0.101331	0.039039	TRUE	FALSE
8	4	WATYNECC	Genghis Kar	0.098819	0.036053	TRUE	FALSE
9	5	URV	Condor GRV	0.099676	0.040891	TRUE	FALSE
10	6	SWIMPY	Tesla	0.101146	0.036169	TRUE	FALSE
11	7	INS	Condor GRV	0.101678	0.041389	TRUE	FALSE
12	8	INIGO	Albatross GT	0.104792	0.034352	TRUE	FALSE
13	9	BALESZ	RC Phink	02:23:026	0.036829	TRUE	FALSE
14	10	NARU	Ciagnik	0.110694	0.041366	TRUE	FALSE
15	11	ANTONIUS	Volken Turbo	0.114005	0.03463	TRUE	FALSE

Figura 1.9: Session Log de RVGL

A continuación, la tabla TABLENUM es una especificación de los elementos relevantes de un Session Log. Algunos de dichos elementos son utilizados por RVA para calcular los resultados acumulados de las sesiones, para así obtener una tabla con posiciones finales para cada jugador al final de cada sesión.

Elementos de un Session Log	
<b>Id</b>	<b>Descripción</b>
<b>B1</b>	Versión de RVGL de la sesión jugada.
<b>C1-D1</b>	Protocolo de conexión de la sesión.
<b>B2</b>	Fecha y hora de apertura de la sesión.
<b>C2</b>	Nombre del jugador anfitrión de la sesión.
<b>D2</b>	Tipo de colisión entre autos, seleccionado por el anfitrión.
<b>E2</b>	Número de vueltas de esta carrera.
<b>B3</b>	Nombre de la pista de esta carrera.
<b>C3</b>	Número de jugadores que comenzaron la carrera. Si un jugador se desconecta en mitad de la carrera, este número no se ve afectado.
<b>B5-B15</b>	Nombres de los jugadores que participaron de la carrera, agregados de arriba hacia abajo por orden de llegada a la línea de meta.
<b>C5-C15</b>	Nombres de los autos utilizados por cada jugador en esta carrera.
<b>D5-D15</b>	Tiempo total de la carrera de cada jugador.
<b>E5-E15</b>	Mejor vuelta de cada jugador.
<b>F5-F15</b>	TRUE si el jugador finalizó la carrera. FALSE si el jugador se desconectó en mitad de la carrera.
<b>G5-G15</b>	TRUE si los archivos locales del jugador no corresponden a los archivos del anfitrión. FALSE si estos coinciden. Por ejemplo, "Cheating" es TRUE si un jugador modificó localmente la velocidad de su auto.

### 1.4.3. Funcionamiento Interno de Re-Volt America

#### 1.4.3.1. Sistema de Temporadas, Rankings y Sesiones

Históricamente, Re-Volt America se ha dedicado a organizar sesiones multijugador de Re-Volt para su público, así como también se ha preocupado de llevar la cuenta de los resultados de cada carrera que se ha jugado en ellas. Esto lo hace mediante un sistema de temporadas.

Todos los jugadores que, en algún momento u otro, han participado en las sesiones online organizadas por Re-Volt America, han sido indexados en la base de datos de jugadores que mantiene la comunidad. De esta forma, sus victorias, puntos y otras estadísticas asociadas han sido preservadas a lo largo del tiempo.

Para conseguir un sistema atractivo para sus jugadores, Re-Volt America organiza sus sesiones en una serie de rankings, los cuales consisten en 28 sesiones multijugador cada uno, jugándose una sesión al día. Al cumplirse 6 de estos rankings, se completa lo que en RVA se conoce como una temporada.

Cada sesión organizada por RVA consiste en 20 carreras, las cuales se juegan en 20 pistas diferentes. Es por esto que, en una sesión, se producen 20 sets de resultados (1 por carrera).

Cada año, en promedio, se inician dos temporadas, y cada una es nombrada en base a su año de inicio, y su periodo en relación a otras temporadas. Por ejemplo, si en el año 2023 se da inicio a una temporada en el mes de enero, esta vendría a llamarse "2023", ya que terminará dentro del mismo año en el que inició. Por otra parte, si la temporada

inicia en el mes de diciembre de 2023, esto quiere decir que terminará dentro del año 2024, por lo que pasaría a llamarse "2023-24".

Es así como Re-Volt America, desde el año 2017, hasta el presente, ha conseguido consolidarse como la comunidad de Re-Volt predilecta para los jugadores tanto de norteamérica como América latina.

#### **1.4.3.2. Paquete de Contenido de RVA**

Re-Volt America cuenta con un paquete de contenido extra para RVGL, el cual es construido por sus administradores y organizadores al principio de cada temporada.

Este paquete consiste en una selección de autos y pistas para RVGL hechos por la comunidad, los cuales tienen el objetivo de extender el contenido original del juego, y así proveer a los usuarios con una experiencia más enriquecedora y variada al momento de jugar en línea.

El paquete de contenido de RVA, más conocido como RVA Pack, es, técnicamente, un conjunto de archivos correspondientes a los autos y pistas que este busca agregar. Estos archivos son mandatorios para poder participar de las sesiones organizadas por RVA. Esto quiere decir que, para que un usuario pueda entrar a una sesión de RVA, este debe tener instalado el RVA Pack en su instalación de RVGL local.

#### **1.4.3.3. Sistema de Puntuación**

Como se ha aludido a lo largo del estudio del problema, la comunidad de Re-Volt America cuenta con su propio sistema de puntuación, el cual le sirve para generar estadísticas por jugador, tablas de resultados al final de cada sesión, y demás métricas que, finalmente, promueven la competitividad y enriquecen la experiencia de los usuarios.

El funcionamiento de dicho sistema de puntuación es el siguiente: dependiendo de la posición de cada jugador, al final de cada carrera, a estos se les asigna un puntaje. El mapeo de puntos para una carrera con menos de 10 participantes viene definido por la siguiente función que relaciona posiciones con puntos, respectivamente:

$$f_1 = \begin{cases} pos & f_1(pos) \\ 1 \rightarrow 15 \\ 2 \rightarrow 12 \\ 3 \rightarrow 10 \\ 4 \rightarrow 7 \\ 5 \rightarrow 5 \\ 6 \rightarrow 4 \\ 7 \rightarrow 2 \\ 8 \rightarrow 2 \\ 9 \rightarrow 1 \end{cases}$$

Por otro lado, en caso de que la carrera cuente con 10 jugadores o más, el mapeo de puntos sería el siguiente:

$$f_2 = \begin{cases} pos & f_2(pos) \\ 1 \rightarrow 20 \\ 2 \rightarrow 16 \\ 3 \rightarrow 12 \\ 4 \rightarrow 10 \\ 5 \rightarrow 8 \\ 6 \rightarrow 8 \\ 7 \rightarrow 6 \\ 8 \rightarrow 4 \\ 9 \rightarrow 2 \\ 10 \rightarrow 2 \\ 11 \rightarrow 1 \\ 12 \rightarrow 1 \\ 13 \rightarrow 1 \\ 14 \rightarrow 1 \\ 15 \rightarrow 1 \\ 16 \rightarrow 1 \end{cases}$$

Por lo tanto, el conjunto que define los posibles puntajes obtenibles viene definido de la siguiente manera:

$$P = \{1, 2, 4, 5, 6, 7, 8, 10, 12, 15, 16, 20\}$$

Si se tiene que " $n$ " es la cantidad de jugadores que participaron de una carrera, entonces:

$$A = \{1, \dots, n\} \subset \mathbb{N}$$

$$n = |A|$$

Teniendo en cuenta lo anterior, es posible definir una función que, al recibir una posición final de un jugador en una carrera determinada, retorne la cantidad de puntos correspondientes a dicha posición. Esta función vendría definida a continuación, en donde " $pos$ " es la posición final del jugador en la carrera en cuestión.

$$f : pos \in A \rightarrow \begin{cases} f_1(pos) & \text{if } n < 10 \\ f_2(pos) & \text{if } n \geq 10 \end{cases} \in P$$

Al finalizar una sesión, los puntajes de cada jugador en cada carrera se suman, lo que entrega un total de puntos por jugador. Una vez sumados, los puntajes finales son normalizados, y sometidos a diferentes procesos de ajuste propios del sistema de RVA. Una vez ajustados, estos pasan a ser los puntajes finales de la sesión de cada jugador.

#### 1.4.3.4. Cálculo de Resultados

Una vez procesados, los resultados son llevados a una tabla final, tal como se puede apreciar en la figura 1.10.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC		
1	# Fecha/Session	PF	jugador	MEX	FG2	S02	DEL	THI	S01	CT2	SCI	WS2	MD	SM2	FG1	PV	GEX	TGS	TSZ	RT	MM	M3	CT	PP	PA	CC	MP	PO			
122	5	17/02/2023	1	BGM	1	3	1	4	3	3	1	4	1	1	2	3	2	1	1	1	1	2	5	2.11	215	19	0.95	9.68			
123					<i>Rebound 4X4</i>																										
124			2	TOMDOM	3	2	4	5	2	7	4	3	4	2	1	7	4	6	5	4	2	3	5	2	3.75	139	20	1	3.71		
125					<i>B59</i>																										
126			3	ROC_97	1	3	7	1	4	2	1	2	7	7	5	5	2	4	3	3	4	1		3.44	140	18	0.9	3.66			
127					<i>Albatross GT</i>																										
128			4	OSVALDOX	6	4	6	2	3	8	8	4	5	6	1	5	3	6	5	5	2	4		4.61	143	18	0.9	2.79			
129					<i>Manko XS</i>																										
130			5	LOQ	5	5	8	3	10	9	8	7	7	8	6	2	8	3	6	7	6	4	9		6.37	112	19	0.95	1.67		
131					<i>Panorama Gunior</i>																										
132			6	WALUGI MB	7	6	7	1	4	1	7	9	3	9	3	1	6	7								5.07	108	14	0.7	1.49	
133					<i>Trirraut Get Air</i>																										
134			7	ESTEBANMZ	2	8	5	5	5	6	9							2	2	4	1	1	8		4.46	84	13	0.65	1.22		
135					<i>Panorama Col Moss</i>																										
136			8	ASURA	4	8	6	2	7	8	6	2	10	3	4	8	7	8	7	5	6	3	6		5.79	73	19	0.95	1.20		
137					<i>Hot Spl. MoSprinter XL</i>																										
138			9	ALE OWO	2	7	5	6	8	6		5	6	5												5.56	37	9	0.45	0.30	
139					<i>RC Bandit</i>																										
140			10	TIOROTTI											5	6	8	4	3	4	8						5.43	28	7	0.35	0.18
141					<i>Sprinter XL</i>																										
142			11	BOMB FACTORY	8	9	9	9	9	9	10																9.00	7	6	0.3	0.02
143					<i>Sprinter XE1 Gekko Volken Turbo</i>																										
144			12	BURDANG																							3.00	7	1	0.05	0.01
145			13	PHON7X																											
146					<i>Dr. Gru</i>																										
147					<i>Volken T</i>																										

Figura 1.10: Tabla de resultados de RVA

Parámetros			
Nombre	Abreviación	Descripción	
Posición Promedio	PP	La suma de las posiciones del jugador, dividida por su cantidad de carreras corridas.	
Puntaje Acumulado	PA	La suma total de los puntos correspondientes a las posiciones obtenidas por el jugador.	
Carreras Corridas	CC	La cantidad de carreras corridas por el jugador.	
Multiplicador por Participación	MP	La cantidad de carreras corridas por el jugador, dividida por la cantidad de carreras totales de la sesión.	
Puntaje Oficial	PO	El puntaje acumulado del jugador, multiplicado por un factor normalizador de 0.1.	

De acuerdo con la tabla anterior, los cálculos de PA, PP y MP pueden ser representados utilizando las siguientes fórmulas, respectivamente, en donde "pos" es la posición del jugador en cada carrera, y "tot" la cantidad total de carreras de la sesión.

$$PA = \sum_{i=1}^{CC} x = x + f(pos_i)$$

$$PP = \frac{\left( \sum_{i=1}^{CC} x = x + pos_i \right)}{CC}$$

$$MP = \frac{CC}{tot}$$

A partir de estos parámetros, podemos calcular el puntaje oficial, o PO, obtenido por el jugador en la sesión en cuestión.

$$PO = \left( \frac{PA}{PP} \cdot MP \right) \cdot 0.1$$

#### 1.4.3.5. Multiplicadores por Auto

Como ya se mencionó anteriormente, al principio de cada temporada, se agregan y se eliminan autos del pack de RVA. Asimismo, RVA introduce el concepto de multiplicadores para dichos autos por temporada.

De acuerdo con el sistema de RVA, el puntaje obtenido por un jugador al finalizar una carrera es alterado por el multiplicador del auto que utiliza en dicha carrera. De esta forma, el puntaje obtenido por el jugador incrementa o disminuye dependiendo del multiplicador asociado a su auto.

Los multiplicadores de cada auto son un reflejo de su potencial para desempeñarse en carreras en línea en comparación a los demás autos de su categoría. Por ejemplo, si un auto es increíblemente rápido, fácil de manejar y no tiene dificultad alguna, entonces a este se le asocia con un multiplicador bajo, para así nivelar los puntos que, potencialmente, pueda obtener un jugador al utilizarlo. Por el contrario, si un auto es muy lento, difícil de manejar, y simplemente es peor que los demás autos de su categoría, entonces se le asocia con un multiplicador alto, para así otorgar puntos extra a quien lo utilice.

La decisión de qué autos son buenos y qué autos son malos, en comparación con los demás de su categoría, es tomada por la administración de RVA en base a pruebas de manejo para los autos, y la experiencia previa en temporadas anteriores. A partir de esta decisión, se le otorga un multiplicador a cada auto.

Considerando los distintos multiplicadores que puede tener cada auto, la fórmula para el cálculo del PA es modificada ligeramente, en donde "*mul*" es el multiplicador del auto asociado.

$$PA = \sum_{i=1}^{CC} x = x + f(pos_i) \cdot mul_i$$

Como se explicó anteriormente, existen varias categorías de autos en Re-Volt. Si bien las sesiones de RVA corresponden a una categoría a la vez, el sistema admite que jugadores elijan autos hasta 3 clases por debajo de la categoría de la sesión. Por ejemplo,

si la sesión es publicada como de categoría "Pro", un jugador puede elegir un auto de categoría "Semi-Pro".

De acuerdo con el sistema de RVA, por cada categoría de diferencia entre la sesión y el auto elegido por el jugador, a este se le otorga un bono de 0.25, el cual se adiciona al multiplicador de su auto. Si en el ejemplo anterior la sesión era "Pro" y el auto "Semi-Pro", entonces al multiplicador del auto se le suma directamente 0.25. Si la diferencia fuese de 2 categorías, entonces se le sumaría 0.5.

Teniendo en cuenta lo anterior, la fórmula es modificada nuevamente. En este caso, la variable "*delta*" representa el bono asignado al jugador en la carrera.

$$PA = \sum_{i=1}^{CC} x = x + f(pos_i) \cdot (mul_i + delta_i)$$

## 1.5. Sistema Actual

## 1.6. Problemática Actual

Hoy en día, RVA utiliza una aplicación llamada "RVA-Points", la cual está hecha exclusivamente para procesar los Session Logs generados por RVGL y transformarlos en archivos separados por comas que contienen los resultados de las sesiones en el formato de RVA.

Una vez procesados los Session Logs, RVA lleva la cuenta de sus temporadas y jugadores utilizando Microsoft Excel como una pseudo base de datos para registrarlos, y Dropbox para la sincronización de archivos entre administradores.

Cada temporada, los administradores manejan un documento maestro de Excel, el cual, en resumidas cuentas, es utilizado para registrar los resultados de cada sesión y, a la vez, llevar la cuenta de los puntos por jugador. Esto se consigue utilizando macros y demás componentes característicos de Excel.

Los documentos relacionados a cada temporada son archivados utilizando una carpeta compartida en Dropbox.

Actualmente, el proceso diario para calcular y publicar los resultados de cada sesión es llevado a cabo, de manera semiautomática, por los administradores de RVA. Este proceso consiste en los siguientes pasos:

1. Una vez finalizada la sesión, se busca el Session Log generado por RVGL, y este llega a los administradores de RVA.
2. Se revisan los nombres de usuario manualmente. Esto se hace en caso de que algún jugador haya utilizado un nombre ligeramente distinto al que usa normalmente.
3. El Session Log de la sesión es importado desde "RVA-Points". Este programa procesa el Session Log, y lo transforma en resultados ordenados en el formato de RVA.

4. También desde "RVA-Points", los resultados procesados son exportados a otro archivo separado por comas.
5. Se copian los contenidos del archivo exportado al documento maestro de la temporada actual.
6. Se agrega cierta información de manera manual, como la fecha y número de la sesión.
7. Se revisa manualmente que los nombres de usuario de la sesión correspondan a los nombres que ya figuran en el documento maestro.
8. Se publica una fotografía de la tabla de resultados de la sesión en el Discord de RVA.
9. Se publica una fotografía del ranking actualizado con los datos de la sesión en el Discord de RVA.

A raíz de este largo y tedioso proceso de cálculo y manejo de resultados, se ha dado origen a este proyecto de título, como una oportunidad de mejorar dicho proceso, y poder ofrecer así una mejor experiencia de usuario tanto para los jugadores de RVA, como para los organizadores de la comunidad, quienes dedican su tiempo y esfuerzo a mantener estos registros históricos al día para todos.

### **1.6.1. Diagrama de la Situación en la Actualidad**

Teniendo en cuenta el contexto del problema, la situación en la actualidad puede representarse a través del siguiente diagrama BPMN:

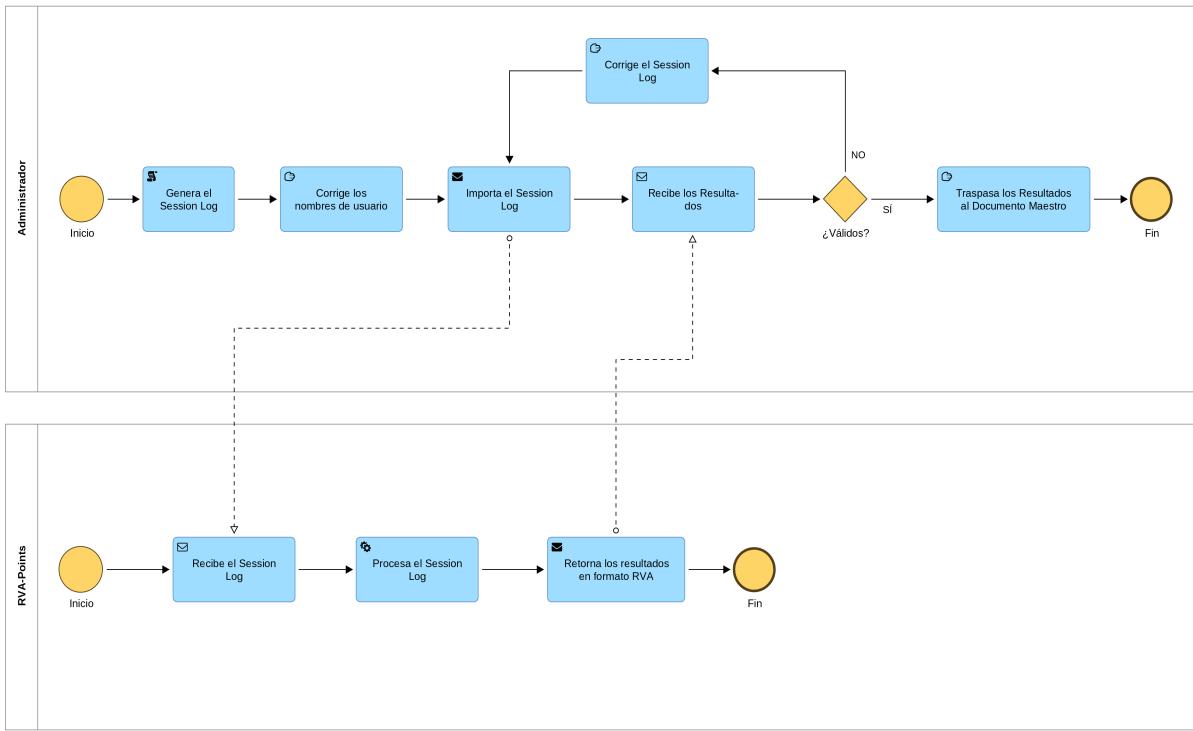


Figura 1.11: Diagrama BPMN del proceso actual de cálculo de puntos

## 1.7. Propuesta de solución

Debe explicar en términos generales cómo las TIC pueden resolver o mejorar la(s) problemática identificada y quienes serán los usuarios principales, que tecnología se utilizaría para dar soporte a la propuesta.

## 1.8. Soluciones similares disponibles

A continuación, se describen las soluciones disponibles que pueden ser catalogadas como similares al proyecto que se presenta.

### 1.8.1. Aplicación para Cálculo de Puntos de Re-Volt I/O

Existe un trabajo similar hace varios años, el cual fue desarrollado por la comunidad europea de RVGL: Re-Volt I/O.

Este trabajo se trata de una aplicación web que permite a los administradores de Re-Volt I/O importar los resultados de las sesiones multijugador, y poder visualizarlos dentro de la misma página; sin embargo, dicho proyecto no cuenta con ningún tipo de interconexión entre resultados, lo que quiere decir que cada sesión de carreras publicada en dicho sitio es independiente de otras. Debido a lo anterior, este proyecto no cuenta

con perfiles de usuario, y por ende no permite visualizar estadísticas de ningún tipo, ni tampoco relacionar tablas de resultados entre si.

En general, esta solución fue concebida para ser algo simple y rápido que sirviera para calcular y renderizar resultados de manera oportuna, y no con una visión de persistencia en mente. A continuación, en la figura 1.12, puede apreciarse la tabla de resultados generada por la aplicación web de Re-Volt: I/O.

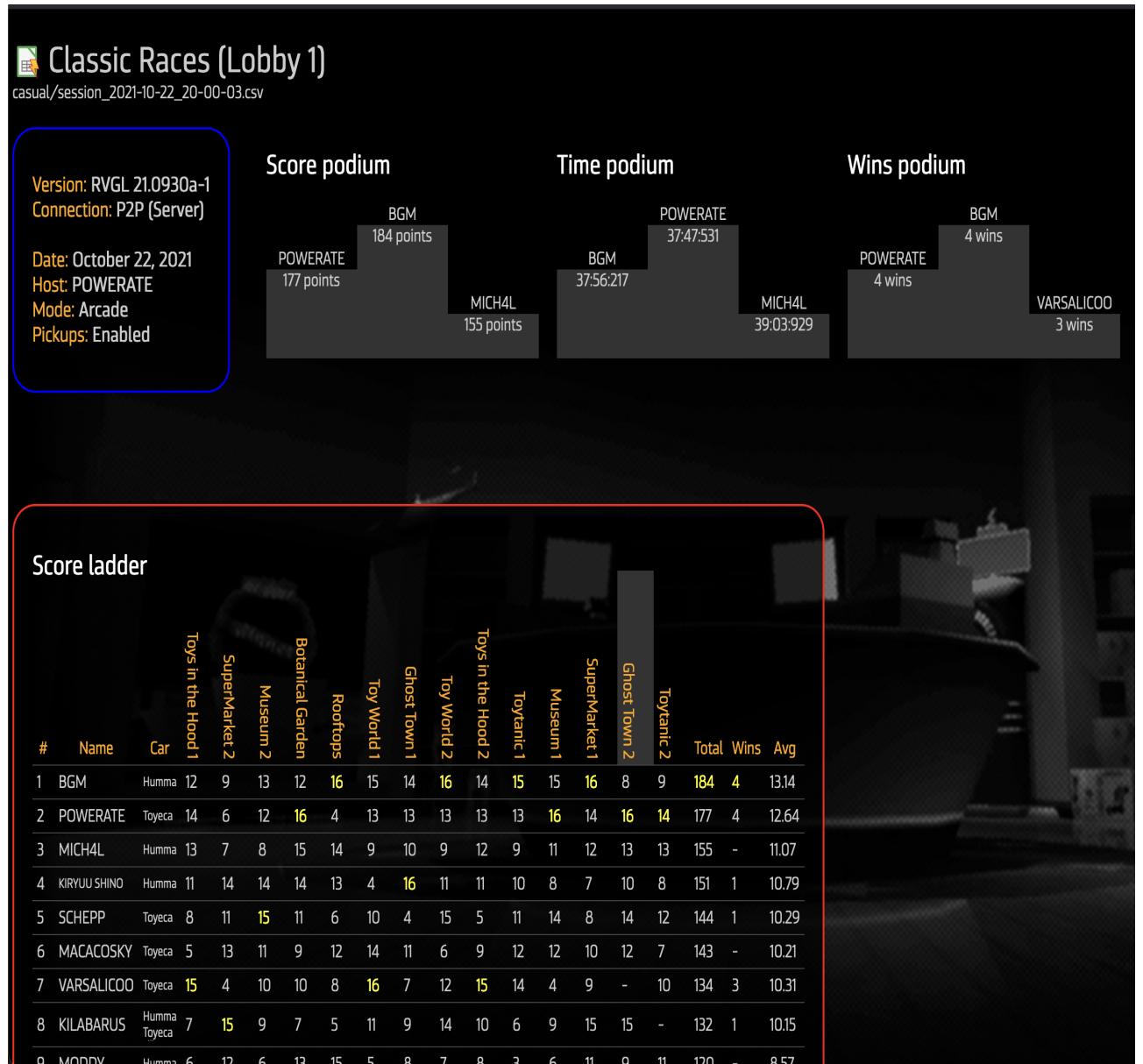


Figura 1.12: Software de cálculo de resultados de Re-Volt: I/O

## **1.9. Justificación del Problema**

Actualmente, no existe ninguna plataforma que permita a los usuarios registrarse o visualizar resultados y estadísticas de las sesiones multijugador organizadas por la comunidad. Lo anterior significa que, cuando se juegan partidas online, no es posible obtener de manera automática rankings, estadísticas por vehículo y menos por usuario, ya que no hay forma de vincular, de manera definitiva, a los jugadores a través de un perfil dentro del juego. Es por ello que un usuario de nuestra comunidad no puede saber cuántas carreras o sesiones ha ganado con cierto auto, o en cierta pista, cómo se compara al resto, qué porcentaje de carreras ha perdido, ganado, etc.

# **Capítulo 2**

# **Proyecto**

## **2.1. Objetivo General del Proyecto**

Desarrollar una aplicación web para Re-Volt America, la comunidad del continente americano formada alrededor del videojuego de carreras Re-Volt, 1999.

## **2.2. Objetivos Específicos del Proyecto**

1. Elaborar una propuesta que consiga atender las necesidades y problemas de los usuarios de Re-Volt America, en relación con el almacenamiento y visualización de resultados de partidas online, además de proveer visibilidad a la comunidad en general.
2. Diseñar la solución de software de procesamiento de datos de sesiones multijugador de Re-Volt, creando interfaces que permitan a los jugadores visualizar los resultados oficiales de las sesiones de carreras en línea, además de acceder a sus estadísticas personales.
3. Implementar la aplicación web, la cual permitirá a los organizadores de RVA subir y publicar los resultados de dichas carreras, además de realizar el lazo entre jugadores y cuentas de usuario de esta. El software implementado permitirá, a su vez, procesar dicha información subida a la web, y así mostrar a los usuarios finales una vista clara de sus resultados en carreras y sus estadísticas personales.

## **2.3. Metodología de Desarrollo**

Para poder definir la metodología de desarrollo a utilizar, primero se debe tener en cuenta la tabla de riesgos asociada, la cual puede apreciarse a continuación.

Tabla de Riesgos			
<b>Experiencia en el Problema</b>	Alta	X	Se tienen años de experiencia con la comunidad de RVA y su sistema interno.
	Baja		
<b>Tamaño del Problema</b>	Grande	X	La cantidad de funcionalidades a implementar es muy alta.
	Pequeño		
<b>Complejidad del Problema</b>	Complejo	X	El sistema de puntos de RVA es difícil de comprender y manejar completamente.
	Simple		
<b>Tamaño del Software</b>	Grande	X	El software a construir requiere muchas funcionalidades.
	Pequeño		
<b>Complejidad Software</b>	Complejo	X	El software debe implementar cálculos complejos (ratios, promedios, etc.).
	Simple		
<b>Experiencia Software</b>	Alta	X	Se tiene una alta experiencia desarrollando software para RVA y su sistema de puntos.
	Baja		
<b>Modularidad Funcional</b>	Existe	X	Las funcionalidades pueden implementarse por separado y luego integrarse.
	No existe		

Según la evaluación del proyecto, se concluyó que este tiene un riesgo total asociado bajo, por lo que se contaría con la libertad de utilizar cualquier tipo de metodología de desarrollo para llevarlo a cabo. Como se cuenta con esta libertad de elección, se busca sacar provecho de ella, y se ha seleccionado una metodología de desarrollo iterativa, a través de la cual se desarrollará en ciclos. Estos ciclos permitirán al software evolucionar a medida que se recibe retroalimentación por parte de los usuarios, corrigiendo errores y añadiendo detalles a medida que se progresá en el desarrollo de la aplicación.

Según el contexto en el que se desarrollará esta aplicación, los usuarios de la comunidad tendrán una importante incidencia en el testeo y uso diario de la misma, lo cual implica que estarán presentes durante el proceso de implementación de varias de las funcionalidades que se pretenden lograr con esta propuesta. Se sabe también con antelación que los usuarios estarán dispuestos a ayudar con las pruebas y el testeo de la aplicación. Esta opción fue seleccionada ya que:

1. Permite más flexibilidad en cuanto a lo que se planea desarrollar como aplicación web, debido a que a medida que avance el proyecto es muy probable que surjan cambios o nuevas ideas a partir de la retroalimentación recibida de la comunidad.
2. Debido a la naturaleza del proyecto, utilizar una metodología iterativa es muy beneficioso ya que ésta permitirá una constante supervisión de los cambios realizados al software, lo que finalmente se traducirá en menos errores, y un producto final que se ajuste y esté a la altura de las necesidades de la comunidad.
3. Con una metodología iterativa será posible contar con prototipos que la comunidad pueda comenzar a utilizar, y a su vez adaptarse a las interfaces y funcionalidades de la aplicación mientras esta se encuentra en desarrollo. Esto es muy importante ya que las sesiones multijugador suelen organizarse a diario, por lo que mientras antes se cuente con una solución funcional, antes podrá la comunidad comenzar a llevar un registro histórico de las partidas online que organiza.

## 2.4. Técnicas y Notaciones

- Diagrama de Casos de Usos.
- BPMN para modelar el proceso de negocio actual.
- Carta Gantt para la planificación inicial del proyecto.
- Patrón de diseño MVC (Modelo, Vista, Controlador).

## 2.5. Estándares de Documentación

- Adaptación Basada en IEEE Software Test Documentation Std 829-1998.
- Adaptación Basada en IEEE Software Requirements Specifications Std 830-1998.

## 2.6. Software, Frameworks y Lenguajes Utilizados

A continuación se lista el software, frameworks y lenguajes de programación, marcado y estilos utilizados para la realización de este proyecto.

Para efectos del siguiente listado, los nombres de las herramientas, frameworks y lenguajes se han redactado en negrita, seguidos de paréntesis en itálica que contienen el número de la versión asociada a cada ítem.

### Lenguajes

- **Ruby** (*3.2.2*): Lenguaje de programación de alto nivel.
- **HAML** (*6.2.3*): Lenguaje de marcado para la abstracción de HTML.
- **Sass** (*6.0*): Lenguaje de extensión para CSS.

### Software

- **MongoDB** (*7.0.3*): Base de datos orientada a documentos JSON.
- **Redis** (*7.0.12*): Almacenamiento en memoria, utilizado para el caché de datos.
- **RubyMine** (*2023.2.2*): Entorno de desarrollo integrado especializado para el trabajo con aplicaciones en Ruby, específicamente para Ruby on Rails.
- **Rake** (*13.1*): Librería de Ruby para la definición de tareas interdependientes.
- **MongoDB Compass** (*1.39.0*): Visor para bases de datos de MongoDB.
- **RedisInsight** (*2.30.0*): Visor para el almacenamiento del caché en Redis.
- **Docker Desktop** (*4.21.0*): Visor y gestor de contenedores de Docker, en formato de aplicación de escritorio multiplataforma.
- **NodeJS** (*16.13.0*): Entorno de servidor multiplataforma utilizado para la conversión de archivos en runtime.

- **Yarn** (*1.22.21*): Gestor de paquetes para JavaScript.
- **Docker** (*24.0.2*): Tecnología que permite crear y utilizar contenedores. Para efectos de este proyecto, es utilizado con el fin de probar el software desarrollado en distribuciones de Linux determinadas.
- **Termius** (*8.7.2*): Cliente SSH.
- **Git/Git Bash** (*2.34.1*): Sistema de control de versiones.
- **Ubuntu LTS** (*18.04.6*): Subsistema de Linux para Windows.

## Frameworks

- **Ruby on Rails** (*7.1*): Framework para desarrollo de aplicaciones web full-stack.
- **Jekyll** (*4.0.0*): Framework para desarrollo de aplicaciones web estáticas escrito en Ruby.
- **Bootstrap** (*4.4.1*): Framework para la creación, manejo de elementos visuales y la responsividad en aplicaciones web.

# **Capítulo 3**

## **Factibilidad**

### **3.1. Factibilidad Técnica**

#### **3.1.1. Conocimientos de los Usuarios**

Para el correcto funcionamiento de la aplicación propuesta, es de esperar que los distintos miembros del staff de RVA, quienes serán los principales usuarios del software, deban tener determinados conocimientos para poder operarla con éxito.

Gracias a que, naturalmente, el staff de RVA lleva un largo tiempo manejando la comunidad y los resultados de las sesiones organizadas, no hace falta mayor capacitación técnica en cuanto al funcionamiento del cálculo de puntos, multiplicadores de autos, y demás características específicas del sistema de RVA. Para efectos de factibilidad técnica, el staff ya cuenta con los conocimientos necesarios para poder migrar a un sistema que sólo busca mejorar y facilitar los procesos actuales.

En cuanto a la aplicación propuesta en este proyecto y su funcionamiento operacional, el staff será habituado al nuevo sistema mediante un video de inducción a la nueva plataforma web. También se asignará un periodo de prueba para que puedan utilizar la página ellos mismos, y así puedan adaptarse fácilmente.

#### **3.1.2. Disponibilidad Profesional**

Para el desarrollo de la aplicación propuesta, se necesita del trabajo de un profesional en el área del desarrollo de software, el cual sea capaz de satisfacer las necesidades de RVA y cumplir los objetivos de desarrollo propuestos.

Para efectos de este proyecto, se cuenta tanto con el tiempo profesional, como también con los conocimientos técnicos requeridos.

Adicionalmente, se cuenta con el equipo físico para poder desarrollar la aplicación, como puede ser un computador, acceso a internet y demás software para desarrollo. A continuación, se presentan tablas de especificación de los equipos físicos y el software con el que se cuenta para desarrollar la aplicación.

Equipos Físicos	
Nombre	Acceso
Windows PC	Equipo personal
MacBook Pro M1	Equipo personal

Software	
Nombre	Acceso
Notepad++	Software libre
MongoDB Compass	Software libre
RedisInsight	Software libre
Git/Git Bash	Software libre
Ubuntu LTS	Software libre
RubyMine	Licencia de estudiante
Termius	Licencia de estudiante
Microsoft Excel	Licencia de estudiante

### 3.1.3. Despliegue y Servidor

Para realizar el despliegue de la aplicación a desarrollar, se ha escogido un servidor de tipo VPS (Servidor Virtual Privado). Las especificaciones técnicas de dicho servidor son las siguientes:

VPS	
Característica	Detalle
Proveedor	DigitalOcean
Región	Nueva York
Sistema Operativo	Ubuntu 22.04 (LTS) x64
Tipo de CPU	Intel Regular
Número de vCPUs	1 CPU
Memoria	2 GB
Almacenamiento (SSD)	50 GB
Transferencia	2 TB

El software instalado en este servidor para el despliegue de la aplicación es el mismo especificado en la sección 2.6.

## 3.2. Factibilidad Operativa

En cuanto a la factibilidad operativa de este proyecto, se sabe que el staff de Re-Volt America tiene gran disposición al cambio, ya que el sistema antiguo no solamente les demanda demasiado tiempo y esfuerzo, sino que, además, resulta poco preciso y muy propenso a errores en su uso diario debido a la gran cantidad de pasos manuales que este conlleva.

La factibilidad operativa es fácilmente demostrada al comparar la solución propuesta con el sistema que ésta pretende reemplazar. Vale decir que, actualmente, quienes hacen uso del sistema de hojas de cálculo maestras en Excel y la aplicación de RVA-Points para el procesamiento de las sesiones, verían su trabajo facilitado en todos los sentidos al contar con una plataforma que se encargue de llevar la cuenta de todo, procesar los resultados y mantener los rankings y temporadas al día de manera automática y confiable.

### 3.3. Factibilidad Económica

A continuación se presenta un detalle de tablas de costos y el flujo de caja asociado al proyecto.

#### 3.3.1. Tablas de Costos

En esta sección se presentan dos tablas de costos relacionadas al proyecto. La primera tabla contiene todo el software utilizado para el desarrollo de este proyecto, mientras que la segunda tabla contempla los costos inherentes de RVA, combinando aquellos costos adquiridos que vienen de antes del despliegue a producción del software que se ha desarrollado.

Software		
Nombre	Acceso	Precio (anual)
<b>Notepad++</b>	Software libre	\$0
<b>MongoDB Compass</b>	Software libre	\$0
<b>RedisInsight</b>	Software libre	\$0
<b>Git/Git Bash</b>	Software libre	\$0
<b>Ubuntu LTS</b>	Software libre	\$0
<b>RubyMine</b>	Licencia de estudiante	\$0
<b>Termius</b>	Licencia de estudiante	\$0
<b>Microsoft Excel</b>	Licencia de estudiante	\$0

Costos de Producción		
Nombre	Proveedor	Precio (anual)
<b>VPS</b>	DigitalOcean	\$125.000
<b>Mailer</b>	Postmark	\$156.000
<b>Dominio (rva.lat)</b>	Namecheap	\$22.600
<b>Git/Git Bash</b>	Software libre	\$0
<b>Ubuntu LTS</b>	Software libre	\$0
<b>Crowdin</b>	Licencia Open Source	\$0
<b>Sentry</b>	Licencia de estudiante	\$0
<b>RubyMine</b>	Licencia de estudiante	\$0
<b>Termius</b>	Licencia de estudiante	\$0
<b>Microsoft Excel</b>	Licencia de estudiante	\$0

Cabe destacar que, dentro de los costos de producción, tanto el precio anual del VPS como el del dominio de RVA son costos adquiridos que han sido mantenidos desde antes de la puesta en marcha de este proyecto. Su mención en la tabla anterior es puramente una formalidad.

### 3.3.2. Flujo de Caja

Para la redacción del flujo de caja, en primer lugar se hablará de los indicadores económicos utilizados para llegar a él, y luego se calculará el Valor Actual Neto (VAN) a 5 años de la puesta en marcha del proyecto.

La valorización del tiempo de trabajo será determinada a partir de promedios y estimaciones que pueden encontrarse hoy en día en el mercado del desarrollo de software. Las referencias serán mencionadas oportunamente.

#### 3.3.2.1. Contexto e Indicadores Económicos

La Tasa Mínima Aceptable de Rendimiento (TMAR) para proyectos de desarrollo de software, a la cual llamaremos " $r$ ", ha sido calculada para este proyecto a partir de la inflación promedio anual reportada por el Banco Central de Chile, y la prima de riesgo asociada a proyectos tecnológicos reportada por el Standish Group International en 2011 (CHAOS Manifesto).

Inflación Promedio Anual	4.3 %
Tasa Prima de Riesgo	21 %

Con la inflación promedio anual y la tasa prima de riesgo, es posible calcular el valor de la TMAR asociada al proyecto.

$$r = 0.04 + 0.21 + (0.04 \cdot 0.21) = 0.2584 \approx 26\%$$

#### 3.3.2.2. Puesta en Marcha

A partir de los datos obtenidos en la subsección 3.3.2.1, es posible realizar un cálculo del valor actual neto (VAN) del proyecto. En este caso, todos los costos asociados figurarán como un ahorro para la comunidad de RVA, ya que debemos considerar que todo lo realizado vendría representar dinero que, potencialmente, la comunidad hubiese tenido que gastar en otro profesional y recursos de no haberse desarrollado el software en cuestión.

La valorización del tiempo empleado en el desarrollo de software a lo largo del proyecto está basada en el sueldo promedio de un analista programador en Chile según Talent.com (\$900.000 mensual; \$5.538 por hora), ajustado a las horas de trabajo efectivas empleadas en el proyecto, las cuales fueron 4 horas de trabajo efectivo durante 6 días de la semana por mes (4 semanas) de desarrollo.

$$\text{Desarrollo} = \$5.538 \cdot (4 \text{ horas} \cdot 6 \text{ días} \cdot 4 \text{ semanas}) = \$531.648$$

Además del tiempo valorizado, también se ha estimado una valorización de los costos de internet y electricidad asociados con el proyecto, basados en su costo promedio mensual y dividiéndolos por la mitad, para así obtener un valor realista.

$$Internet = \frac{\$20.000}{2} = \$10.000$$

$$Electricidad = \frac{\$40.000}{2} = \$20.000$$

A continuación, se presenta una tabla que contiene los gastos mensuales estimados de la puesta en marcha del proyecto:

<b>Costos</b>	
Desarrollo	\$531.648
Internet	\$10.000
Electricidad	\$20.600
<b>Total</b>	<b>\$561.648</b>

Una vez calculado el total de gastos para la puesta en marcha, es posible extraerlo a 4 meses y calcular la inversión inicial del proyecto a través de la siguiente fórmula.

$$I_0 = \$561.648 \cdot 4 \text{ meses} = \$2.246.592$$

### 3.3.2.3. Cálculo del Valor Actual Neto

Siguiendo con la lógica de la valorización de las horas de trabajo efectivas, se tiene que, una vez terminado el proyecto (al final de los 4 meses), la cantidad de horas efectivas necesarias para mantener el proyecto a lo largo del tiempo disminuirá sustancialmente.

Se dirá que, si antes eran 4 las horas efectivas en 6 días de la semana, ahora sólo será 1 hora por 5 días de la semana, ya que el trabajo requerido por la comunidad ya no será de desarrollo de software, sino que de mantenimiento, realización de mejoras oportunas, y soporte para la plataforma.

$$Mantenimiento = \$5.538 \cdot (1 \text{ horas} \cdot 5 \text{ días} \cdot 4 \text{ semanas}) = \$110.760$$

Además de lo anterior, se realiza un ajuste proporcional del costo de internet y electricidad en función de las valorizaciones del tiempo de desarrollo y mantenimiento calculadas.

$$\frac{Desarrollo}{Mantenimiento} = \frac{\$531.648}{\$110.760} = 4.8$$

$$Internet = \frac{\$10.000}{4.8} = \$2.083$$

$$Electricidad = \frac{\$20.600}{4.8} = \$5.208$$

Si proyectamos a 5 años, podemos calcular el Valor Actual Neto (VAN) de ahorro para el proyecto.

	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>
<b>Mantenimiento</b>		\$1.329.120	\$1.329.120	\$1.329.120	\$1.329.120	\$1.329.120
<b>Internet</b>		\$24.996	\$24.996	\$24.996	\$24.996	\$24.996
<b>Electricidad</b>		\$62.496	\$62.496	\$62.496	\$62.496	\$62.496
<b>Hosting</b>		-\$126.312	-\$126.312	-\$126.312	-\$126.312	-\$126.312
<b>Flujo</b>		\$1.290.300	\$1.290.300	\$1.290.300	\$1.290.300	\$1.290.300
<b>Inv. Inicial</b>	\$2.246.592					
<b>Flujo Total</b>	\$6.451.500					

Como clarificación general para la siguiente fórmula, el número de años de proyección vendrá definido como " $n$ ", y " $C$ " representará los flujos de caja.

Asimismo, recordar también que " $r$ " representa la TMAR del proyecto calculada en la subsección 3.3.2.1, e " $I_0$ " representa la inversión inicial calculada en la subsección 3.3.2.2.

$$VAN = I_0 + \sum_{t=i}^n \frac{C_t}{(1+r)^t} = \$5.646.624$$

[https://www.bcentral.cl/web/banco-central/contenido/-/detalle/informe-de-politica-monetaria-septiembre-2023 \(INFLACIÓN\)](https://www.bcentral.cl/web/banco-central/contenido/-/detalle/informe-de-politica-monetaria-septiembre-2023 (INFLACIÓN))

[https://www.immagic.com/eLibrary/ARCHIVES/GENERAL/GENREF/ChaosManifest\\_-2011.pdf \(TASA DE RIESGO\)](https://www.immagic.com/eLibrary/ARCHIVES/GENERAL/GENREF/ChaosManifest_-2011.pdf (TASA DE RIESGO))

<https://radio.uchile.cl/2023/04/13/cuentas-de-electricidad-de-hogares-de-mayor-consumo-subiran-entre-10-y-16/>

<https://www.ine.gob.cl/estadisticas/economia/transporte-y-comunicaciones/estructura-del-transporte-por-carretera/2020/05/16/conexiones-a-internet-en-hogares-del-pa>

<https://cl.talent.com/salary?job=analista+programador>

### 3.4. Conclusión de Factibilidad

Para concluir, podemos decir que en cuanto a los distintos tipos de factibilidad evaluados se cuenta con usuarios técnicamente capaces y con la disponibilidad profesional necesaria para el desarrollo del proyecto. Por el lado operativo, existe disposición al cambio por parte de los administradores y el staff en general. Económicamente, a partir de los costos y beneficios recopilados, se ha obtenido que existe un valor económico sustancial asociado al proyecto.

Gracias al análisis realizado en los puntos anteriores, se puede concluir que el proyecto es factible.

# Capítulo 4

## Arquitectura de Re-Volt America

La arquitectura de Re-Volt America, a nivel de software y ambiente operacional, contempla diversas características que involucran tanto conocimientos técnicos como de Re-Volt y sus elementos propios.

A lo largo de las secciones de este capítulo, se detallará cómo Re-Volt America está construido desde el punto de vista de su arquitectura de software, y cómo todos sus componentes son orquestados para converger en el servicio que provee a sus usuarios.

Dentro de las diversas cosas que se detallarán están:

- Paquete de Contenido de RVA.
- El repositorio de RVA-Data.
- Despliegue de servicios.
- Organización de GitHub.
- Servidor de distribución.

Al final de este capítulo, se concluirá sobre cómo el nuevo sistema desarrollado se integra a la arquitectura de RVA para poder funcionar de manera efectiva y ser lo menos disruptiva posible.

### 4.1. Organización de GitHub

Como introducción general, las organizaciones de GitHub no son más que un conjunto de repositorios de Git. El dueño de la organización de GitHub puede definir ciertos roles y límites de acceso a determinados repositorios, así como también definir variables secretas para el despliegue de servicios, llaves criptográficas asociadas a la organización, dominios, entre otras características administrativas.

Re-Volt America cuenta con una organización de GitHub hace ya varios años. Haciendo uso de esta organización, ha conseguido mantener un control de los archivos y proyectos asociados a la comunidad.

Dentro de la organización, es posible encontrar los siguientes repositorios:

- **Website:** Repositorio de la página web estática de RVA.
- **rva\_packs:** Repositorio para el paquete de contenido de RVA.
- **rva\_cars:** Sub-módulo de rva\_pack. Contiene todos los archivos de autos que se utilizan en RVA.
- **rva\_tracks:** Sub-módulo de rva\_pack. Contiene todos los archivos de pistas que se utilizan en RVA.
- **RVA-Data:** Repositorio de datos estáticos para el backend de RVA.

Todos estos repositorios funcionan de manera conjunta y son mantenidos por la administración de RVA.

A nivel organizacional, se cuenta con algunas variables secretas para el despliegue e integración continua que se deben tener en cuenta. Estas variables son definidas internamente en la organización de GitHub. Los flujos de trabajo de GitHub pueden hacer referencia a ellas agregando el prefijo "*secrets*". Por ejemplo, *secrets.FTP\_PASSWORD*.

Variables Secretas RVA	
Secreto	Detalle
<b>FTP_SERVER</b>	URL del servidor de distribución.
<b>FTP_PASSWORD</b>	Contraseña del servidor FTP de distribución RVA.
<b>DEPLOY_PASSWORD</b>	Contraseña para descifrar la llave de despliegue de Capistrano.

## 4.2. Repositorios del Paquete de Contenido de RVA

Como se explicó en la subsección 1.4.3.2, RVA maneja un paquete de contenido de expansión para RVGL. Este paquete de contenido es versionado a través de Git en los repositorios de `rva_cars/` y `rva_tracks/`. Estos dos repositorios son sub-módulos del repositorio `rva_pack/`. Para integrar todos estos repositorios, se tiene un archivo **Rakefile**, el cual contiene las tareas necesarias para comprimir y desplegar el paquete de contenido de RVA.

El árbol de directorios del repositorio `rva_pack/` se puede ver en la figura 4.1.

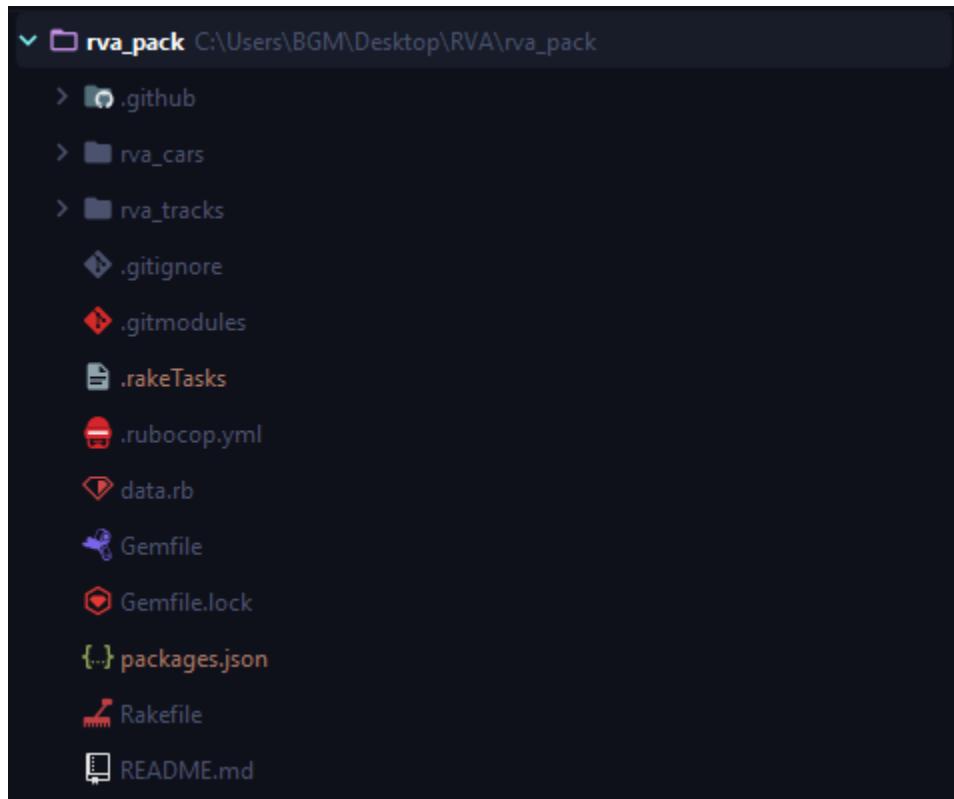


Figura 4.1: Árbol de directorios de rva\_pack

Para versionar los paquetes, se utiliza un sistema basado en fechas. Por ejemplo, si la fecha es 30/10/2023, entonces ésta se traduce a la versión *23.1030a-1*. El sufijo *a-1* es simplemente una forma de diferenciar las versiones que fueron lanzadas en un mismo día.

El versionado de todos los paquetes se encuentra en el archivo *data.rb*, el cual se puede apreciar a continuación.

```
1 module RVA
2   NAME = 'Re-Volt America'
3   DESCRIPTION = "Re-Volt America's Official Content Pack"
4   YEAR = 23
5   MONTH = 10
6   DAY = 30
7   REVISION = 1
8   SUFFIX = 'a'
9   VERSION = "#{YEAR}.#{MONTH < 10 ? "0#{MONTH}" : MONTH}##{DAY < 10 ?
10  ↳ "0##{DAY}" : DAY}##{SUFFIX}-##{REVISION}"
11
12 module RVACars
13   NAME = 'rva_cars'
14   DESCRIPTION = "Re-Volt America's Cars Pack"
15   YEAR = 23
16   MONTH = 10
17   DAY = 30
18   REVISION = 1
19   SUFFIX = 'a'
20   VERSION = "#{YEAR}.#{MONTH < 10 ? "0#{MONTH}" : MONTH}##{DAY < 10
21  ↳ ? "0##{DAY}" : DAY}##{SUFFIX}-##{REVISION}"
22   URL = 'https://distribute.rva.lat/rva/rva_cars.zip'
23 end
24
25 module RVATracks
26   NAME = 'rva_tracks'
27   DESCRIPTION = "Re-Volt America's Tracks Pack"
28   YEAR = 23
29   MONTH = 10
30   DAY = 30
31   REVISION = 1
32   SUFFIX = 'a'
33   VERSION = "#{YEAR}.#{MONTH < 10 ? "0#{MONTH}" : MONTH}##{DAY < 10
34  ↳ ? "0##{DAY}" : DAY}##{SUFFIX}-##{REVISION}"
35   URL = 'https://distribute.rva.lat/rva/rva_tracks.zip'
36 end
37 end
```

Listing 1: Estructura de versiones de paquetes de RVA (*data.rb*)

El procedimiento de compilación y despliegue de los paquetes es el siguiente:

1. Se comprimen *rva\_cars/* y *rva\_tracks/* en *rva\_cars.zip* y *rva\_tracks.zip*, respectivamente.
2. Se calcula un checksum para cada archivo comprimido.
3. Se extraen las versiones de cada uno de los paquetes.
4. Se genera una archivo JSON llamado *packages.json*, el cual contiene toda la información del paquete de RVA en un formato interpretable por RVGL.
5. El archivo *package.json* es enviado, a través de FTP, al servidor de distribución de RVA (<https://distribute.rva.lat>).

Como ejemplo general, el archivo *packages.json* contiene información como el nombre que engloba a los paquetes, "Re-Volt America" en este caso. También cuenta con una versión global ("23.1030a-1") y versiones para cada uno de los paquetes. Todos los paquetes cuentan con una "url", la cual apunta a un archivo comprimido listo para ser descargado.

El archivo *packages.json* compilado se ve de la siguiente manera:

```
1  {
2      "name": "Re-Volt America",
3      "version": "23.1030a-1",
4      "packages": {
5          "rva_cars": {
6              "description": "Re-Volt America's Cars Pack",
7              "version": "23.1030a-1",
8              "checksum": "f8d552",
9              "url": "https://distribute.rva.lat/rva/rva_cars.zip"
10         },
11         "rva_tracks": {
12             "description": "Re-Volt America's Tracks Pack",
13             "version": "23.1030a-1",
14             "checksum": "8306f6",
15             "url": "https://distribute.rva.lat/rva/rva_tracks.zip"
16         }
17     }
18 }
```

Listing 2: Estructura JSON para la distribución de paquetes RVGL (package.json).

## 4.3. Repositorio de RVA-Data

El repositorio de RVA-Data contiene toda la información estática de los modelos de autos y pistas del paquete de RVA en formato CSV. Estos son los archivos CSV que utilizará la plataforma desarrollada en este proyecto de título para realizar la importación de modelos a la base de datos. Este repositorio es mantenido manualmente por los administradores de la comunidad.

El sistema de versionado es exactamente el mismo que el que utiliza el paquete de contenido de RVA.

```
1 module RVAData
2   YEAR = 23
3   MONTH = 10
4   DAY = 30
5   REVISION = 2
6   SUFFIX = 'a'
7   VERSION = "#{YEAR}.#{MONTH < 10 ? "0#{MONTH}" : MONTH}##{DAY < 10 ?
8     ↳ "0##{DAY}" : DAY}##{SUFFIX}-##{REVISION}"
9 end
```

Listing 3: Estructura de versionado de RVA-Data (*data.rb*)

## 4.4. Servidor de Distribución

Re-Volt America cuenta con un servidor de distribución. En este servidor, se encuentra funcionando **vsftpd** (servidor FTP). Este servidor está configurado de tal manera que puede recibir conexiones ajenas y permite la descarga pública de archivos desde el mismo.

La URL en la que funciona este servidor es <https://distribute.rva.lat/>. Todos los administradores cuentan con credenciales FTP para el servidor, con lo que pueden conectarse y realizar cambios manuales; sin embargo, para lo que respecta a la integración continua, existe el usuario **deploy** en la máquina. Este usuario cuenta también con su set de credenciales y puede subir archivos al servidor.

De esta forma, si se desea subir archivos al servidor mediante el protocolo FTP, basta con un usuario y contraseña.

## 4.5. Despliegue e Integración Continua

El despliegue e integración continua funciona con GitHub Actions. En este contexto, GitHub utiliza un archivo YML para la configuración del despliegue.

```
1 name: deploy
2 on:
3   push:
4     branches:
5       - production
6 jobs:
7   deploy:
8     runs-on: ubuntu-latest
9     steps:
10      - uses: actions/checkout@v1
11      - name: Set up Ruby 2.7.3
12        uses: ruby/setup-ruby@v1
13        with:
14          ruby-version: 2.7.3
15      - name: Bundle gems
16        run: |
17          gem install bundler
18          bundler install --jobs 4 --retry 3
19      - name: Run Rake tasks
20        run: |
21          bundle exec rake
22      - name: Prepare FTP directories
23        run: |
24          mkdir rva_data
25          mv yml rva_data
26          mv rva_data.json rva_data
27      - name: Deploy via FTP
28        uses: SamKirkland/FTP-Deploy-Action@4.3.0
29        with:
30          server: ${{ secrets.FTP_SERVER }}
31          username: rva
32          password: ${{ secrets.FTP_PASSWORD }}
33          local-dir: "./rva_data/"
34          server-dir: "rva_data/"
```

Listing 4: Estructura de despliegue de RVA-Data (*deploy.yml*)

Para el despliegue y la integración continua de todos los componentes de RVA, se cuenta con una rama llamada *production* en los repositorios de rva\_pack y RVA-Data. Al enviarse cambios a esta rama, se gatilla el workflow listado anteriormente, haciendo llegar una nueva versión al servidor de distribución.

# **Capítulo 5**

## **Requerimientos del Software**

### **5.1. Límites**

- El software no permitirá relacionar automáticamente cualquier nombre de jugador con su perfil de usuario registrado en ella.
- El software no registrará estadísticas de manera retroactiva para jugadores no registrados.
- El software no permitirá visualizar carreras de manera individual.
- Las estadísticas por jugador no pueden ser editadas manualmente, sino que la modificación de estas debe ser un efecto de subir o eliminar sesiones.
- Las sesiones eliminadas no se podrán visualizar. Esto quiere decir que las sesiones que se eliminan no podrán ser accedidas nunca más una vez borradas.
- Los usuarios no podrán cancelar sus cuentas por si mismos.
- El software no permitirá editar los modelos de sesiones de manera manual.

### **5.2. Caracterización de los Usuarios**

Como se ha mencionado a lo largo de este informe, los usuarios a quienes apunta este proyecto de título son aquellos que forman parte de la comunidad de Re-Volt America. Estos usuarios, de manera general, se caracterizan por:

- Pertenencia a un grupo etario entre 14 y 26 años.
- Competencias técnicas en uso de software similares.
- Familiaridad con el juego y el sistema de puntos de RVA.

Por otra parte, tenemos la definición de quienes pueden ser clasificados como usuarios del software, así como la frecuencia con la que utilizarán la plataforma y su rol dentro de la misma:

Usuarios		
Rol	Nivel	Descripción
<b>Administrador</b>	Primario	Utiliza todas las funciones de la página, como subir autos, pistas, sesiones y manejo general del software. Configura el software para el resto de los usuarios.
<b>Organizador</b>	Primario	Utiliza las funciones específicamente relacionadas a la subida y manejo de sesiones en la página.
<b>Moderador</b>	Secundario	Utiliza las funciones específicas de manejo de usuarios, tal como la edición de perfiles o aplicación de infracciones.
<b>Jugador</b>	Terciario	Utiliza la página sólo para visualizar la información que esta ofrece.

### 5.3. Objetivo General del Software

El sistema manejará información del proceso de cálculo de resultados de carreras para que la comunidad centralice los rankings y estadísticas por usuario dentro del mismo, es decir, reducirá el trabajo manual requerido actualmente para este procesamiento de resultados, y así hará más eficiente todo el proceso que conlleva mantener los rankings actualizados.

#### 5.3.1. Objetivos Específicos del Software

- El sistema permite que los administradores de la comunidad puedan subir los archivos de resultados generados por RVGL y, de esta forma, los resultados son generados automáticamente dentro de la aplicación. Esto elimina el tiempo de los organizadores de calcular los resultados manualmente.
- El sistema permite que los usuarios puedan ver sus estadísticas en tiempo real, lo que elimina la necesidad de llevar la cuenta de manera manual por cada uno de ellos.
- El sistema permite enlazar los nombres de usuario utilizados en las sesiones multi-jugador de RVGL a perfiles dentro de la aplicación, lo cual hace posible la recopilación y atribución de métricas individuales por jugador, y agiliza la visualización de resultados.
- El sistema permite llevar un registro histórico de manera automática según se suben y se procesan los archivos de resultados en la aplicación, lo cual elimina completamente la necesidad de la comunidad de mantener toda esta información actualizada de manera manual.

## 5.4. Requerimientos Funcionales del Software

Módulo de Registro de Autos de Re-Volt America	
<b>Id</b>	<b>Descripción</b>
<b>RF_01</b>	La plataforma contará con un módulo de creación de autos. Dentro de este módulo, los autos deben ser relacionados con una temporada. No podrán existir autos con nombres duplicados dentro de la misma temporada. Los autos deben ser importados desde un archivo separado por comas, el cual debe contar con todos los parámetros obligatorios del modelo de auto. Sólo los administradores pueden crear autos.
<b>RF_02</b>	La plataforma contará con un módulo de visualización de autos. El listado estará separado por temporadas y por clases de autos. Este módulo estará disponible para cualquier tipo de usuario.
<b>RF_03</b>	La plataforma contará con un módulo de edición de un auto. Los autos deben contar con todos los parámetros obligatorios para ser modificados. Este módulo estará disponible sólo para los administradores.
<b>RF_04</b>	La plataforma contará con un módulo de eliminación de un auto. La eliminación no tendrá efectos secundarios a nivel de la base de datos, ya que la aplicación estará preparada para manejar excepciones cuando las entradas de corredores estén asociadas a un auto que no existe. Este módulo sólo estará disponible para administradores.
<b>RF_05</b>	La plataforma contará con un módulo de visualización de un sólo auto. En esta vista se podrán ver todos los parámetros del auto. Este módulo estará disponible para cualquier tipo de usuario.

Módulo de Registro de Pistas de Re-Volt America	
<b>Id</b>	<b>Descripción</b>
<b>RF_06</b>	La plataforma contará con un módulo de creación de pistas. Dentro de este módulo, las pistas deben ser relacionados con una temporada. No podrán existir pistas con nombres duplicados dentro de la misma temporada. Las pistas deben contar con todos los parámetros obligatorios del modelo de pista. Sólo los administradores pueden crear pistas.
<b>RF_07</b>	La plataforma contará con un módulo de visualización de pistas. El listado estará separado por temporadas y paginado. Este módulo estará disponible para cualquier tipo de usuario.
<b>RF_08</b>	La plataforma contará con un módulo de edición de una pistas. Las pistas deben contar con todos los parámetros obligatorios para ser modificadas. Este módulo estará disponible sólo para los administradores.
<b>RF_09</b>	La plataforma contará con un módulo de eliminación de una pista. La eliminación no tendrá efectos secundarios a nivel de la base de datos, ya que la aplicación estará preparada para manejar excepciones cuando las carreras estén asociadas a una pista que no existe. Este módulo sólo estará disponible para administradores.
<b>RF_10</b>	La plataforma contará con un módulo de visualización de una sola pista. En esta vista se podrán ver todos los parámetros de la pista. Este módulo estará disponible para cualquier tipo de usuario.

Módulo de Registro de Temporadas de Re-Volt America	
<b>Id</b>	<b>Descripción</b>
<b>RF_11</b>	La plataforma contará con un módulo de creación de temporadas. Las temporadas deben contar con todos los parámetros obligatorios. Al momento de crear una temporada, sus 6 rankings asociados deberán crearse automáticamente. Sólo los administradores pueden crear temporadas.
<b>RF_12</b>	La plataforma contará con un módulo de visualización de temporadas. El listado contendrá todas las temporadas, las cuales vendrán ordenadas por fecha de inicio. Este módulo estará disponible para cualquier tipo de usuario.
<b>RF_13</b>	La plataforma contará con un módulo de edición de una temporada, las cuales deben contar con todos los parámetros obligatorios para ser modificados. Este módulo estará disponible sólo para los administradores.
<b>RF_14</b>	La plataforma contará con un módulo de eliminación de una temporada. Al eliminarse una temporada, se eliminarán también todos sus rankings asociados. Al mismo tiempo, todas las sesiones asociadas a dichos rankings también serán eliminadas. Las estadísticas globales de cada jugador también serán substraídas de sus perfiles. Este módulo sólo estará disponible para administradores.
<b>RF_15</b>	La plataforma contará con un módulo de visualización de una sola temporada. En esta vista se podrán ver los rankings asociados y la tabla de resultados de la temporada. Este módulo estará disponible sólo para cualquier tipo de usuario.

Módulo de Visualización de Rankings de Re-Volt America	
<b>Id</b>	<b>Descripción</b>
<b>RF_16</b>	La plataforma contará con un módulo de visualización de rankings, el cual se encontrará dentro de cada visualización de temporada. Este módulo estará disponible para cualquier tipo de usuario.
<b>RF_17</b>	La plataforma contará con un módulo de visualización de un solo ranking. En esta vista se podrán ver todas las sesiones asociadas al ranking, junto con su tabla de resultados acumulados. Este módulo estará disponible para cualquier tipo de usuario.

Módulo de Sesiones de Re-Volt America	
<b>Id</b>	<b>Descripción</b>
<b>RF_18</b>	La plataforma contará con un módulo de subida de sesiones en forma de archivo separado por comas. El archivo separado por comas deberá ser un Session Log generado por RVGL. Este módulo deberá permitir al usuario seleccionar un archivo separado por comas. Las sesiones deben contar con todos los parámetros obligatorios. Sólo los administradores pueden crear sesiones.
<b>RF_19</b>	La plataforma contará con un módulo de visualización de sesiones. El listado contendrá las últimas sesiones del ranking vigente de la temporada actual. Este módulo estará disponible para cualquier tipo de usuario.
<b>RF_20</b>	La plataforma contará con un módulo de eliminación de una sesión. La eliminación gatilla la sustracción de las estadísticas de los usuarios que participaron en ella. Este módulo sólo estará disponible para administradores.
<b>RF_21</b>	La plataforma contará con un módulo de visualización de una sola sesión. En esta vista se podrán ver todos los parámetros de la sesión, ordenados en el formato de RVA. Este módulo estará disponible para cualquier tipo de usuario.

## 5.5. Requerimientos No Funcionales del Software

La presente sección hablará de los requerimientos no funcionales de la aplicación desarrollada. Todos los requerimientos no funcionales se relacionarán con uno o más atributos. Si un atributo aplica a un requerimiento no funcional, eso quiere decir que el requerimiento contribuye a la calidad del software desarrollado a través de ese atributo. Todos los atributos listados están basados en la norma ISO 25010.

<b>RNF_01</b>		
<b>Descripción</b>	La plataforma contará con una API REST, la cual estará diseñada para ser consumida por aplicaciones de terceros. Todos los endpoints estarán disponibles sólo para lectura por parte de terceros.	
<b>Atributo</b>	<b>Aplica</b>	<b>Especificación</b>
Adecuación Funcional	X	La API contribuye a la corrección funcional, ya que facilita la obtención de datos precisos del sistema a terceros.
Eficiencia de Desempeño		
Compatibilidad	X	La API contribuye a la coexistencia con otras piezas de software independientes, ya que permite a dicho software consumir información del sistema en tiempo real.
Usabilidad		
Fiabilidad	X	La API contribuye a la madurez del software, ya que es gracias a ella que el sistema puede satisfacer las necesidades de los usuarios que consumen información del mismo.
Seguridad	X	Gracias al diseño de la API, sólo se exponen endpoints de lectura, por lo que esta contribuye a la confidencialidad e integridad de la información.
Mantenibilidad		
Portabilidad		

RNF_02		
Descripción	<p>La plataforma contará con una separación interna de roles a nivel de software. Estos roles serán: administrador, organizador, moderador y jugador, siendo el último denotado por no tener ninguno de los otros roles. El rol del primer administrador del sistema deberá ser asignado de manera interna. Una vez que existe un administrador, este podrá modificar los roles de todos los usuarios desde la misma página web.</p>	
Atributo	Aplica	Especificación
Adecuación Funcional	X	Contribuye a la completitud y pertinencia funcional, ya que, gracias a contar con una separación de privilegios de usuario, el sistema puede proveerles opciones específicas basado en la jerarquía de roles definida.
Eficiencia de Desempeño		
Compatibilidad		
Usabilidad		
Fiabilidad		
Seguridad	X	La presencia de roles en el sistema contribuye directamente a la autenticidad, ya que con ellos el sistema es capaz de decernir si el usuario debiera poder realizar una determinada acción o no basándose en su identidad dentro del software.
Mantenibilidad		
Portabilidad		

RNF_03		
Descripción	El código estará versionado a través de Git, utilizando GitHub como plataforma de almacenamiento en la nube para el control de versiones. El proyecto será de código abierto, y se contará con un sistema de integración continua para el software implementado directamente en GitHub. A través de este sistema se podrá ser desplegar el software de manera automática cuando se detecten cambios en una rama de Git determinada.	
Atributo	Aplica	Especificación
Adecuación Funcional		
Eficiencia de Desempeño		
Compatibilidad	X	Contribuye a la coexistencia del software con otros sistemas, ya que, gracias a estar versionado en Git y ser de código abierto, los mantenedores y desarrolladores de otras piezas de software podrán tener acceso a toda la información técnica que necesiten para poder integrar sus productos con la plataforma de RVA exitosamente.
Usabilidad	X	Existe una contribución al reconocimiento de la adecuación, ya que un usuario, al poder visualizar el historial de desarrollo completo en Git, podrá entender si el software, a nivel técnico, es adecuado para sus necesidades.
Fiabilidad	X	Gracias al sistema de despliegue que detalla, este requerimiento no funciona contribuye a la madurez, disponibilidad y tolerancia a fallos del sistema. Al contar con un sistema de despliegue y versionado del proyecto, los desarrolladores pueden trabajar en él y hacer llegar actualizaciones muchísimo más rápido al entorno de producción, y por ende a los usuarios finales del software.
Seguridad	X	Contribuye al no repudio, ya que, gracias a que el control de versiones lleva registro de todo lo que se modifica en la base de código, será posible rastrear malas decisiones o errores con facilidad.
Mantenibilidad	X	Este requerimiento contribuye a la modularidad, reusabilidad analizabilidad y a la capacidad del software para ser modificado y probado. Todo esto gracias a que los cambios realizados en el software son documentados a través del control de versiones.
Portabilidad	X	Contribuye a la adaptabilidad del software, y a la capacidad del mismo para ser instalado. Como todo el software es público y se encuentra en Git, analizarlo para adaptarlo resulta muy sencillo. Por otra parte, el sistema de despliegue facilita enormemente el instalar la aplicación en un entorno determinado.

## 5.6. Interfaces Internas de Salida

<b>Id</b>	<b>Nombre</b>	<b>Detalle de Datos</b>
<b>IN_01</b>	Car	name, speed, accel, weight, multiplier, folder_name, category, stock, season
<b>IN_02</b>	Track	name, short_name, difficulty, lenght, folder_name, stock, season
<b>IN_03</b>	Season	name, start_date, end_date, current, racer_result_entries
<b>IN_04</b>	Ranking	number, racer_result_entries, season
<b>IN_05</b>	Session	number, host, version, physics, protocol, pickups, date, teams, category, session_log_data, ranking, races, racer_result_entries
<b>IN_06</b>	Race	track_name, laps, racers_count, racer_entries
<b>IN_07</b>	RacerEntry	car_name, position, username, time, best_lap, finished, cheating, laps, racers_count
<b>IN_08</b>	RacerResultEntry	username, country, session_count, race_count, positions_sum, average_position, obtained_points, official_score, participation_multiplier, team
<b>IN_09</b>	User	username, encrypted_password, reset_password_token, reset_password_sent_at, remember_created_at, sign_in_count, current_sign_in_at, last_sign_in_at, current_sign_in_ip, last_sign_in_ip, confirmation_token, confirmation_token, confirmed_at, confirmation_sent_at, unconfirmed_email, failed_attempts, unlock_token, locked_at, admin, mod, organizar, locale, country, profile, stats

## 5.7. Interfaces Externas de Salida

<b>Id</b>	<b>Nombre</b>	<b>Detalle de Datos</b>	<b>Medio de Salida</b>
<b>OUT_01</b>	Car	name, speed, accel, weight, multiplier, folder_name, category, stock, season	Pantalla
<b>OUT_02</b>	Track	name, short_name, difficulty, lenght, folder_name, stock, season	Pantalla
<b>OUT_03</b>	Season	name, start_date, end_date, current, racer_result_entries	Pantalla
<b>OUT_04</b>	Ranking	number, racer_result_entries, season	Pantalla
<b>OUT_05</b>	Session	number, host, version, physics, protocol, pickups, date, teams, category, session_log_data, ranking, races, racer_result_entries	Pantalla, Archivo CSV.
<b>OUT_06</b>	User	username, admin, mod, organizer, locale, country, profile, stats	Pantalla

# **Capítulo 6**

## **Análisis Funcional**

### **6.1. Actores**

En el software desarrollado, cada rol dentro del staff de RVA ha sido considerado como un actor diferente. Para el contexto del presente proyecto, el cargo de cada actor corresponde a su nombre, lo que quiere decir que si un actor lleva por nombre "Administrador", se entenderá que su cargo es "Administrador de la comunidad de RVA". Sólo los jugadores no cuentan con un cargo dentro de RVA.

La especificación de todos los actores se puede encontrar a continuación en la tabla TABLENUM.

Actores			
Actor	Función	Conocimientos	Privilegio
<b>Administrador</b>	Cumple con todas las funciones dentro de la comunidad como mantener autos, pistas, sesiones y manejo general del software. Configura el software para el resto de los usuarios.	Requiere conocimiento sobre como funciona el sistema en su totalidad, comprendiendo también el contexto en el que funciona RVA como comunidad.	Máximo.
<b>Organizador</b>	Cumple con funciones específicamente relacionadas al manejo de sesiones multijugador y el procesamiento de los puntajes.	Requiere conocimiento específico de como subir archivos de sesión a la aplicación web, y un contexto general de como funciona RVA.	Intermedio (módulo de sesiones).
<b>Moderador</b>	Utiliza las funciones específicas de manejo de usuarios, tal como la edición de perfiles o aplicación de infracciones.	Requiere conocimientos que se limitan al manejo de usuarios dentro de la web.	Intermedio (manejo de usuarios).
<b>Jugador</b>	Utiliza la página sólo para visualizar la información que esta ofrece.	No requiere conocimientos técnicos más allá de iniciar sesión.	Ninguno.

## 6.2. Casos de Uso

Esta sección contiene todos los diagramas de casos de uso relevantes para el proyecto. Todos los casos de uso cuentan con su respectiva especificación y detalle según corresponda.

### 6.2.1. Diagramas de Casos de Uso

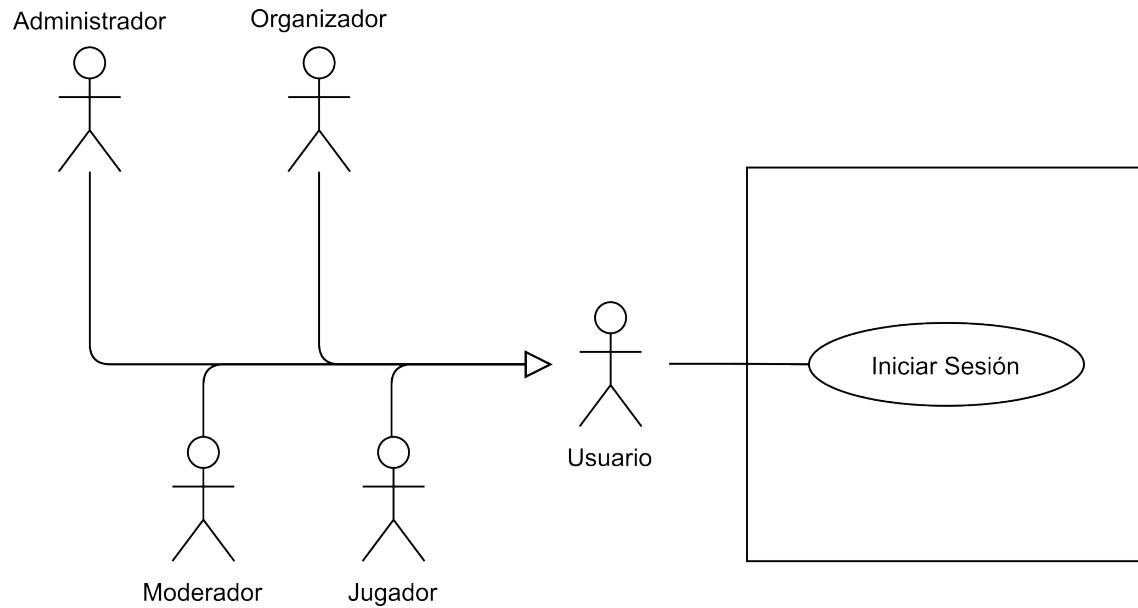


Figura 6.1: Diagrama de Casos de Uso del módulo de privilegios

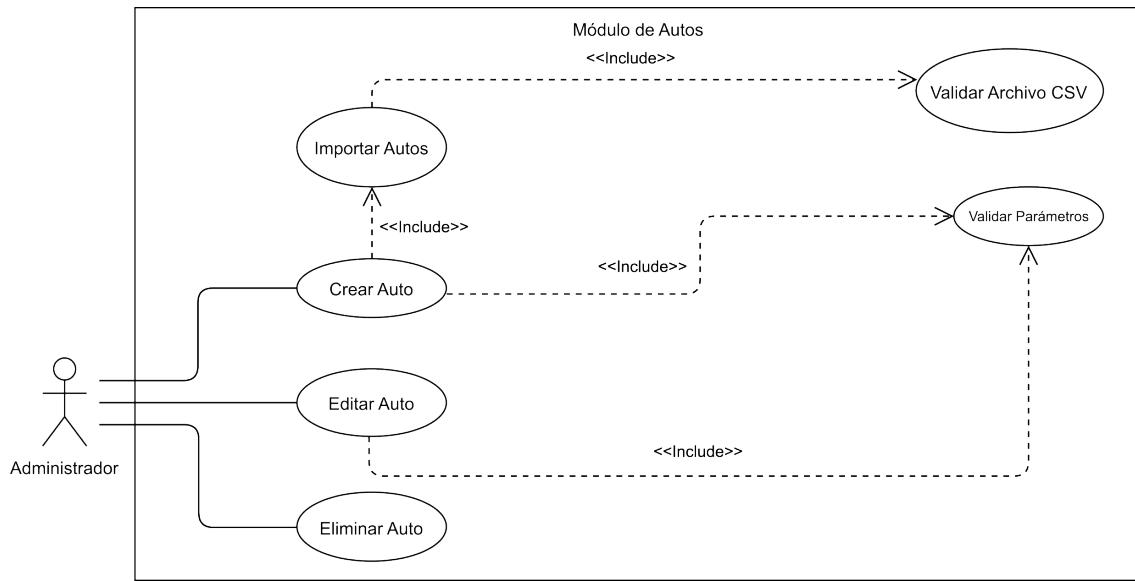


Figura 6.2: Diagrama de Casos de Uso del módulo de autos 1

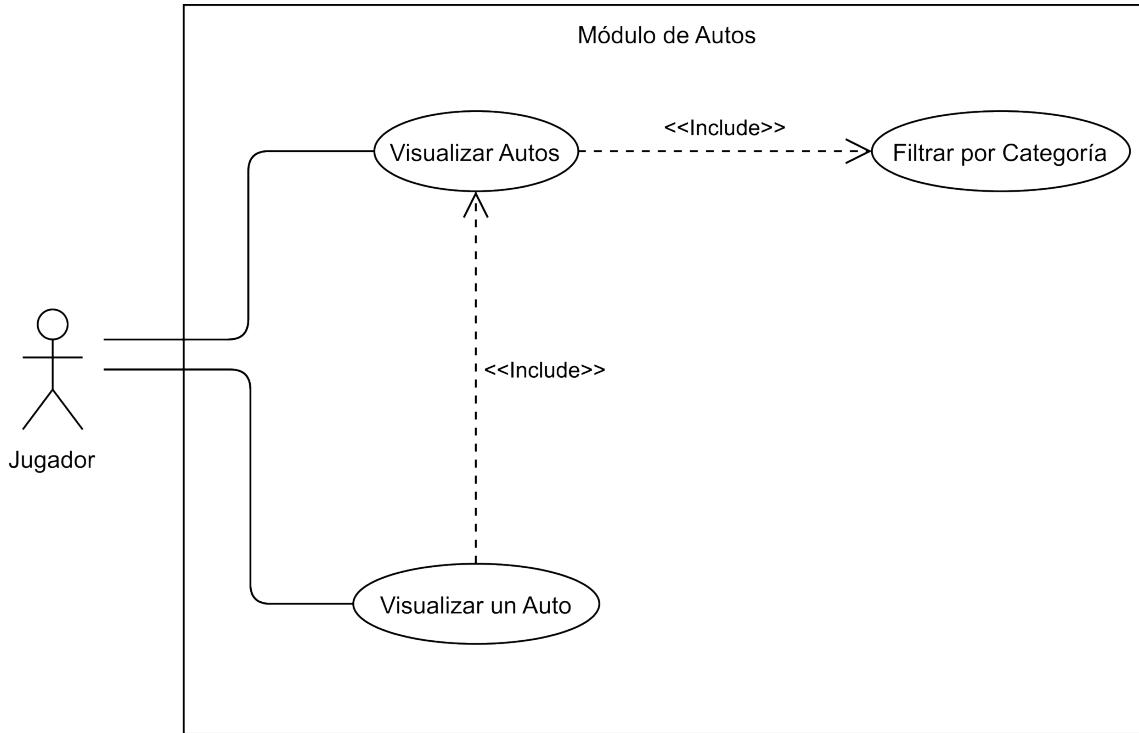


Figura 6.3: Diagrama de Casos de Uso del módulo de autos 2

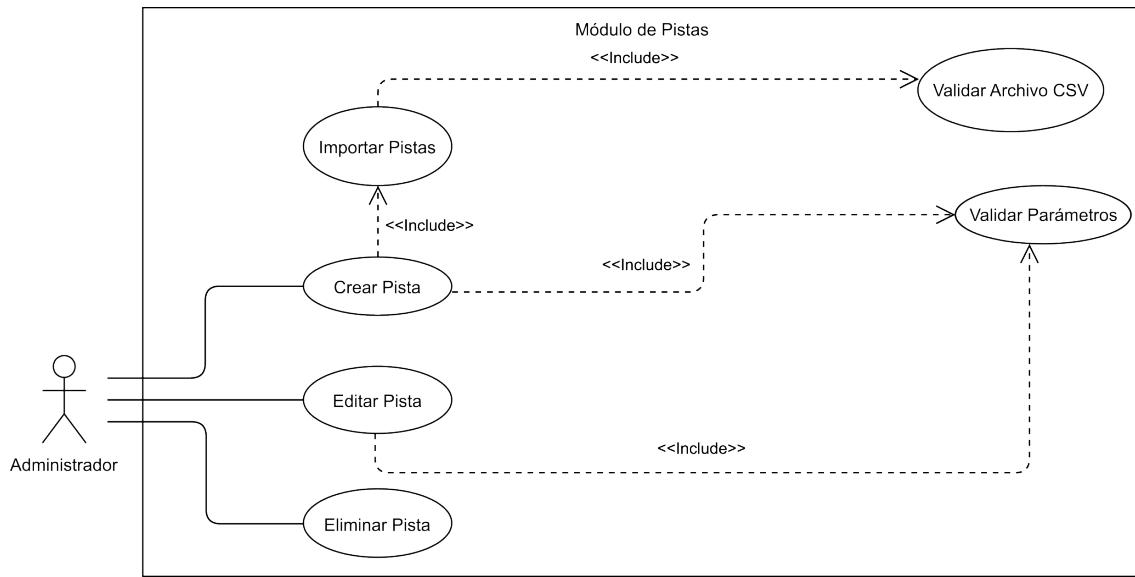


Figura 6.4: Diagrama de Casos de Uso del módulo de pistas 1

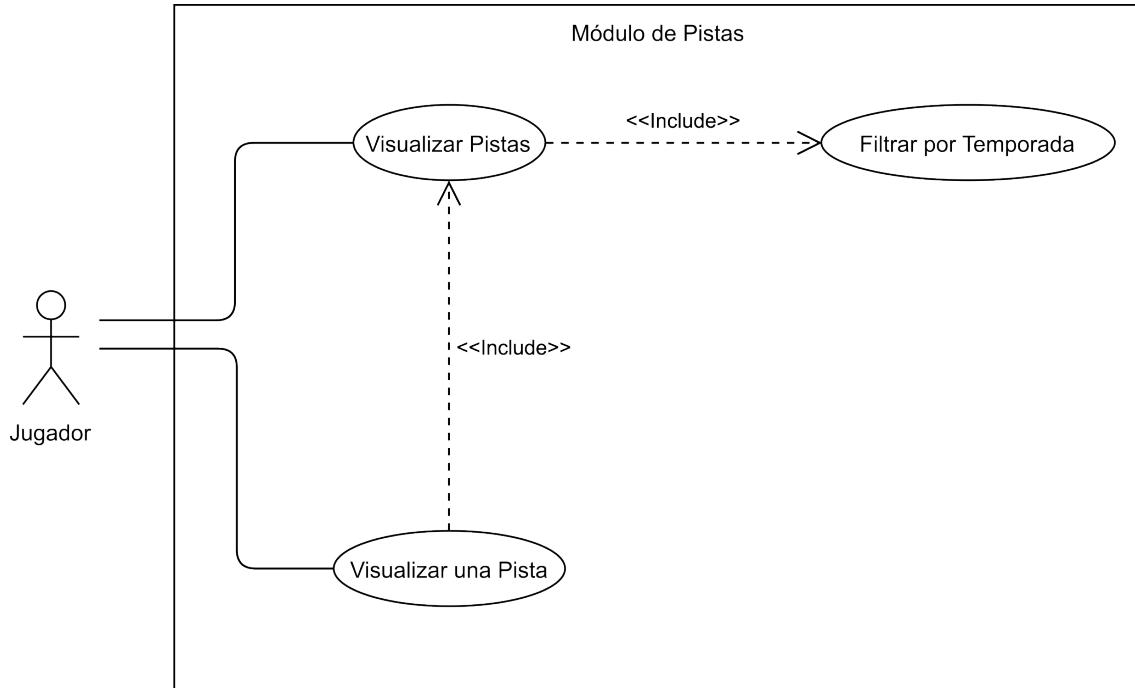


Figura 6.5: Diagrama de Casos de Uso del módulo de pistas 2

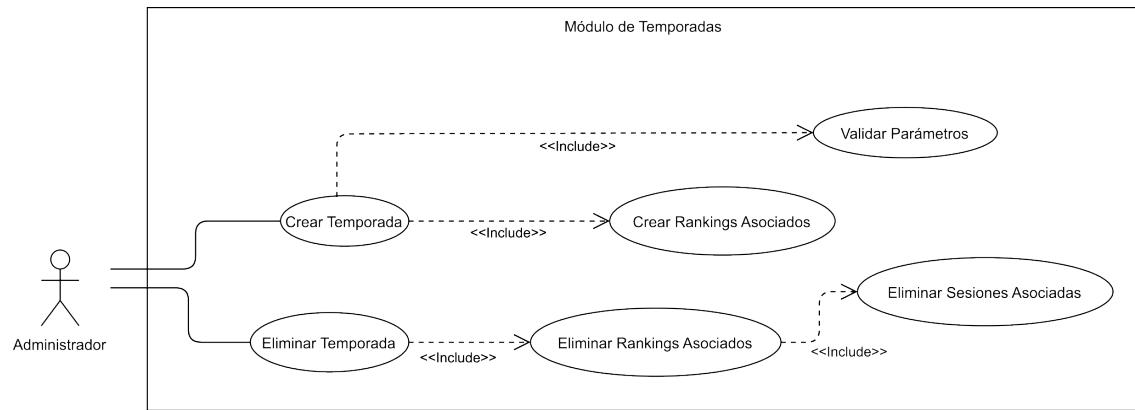


Figura 6.6: Diagrama de Casos de Uso del módulo de temporadas 1

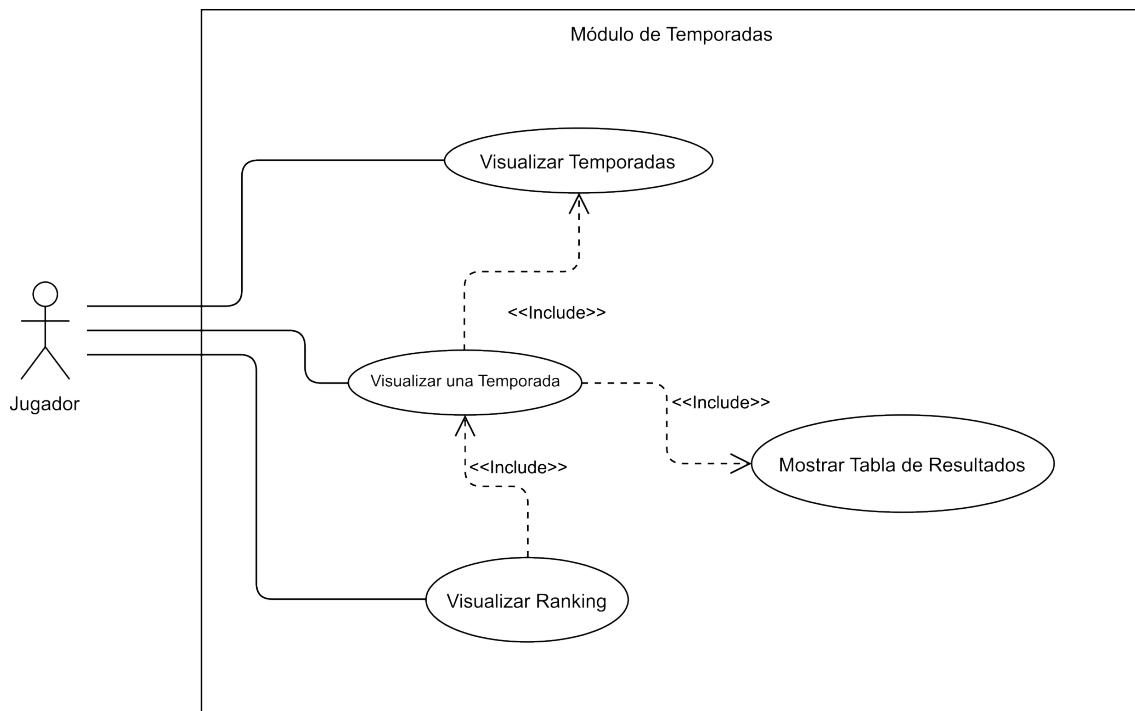


Figura 6.7: Diagrama de Casos de Uso del módulo de temporadas 2

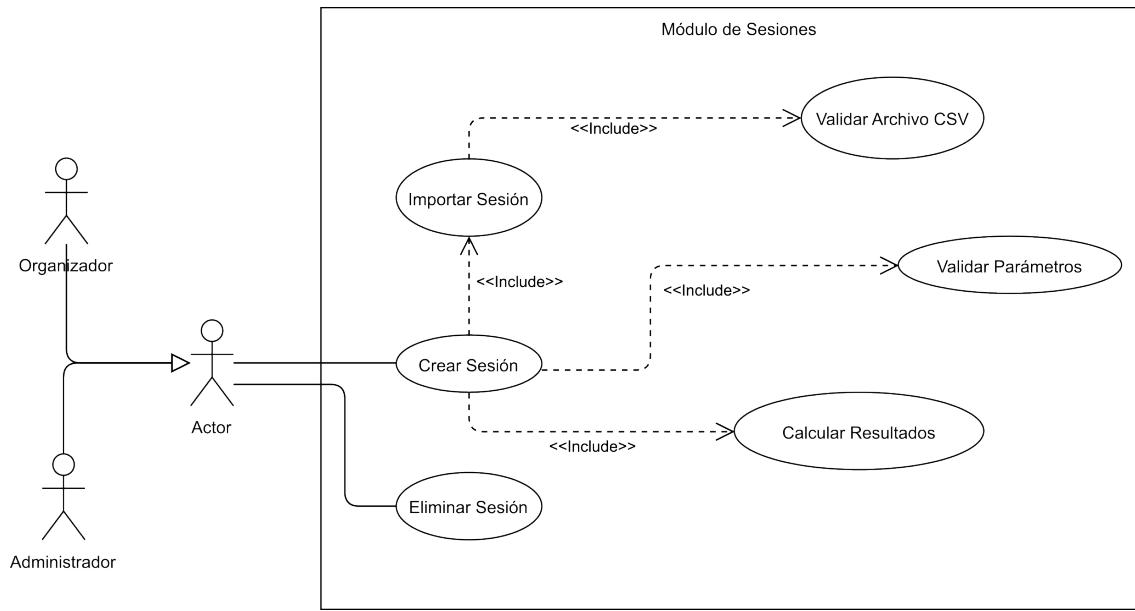


Figura 6.8: Diagrama de Casos de Uso del módulo de sesiones 1

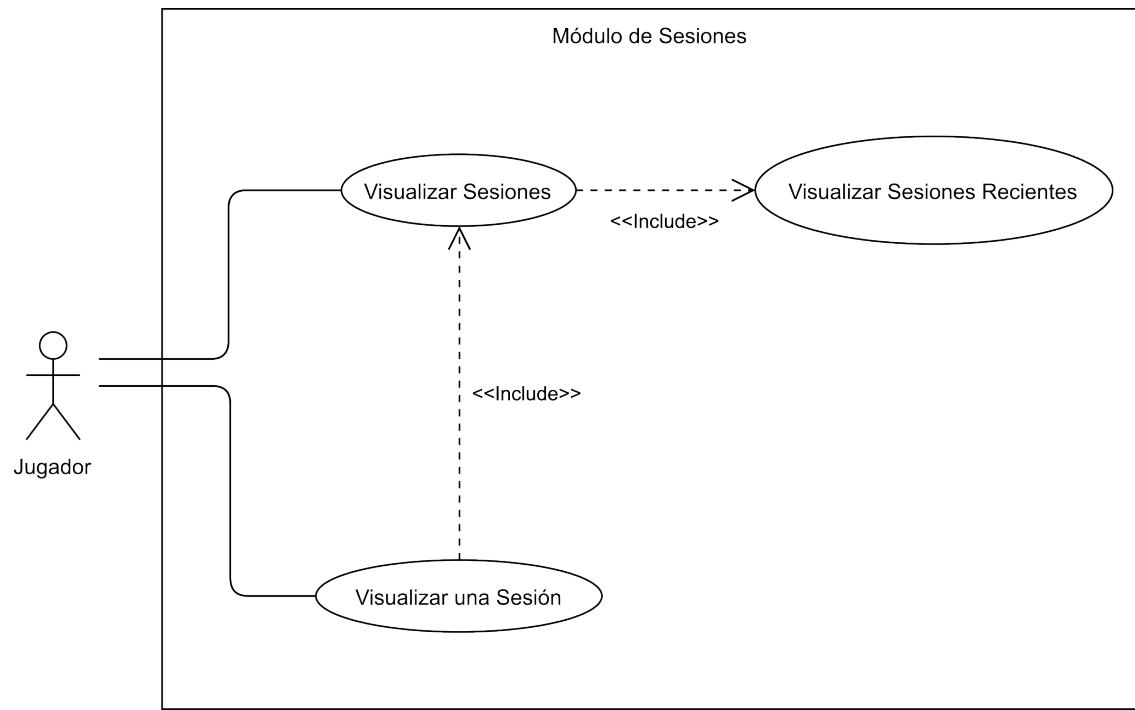


Figura 6.9: Diagrama de Casos de Uso del módulo de sesiones 1

## 6.2.2. Especificación de los Casos de Uso

En esta sección se detallan todos los casos de uso diagramados en la subsección 6.2.1. De acuerdo con la jerarquía de actores presentada en la sección 6.1, se entenderá que, si un actor de tipo "Jugador" puede realizar una acción específica, entonces cualquier otro de los actores con cargo podrá realizarla también.

A continuación, la tabla TABLENUM contiene los nombres y actores asociados a cada uno de los diagramas de casos de uso. Los casos de uso cuyos actores y nombre han sido marcados en negrita serán detallados posteriormente.

Casos de Uso		
<b>Id</b>	<b>Actor</b>	<b>Nombre</b>
<b>CU_01</b>	<b>Administrador, Moderador, Organizador y Jugador</b>	<b>Iniciar Sesión</b>
<b>CU_02</b>	<b>Administrador</b>	<b>Importar Autos</b>
<b>CU_03</b>	Administrador	Crear Auto
<b>CU_04</b>	Administrador	Editar Auto
<b>CU_05</b>	Administrador	Eliminar Auto
<b>CU_06</b>	Administrador	Validar Archivo CSV
<b>CU_07</b>	Administrador	Validar Parámetros
<b>CU_08</b>	Jugador	Visualizar Autos
<b>CU_09</b>	<b>Jugador</b>	<b>Visualizar un Auto</b>
<b>CU_10</b>	Jugador	Filtrar por Categoría
<b>CU_11</b>	<b>Administrador</b>	<b>Importar Pistas</b>
<b>CU_12</b>	Administrador	Crear Pista
<b>CU_13</b>	Administrador	Editar Pista
<b>CU_14</b>	Administrador	Eliminar Pista
<b>CU_15</b>	Administrador	Validar Archivo CSV
<b>CU_16</b>	Administrador	Validar Parámetros
<b>CU_17</b>	Jugador	Visualizar Pistas
<b>CU_18</b>	Jugador	Visualizar una Pista
<b>CU_19</b>	Jugador	Filtrar por Temporada
<b>CU_20</b>	<b>Administrador</b>	<b>Crear Temporada</b>
<b>CU_21</b>	<b>Administrador</b>	<b>Eliminar Temporada</b>
<b>CU_22</b>	Administrador	Validar Parámetros
<b>CU_23</b>	Administrador	Crear Rankings Asociados
<b>CU_24</b>	Administrador	Eliminar Rankings Asociados
<b>CU_25</b>	Administrador	Eliminar Sesiones Asociadas
<b>CU_26</b>	Jugador	Visualizar Temporadas
<b>CU_27</b>	Jugador	Visualizar una Temporada
<b>CU_28</b>	Jugador	Visualizar Ranking
<b>CU_29</b>	Jugador	Mostrar Tabla de Resultados
<b>CU_30</b>	<b>Administrador y Organizador</b>	<b>Importar Sesión</b>
<b>CU_31</b>	Administrador y Organizador	Crear Sesión
<b>CU_32</b>	<b>Administrador y Organizador</b>	<b>Eliminar Sesión</b>
<b>CU_33</b>	Administrador y Organizador	Validar Archivo CSV
<b>CU_34</b>	Administrador y Organizador	Validar Parámetros
<b>CU_35</b>	Administrador y Organizador	Calcular Resultados
<b>CU_36</b>	Jugador	Visualizar Sesiones
<b>CU_37</b>	<b>Jugador</b>	<b>Visualizar una Sesión</b>
<b>CU_38</b>	Jugador	Visualizar Sesiones Recientes

### 6.2.3. Detalle de los Casos de Uso

A continuación, se presentan tablas de detalle para cada uno de los casos de uso listados en la subsección 6.2.2 y que fueron marcados en negrita. También contarán con un detalle de su flujo de eventos básico.

Los casos de uso seleccionados para ser detallados fueron elegidos porque cumplen funciones fundamentales de los requisitos funcionales de la aplicación desarrollada. El resto de los casos de uso solamente contarán con sus precondiciones y una descripción simple.

<b>CU_01_INICIAR_SESION</b> (Usuario)	
<b>Pre-Condiciones:</b> El usuario debe estar en la página web. El usuario debe haberse registrado en la plataforma.	
<b>Post-Condiciones:</b> El usuario inicia sesión en la plataforma.	
<b>Flujo de Eventos Básicos</b>	
<b>Usuarios:</b> Administrador, Organizador, Moderador y Jugador	<b>Sistema</b>
	1. Renderiza la pantalla de inicio de sesión.
2. Ingresa su correo y contraseña, y luego pulsa el botón para iniciar sesión.	3. Valida la información ingresada por el usuario. 4. Sesión iniciada. Redirecciona al usuario a la página principal.
<b>Flujo de Eventos Alternativo</b>	
<b>Usuarios:</b> Administrador, Organizador, Moderador y Jugador	<b>Sistema</b>
	3 (b). Si las credenciales son incorrectas, el sistema muestra un mensaje de error.
	4 (b). Vuelve al paso 2 del flujo básico.

<b>CU_02_IMPORTAR_AUTOS</b> (Administrador)	
<b>Pre-Condiciones:</b> El administrador debe haber iniciado sesión con sus credenciales.	
<b>Post-Condiciones:</b> El sistema crea los autos y los guarda en la base de datos.	
<b>Flujo de Eventos Básicos</b>	
<b>Usuario:</b> Administrador	<b>Sistema</b>
	1. Renderiza el formulario de importación de autos.
2. Selecciona el archivo CSV con la información de los autos y la temporada. Luego pulsa el botón para importar.	
	3. Llama al <b>CU_06_VALIDAR_ARCHIVO_CSV</b> .
	4. El sistema crea las colecciones y las almacena en la base de datos. Se notifica éxito al usuario.
<b>Flujo de Eventos Alternativo</b>	
<b>Usuario:</b> Administrador	<b>Sistema</b>
	3 (b). Si el archivo seleccionado no es válido, el sistema muestra un mensaje de error.
	4 (b). Vuelve al paso 2 del flujo básico.

<b>CU_03_CREAR_AUTO</b> (Administrador)
<b>Pre-Condiciones:</b> El administrador debe haber iniciado sesión con sus credenciales.
<b>Descripción:</b> El sistema guarda el modelo de auto en la base de datos con los datos ingresados por el administrador.

<b>CU_04_EDITAR_AUTO</b> (Administrador)
<b>Pre-Condiciones:</b> El administrador debe haber iniciado sesión con sus credenciales.
<b>Descripción:</b> El sistema actualiza el modelo de auto en la base de datos con los datos ingresados por el administrador.

<b>CU_05_ELIMINAR_AUTO</b> (Administrador)
<b>Pre-Condiciones:</b> El administrador debe haber iniciado sesión con sus credenciales.
<b>Descripción:</b> El sistema elimina el modelo de auto en la base de datos.

<b>CU_06_VALIDAR_ARCHIVO_CSV</b>
<b>Pre-Condiciones:</b> Debe ser activado por <b>CU_02</b> .
<b>Descripción:</b> El sistema lee el archivo y verifica que éste esté en el formato adecuado, y que contenga datos válidos para crear modelos de autos.

**CU\_07\_VALIDAR\_PARAMETROS**

**Pre-Condiciones:** Debe ser activado por **CU\_03**.

**Descripción:** Verifica que los parámetros cumplan con las restricciones del modelo de auto.

**CU\_08\_VISUALIZAR\_AUTOS (Jugador)**

**Pre-Condiciones:** No tiene.

**Descripción:** El sistema renderiza los autos por pantalla.

**CU\_09\_VISUALIZAR\_UN\_AUTO (Jugador)**

**Pre-Condiciones:** No tiene.

**Post-Condiciones:** El sistema renderiza un auto por pantalla.

**Flujo de Eventos Básicos**

Usuario:	Sistema
	1. El sistema renderiza un enlace a la página del auto, ya sea una imagen o texto clickeable.
2. El usuario clickea el enlace.	
	3. El sistema renderiza una visualización del auto seleccionado por el usuario.

**CU\_10\_FILTRAR\_POR\_CATEGORIA**

**Pre-Condiciones:** Debe ser activado por **CU\_08**.

**Descripción:** El sistema verifica la categoría de los autos y la utiliza para separarlos en diferentes secciones de la plataforma.

<b>CU_11_IMPORTAR_PISTAS</b> (Administrador)	
<b>Pre-Condiciones:</b> El administrador debe haber iniciado sesión con sus credenciales.	
<b>Post-Condiciones:</b> El sistema crea las pistas y las guarda en la base de datos.	
<b>Flujo de Eventos Básicos</b>	
<b>Usuario:</b> Administrador	<b>Sistema</b>
	1. Renderiza el formulario de importación de pistas.
2. Selecciona el archivo CSV con la información de las pistas y la temporada. Luego pulsa el botón para importar.	
	3. Llama al <b>CU_15_VALIDAR_ARC-HIVO_CSV</b> .
	4. El sistema crea las colecciones y las almacena en la base de datos. Se notifica éxito al usuario.
<b>Flujo de Eventos Alternativo</b>	
<b>Usuario:</b> Administrador	<b>Sistema</b>
	3 (b). Si el archivo seleccionado no es válido, el sistema muestra un mensaje de error.
	4 (b). Vuelve al paso 2 del flujo básico.

<b>CU_12_CREAR_PISTA</b> (Administrador)	
<b>Pre-Condiciones:</b> El administrador debe haber iniciado sesión con sus credenciales.	
<b>Descripción:</b> El sistema guarda el modelo de pista en la base de datos con los datos ingresados por el administrador.	

<b>CU_13_EDITAR_PISTA</b> (Administrador)	
<b>Pre-Condiciones:</b> El administrador debe haber iniciado sesión con sus credenciales.	
<b>Descripción:</b> El sistema actualiza el modelo de pista en la base de datos con los datos ingresados por el administrador.	

<b>CU_14_ELIMINAR_PISTA</b> (Administrador)	
<b>Pre-Condiciones:</b> El administrador debe haber iniciado sesión con sus credenciales.	
<b>Descripción:</b> El sistema elimina el modelo de pista en la base de datos.	

<b>CU_15_VALIDAR_ARCHIVO_CSV</b>	
<b>Pre-Condiciones:</b> Debe ser activado por <b>CU_11</b> .	
<b>Descripción:</b> El sistema lee el archivo y verifica que éste esté en el formato adecuado, y que contenga datos válidos para crear modelos de pistas.	

**CU\_16\_VALIDAR\_PARAMETROS**

**Pre-Condiciones:** Debe ser activado por **CU\_12**.

**Descripción:** Verifica que los parámetros cumplan con las restricciones del modelo de pista.

**CU\_17\_VISUALIZAR\_PISTAS (Jugador)**

**Pre-Condiciones:** No tiene.

**Descripción:** El sistema renderiza las pistas por pantalla.

**CU\_18\_VISUALIZAR\_UNA\_PISTA (Jugador)**

**Pre-Condiciones:** No tiene.

**Post-Condiciones:** El sistema renderiza una pista por pantalla.

**Flujo de Eventos Básicos**

Usuario:	Sistema
	1. El sistema renderiza un enlace a la página de la pista, ya sea una imagen o texto clickeable.
2. El usuario clickea el enlace.	
	3. El sistema renderiza una visualización de la pista seleccionada por el usuario.

**CU\_19\_FILTRAR\_POR\_TEMPORADA**

**Pre-Condiciones:** Debe ser activado por **CU\_17**.

**Descripción:** El sistema verifica la temporada de las pistas y la utiliza para renderizar las pistas de la temporada actual.

<b>CU_20_CREAR_TEMPORADA</b> (Administrador)	
<b>Pre-Condiciones:</b> El administrador debe haber iniciado sesión con sus credenciales.	
<b>Post-Condiciones:</b> El sistema crea la temporada y sus rankings asociados.	
<b>Flujo de Eventos Básicos</b>	
<b>Usuario:</b> Administrador	<b>Sistema</b>
	1. Renderiza el formulario de creación de temporada.
2. Llena los campos del formulario de creación de temporada.	3. Valida los datos ingresados por el administrador. 4. El sistema crea la temporada. 5. El sistema crea 6 rankings y los relaciona con la temporada recién creada. 6. Temporada y rankings asociados creados. Redirecciona al administrador a la página de la temporada recién creada.
<b>Flujo de Eventos Alternativo</b>	
<b>Usuario:</b> Administrador	<b>Sistema</b>
	3 (b). Si los datos ingresados son inválidos, el sistema muestra un mensaje de error. 4 (b). Vuelve al paso 2 del flujo básico.

<b>CU_21_ELIMINAR_TEMPORADA</b> (Administrador)	
<b>Pre-Condiciones:</b> El administrador debe haber iniciado sesión con sus credenciales.	
<b>Post-Condiciones:</b> El sistema elimina la temporada, sus rankings asociados y todas las sesiones asociadas a dichos rankings.	
<b>Flujo de Eventos Básicos</b>	
<b>Usuario:</b> Administrador	<b>Sistema</b>
	1. Renderiza las opciones de administrador en la vista de temporada.
2. El administrador clica el botón de eliminar.	3. El sistema busca y elimina todas las sesiones de todos los rankings de la temporada eliminada. 4. El sistema elimina todos los rankings asociados a la temporada. 5. El sistema elimina la temporada. 6. Temporada eliminada. Redirecciona al administrador a la página de las temporadas.

<b>CU_22_VALIDAR_PARAMETROS</b>
<b>Pre-Condiciones:</b> Debe ser activado por <b>CU_20</b> .
<b>Descripción:</b> Verifica que los parámetros cumplan con las restricciones del modelo de temporada.

**CU\_23\_CREAR\_RANKINGS\_ASOCIADOS**

**Pre-Condiciones:** Debe ser activado por **CU\_20**.

**Descripción:** El sistema crea 6 modelos de ranking y los asocia a la temporada que se está creando en **CU\_20**.

**CU\_24\_ELIMINAR\_RANKINGS\_ASOCIADOS**

**Pre-Condiciones:** Debe ser activado por **CU\_21**.

**Descripción:** El sistema elimina los 6 modelos de ranking asociados a la temporada que se está eliminando en **CU\_20**.

**CU\_25\_ELIMINAR\_SESIONES\_ASOCIADAS**

**Pre-Condiciones:** Debe ser activado por **CU\_24**.

**Descripción:** El sistema elimina todos los modelos de sesión asociados con el ranking que se está eliminando en **CU\_24**.

**CU\_26\_VISUALIZAR\_TEMPORADAS (Jugador)**

**Pre-Condiciones:** No tiene.

**Descripción:** El sistema renderiza las temporadas por pantalla.

**CU\_27\_VISUALIZAR\_UNA\_TEMPORADA (Jugador)**

**Pre-Condiciones:** No tiene.

**Descripción:** El sistema renderiza una temporada por pantalla.

**CU\_28\_VISUALIZAR\_RANKING (Jugador)**

**Pre-Condiciones:** No tiene.

**Descripción:** El sistema renderiza un ranking por pantalla.

**CU\_29\_VISUALIZAR\_TABLA\_DE\_RESULTADOS (Jugador)**

**Pre-Condiciones:** Debe ser activado por **CU\_27**.

**Descripción:** El sistema renderiza una tabla de resultados asociada a la temporada.

<b>CU_30_IMPORTAR_SESION</b> (Administrador y Organizador)	
<b>Pre-Condiciones:</b> El administrador u organizador debe haber iniciado sesión con sus credenciales.	
<b>Post-Condiciones:</b> El sistema crea la sesión y la guarda en la base de datos. El sistema también calcula y suma las estadísticas a la temporada y ranking de la sesión en cuestión. De la misma forma, suma los resultados calculados a las estadísticas individuales de los jugadores que participaron en ella.	
<b>Flujo de Eventos Básicos</b>	
<b>Usuarios:</b> Administrador y Organizador	<b>Sistema</b>
	1. Renderiza el formulario de importación de sesión.
2. Selecciona el archivo CSV (Session Log), la temporada y el ranking. Luego pulsa el botón para importar.	3. Valida el archivo Session Log. Valida si los datos son correctos.
	4. Sesión creada. Redirecciona al administrador u organizador a la página de visualización de la sesión importada.
<b>Flujo de Eventos Alternativo</b>	
<b>Usuario:</b> Administrador y Organizador	<b>Sistema</b>
	3 (b). Si el archivo seleccionado no es válido, el sistema muestra un mensaje de error.
	4 (b). Vuelve al paso 2 del flujo básico.

<b>CU_31_CREAR_SESION</b> (Administrador y Organizador)	
<b>Pre-Condiciones:</b> El administrador u organizador debe haber iniciado sesión con sus credenciales.	
<b>Descripción:</b> El sistema guarda el modelo de sesión en la base de datos. El sistema calcula y suma las estadísticas de cada jugador a sus perfiles, y también a los resultados acumulados de los modelos de ranking y temporada asociados a la sesión.	

<b>CU_32_ELIMINAR_SESION</b> (Administrador y Organizador)	
<b>Pre-Condiciones:</b> El administrador u organizador debe haber iniciado sesión con sus credenciales.	
<b>Post-Condiciones:</b> El sistema elimina la sesión. El sistema sustrae las estadísticas asociadas a la sesión de los perfiles de los jugadores que participaron en ella.	
<b>Flujo de Eventos Básicos</b>	
<b>Usuario:</b> Administrador y Organizador	<b>Sistema</b>
	1. Renderiza las opciones de administrador en la vista de sesión.
2. El administrador u organizador clica el botón de eliminar.	3. El sistema calcula los resultados de la sesión.
	4. El sistema busca los perfiles de cada jugador que participó en la sesión.
	5. El sistema sustrae los resultados calculados de cada perfil de jugador, respectivamente.
	6. Sesión eliminada. Redirecciona al administrador u organizador a la página principal.

<b>CU_33_VALIDAR_ARCHIVO_CSV</b>
<b>Pre-Condiciones:</b> Debe ser activado por <b>CU_30</b> .
<b>Descripción:</b> El sistema lee el archivo y verifica que éste esté en el formato adecuado, y que contenga datos válidos para crear un modelo de sesión.

<b>CU_34_VALIDAR_PARAMETROS</b>
<b>Pre-Condiciones:</b> Debe ser activado por <b>CU_31</b> .
<b>Descripción:</b> Verifica que los parámetros cumplan con las restricciones del modelo de sesión y sus sub-modelos.

<b>CU_35_CALCULAR_RESULTADOS</b>
<b>Pre-Condiciones:</b> Debe ser activado por <b>CU_31</b> .
<b>Descripción:</b> Calcula los resultados de la sesión, transformándolos de datos crudos del Session Log a resultados en el formato de RVA.

<b>CU_36_VISUALIZAR_SESIONES</b> (Jugador)
<b>Pre-Condiciones:</b> No tiene.
<b>Descripción:</b> El sistema renderiza las sesiones por pantalla.

<b>CU_37_VISUALIZAR_UNA_SESION</b> (Jugador)	
<b>Pre-Condiciones:</b> No tiene.	
<b>Post-Condiciones:</b> El sistema renderiza una sesión por pantalla.	
<b>Flujo de Eventos Básicos</b>	
<b>Usuario:</b> Jugador	<b>Sistema</b>
	1. El sistema renderiza un enlace a la página de una sesión.
2. El usuario clickea el enlace.	
	3. El sistema calcula los resultados de la sesión y los transforma en resultados en el formato de RVA.
	3. El sistema renderiza la sesión por pantalla.

<b>CU_38_VISUALIZAR_SESIONES_RECIENTES</b> (Jugador)
<b>Pre-Condiciones:</b> Debe ser activado por <b>CU_36</b> .
<b>Descripción:</b> El sistema las últimas sesiones, buscando la temporada más reciente por fecha, y luego buscando la última sesión del ranking más alto posible (primero al sexto).

### 6.3. Modelo de Datos

Los siguientes diagramas contienen información relevante de los modelos de datos utilizados para desarrollar la aplicación de este proyecto de título. Algunos de ellos se muestran por separado para mayor claridad.

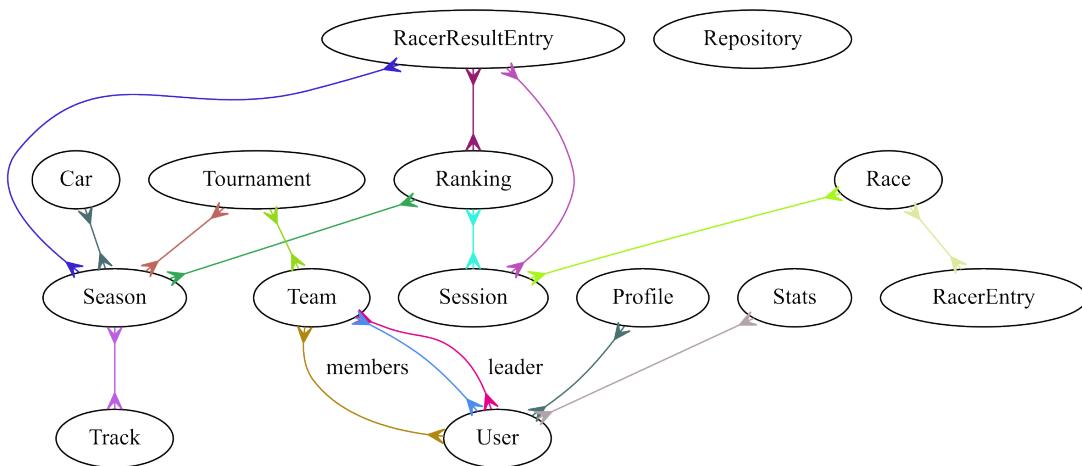


Figura 6.10: Resumen de modelos de datos

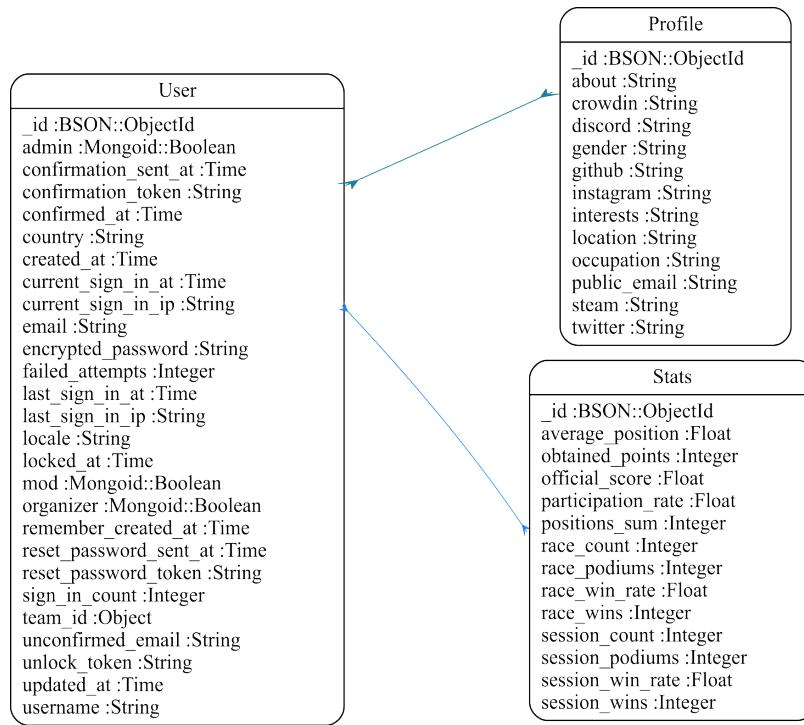


Figura 6.11: Modelo de relaciones entre usuario, perfil y estadísticas

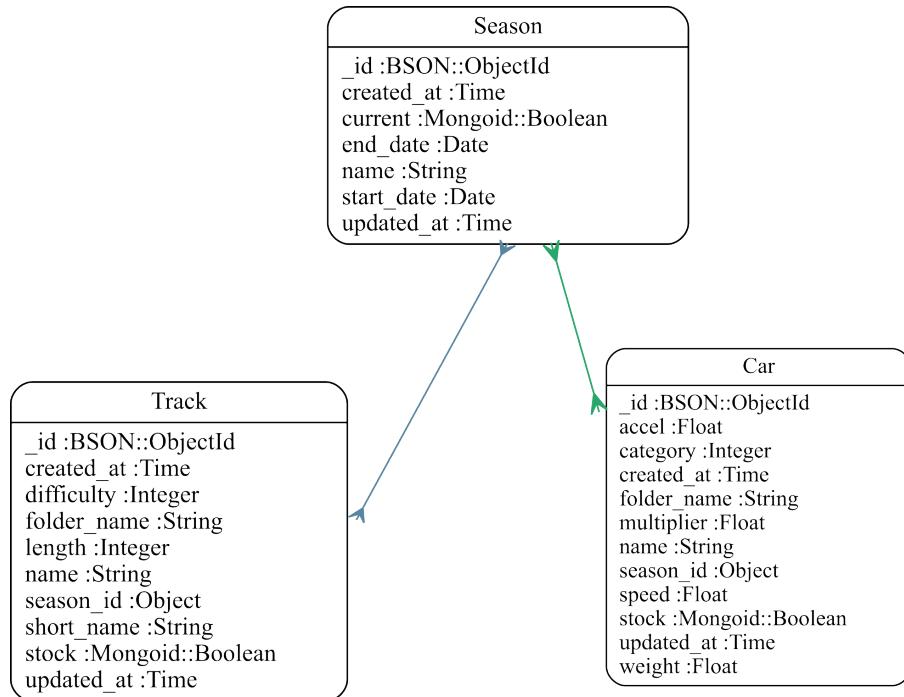


Figura 6.12: Modelo de relación entre temporada, autos y pistas

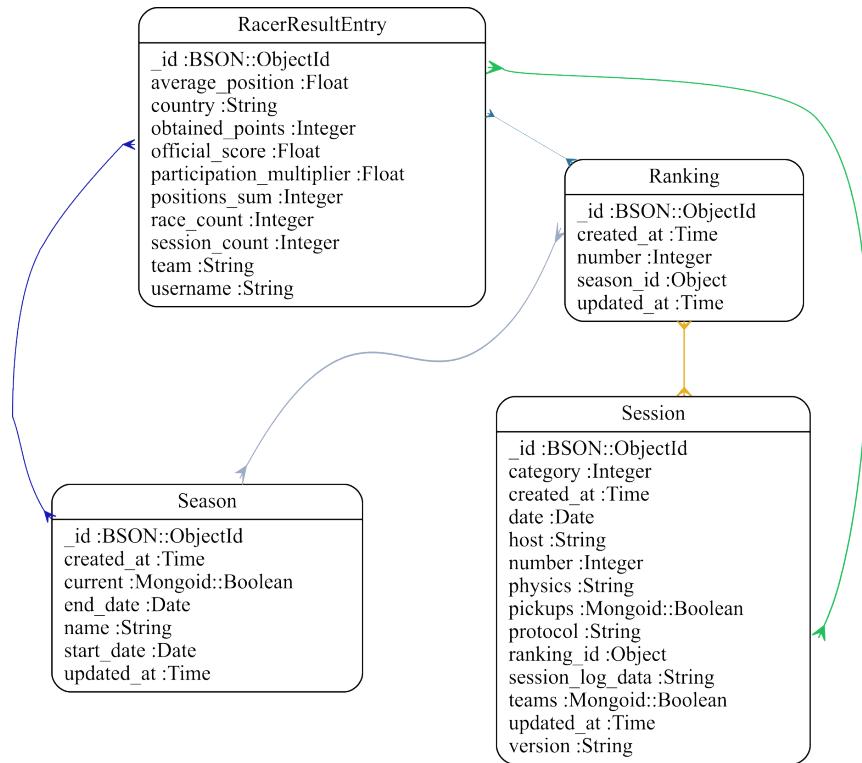


Figura 6.13: Modelo de relaciones entre temporada, ranking y sesiones con sus componentes

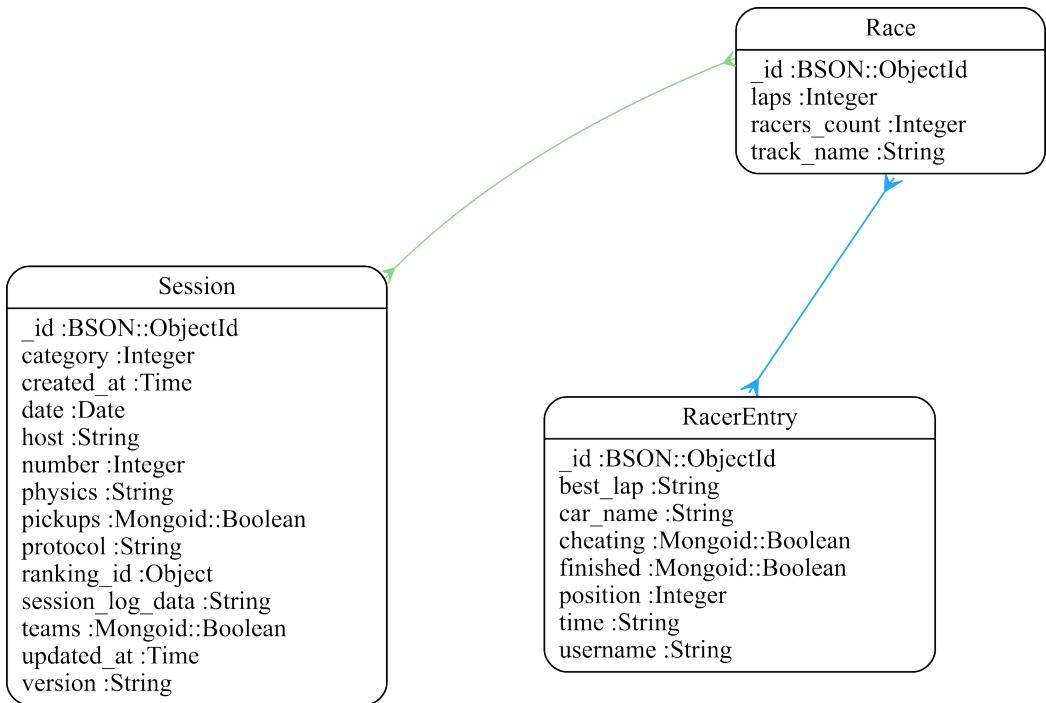


Figura 6.14: Modelo de relación entre sesión, carrera y entrada de corredor

## 6.4. Esquema de la Base de Datos

Todas las bases de datos dentro de MongoDB están prefijadas utilizando el término "rv". (Re-Volt) Por ejemplo, la base de datos que almacena las colecciones de autos se llama "rv\_cars", la de los usuarios "rv\_users", etc.

A continuación, se exponen fragmentos de código de Ruby con los modelos definidos para este proyecto. Los fragmentos de código están en formato de **mongoid** (librería de Ruby para MongoDB). Este formato se compone de la declaración de los campos de los modelos, especificando tipo y otras restricciones asociadas. Además, se mezcla el concepto de relaciones entre modelos de Ruby on Rails.

Para facilitar la comprensión de los esquemas de bases de datos a continuación, la siguiente tabla contiene una pequeña especificación de los atributos utilizados para definir los modelos.

Atributo	Detalle
<b>store.in</b>	Declara el nombre de la base de datos de mongo en donde los documentos de este modelo serán almacenados.
<b>field</b>	Declara el nombre de un campo del modelo.
<b>embeds_many</b>	Declara que el modelo contendrá una colección de modelos anidados.
<b>accepts_nested_attributes_for</b>	Declara que el modelo aceptará atributos para los modelos anidados que éste contenga.
<b>validates_presence_of</b>	Valida si el campo proveído no está vacío al momento de guardar el modelo en la base de datos.
<b>belongs_to</b>	Declara que el modelo tendrá una referencia por ID a otro modelo, quedando esta registrada como un atributo.
<b>has_many</b>	Declara que el modelo podrá ser referenciado por ID desde varios otros modelos.
<b>has_one</b>	Declara que el modelo contará con una referencia singular por ID a otro modelo.
<b>validates</b>	Declara una o más validaciones para un campo del modelo, tales como su presencia garantizada, singularidad, tamaño mínimo o máximo, entre otras.

```

class Season
  include Mongoid::Document
  include Mongoid::Timestamps
  include Mongoid::MultiParameterAttributes

  store_in :database => 'rv_seasons'

  has_many :rankings
  has_many :cars
  has_many :tracks

  embeds_many :racer_result_entries

  accepts_nested_attributes_for :racer_result_entries

  field :name, :type => String
  field :start_date, :type => Date
  field :end_date, :type => Date
  field :current, :type => Boolean, :default => false

  validates_presence_of :name
  validates_presence_of :start_date
  validates_presence_of :current

  validates_uniqueness_of :name
end

```

Listing 5: Representación en código del modelo de Season de RVA.

```
class Ranking
  include Mongoid::Document
  include Mongoid::Timestamps

  store_in :database => 'rv_rankings'

  belongs_to :season
  has_many :sessions

  embeds_many :racer_result_entries

  accepts_nested_attributes_for :racer_result_entries

  field :number, :type => Integer

  validates_presence_of :number
end
```

Listing 6: Representación en código del modelo de Ranking de RVA.

```

class Session
  include Mongoid::Document
  include SessionLogUploader::Attachment(:session_log)
  include Mongoid::Timestamps

  store_in :database => 'rv_sessions'

  belongs_to :ranking
  embeds_many :races
  embeds_many :racer_result_entries

  accepts_nested_attributes_for :racer_result_entries

  field :number, :type => Integer
  field :host, :type => String
  field :version, :type => String
  field :physics, :type => String
  field :protocol, :type => String
  field :pickups, :type => Boolean
  field :date, :type => Date
  field :teams, :type => Boolean
  field :category, :type => Integer
  field :session_log_data, :type => String

  validates_presence_of :number
  validates_presence_of :host
  validates_presence_of :version
  validates_presence_of :physics
  validates_presence_of :protocol
  validates_presence_of :date
  validates_presence_of :session_log
  validates_presence_of :teams
  validates_presence_of :category
end

```

Listing 7: Representación en código del modelo de Session de RVA.

```

class User
  include Mongoid::Document
  include Mongoid::Timestamps

  store_in :database => 'rv_users'

  USERNAME_REGEX = /\A([A-Z0-9_]{1,16}|[0-9a-f]{24})\z/

  validates :email, :presence => true, :uniqueness => true
  validates :username, :presence => true,
    :uniqueness => true,
    :length => { :minimum => 2, :maximum => 16 }
  validates_format_of :username, :with => USERNAME_REGEX

  belongs_to :team, :optional => true
  has_one :team

  embeds_one :profile
  embeds_one :stats
  accepts_nested_attributes_for(
    :profile,
    :update_only => true,
    :allow_destroy => false
  )
  accepts_nested_attributes_for(:stats,
    :update_only => true,
    :allow_destroy => false
  )

  before_create :create_profile
  before_create :create_stats

  field :admin, :type => Boolean, :default => false
  field :mod, :type => Boolean, :default => false
  field :organizer, :type => Boolean, :default => false
  field :locale, :type => String, :default => 'en_us'
  field :country, :type => String
end

```

Listing 8: Representación en código del modelo de Session de RVA.

```
class Race
  include Mongoid::Document
  include Mongoid::Attributes::Dynamic

  embedded_in :session

  embeds_many :racer_entries

  field :track_name, :type => String
  field :laps, :type => Integer
  field :racers_count, :type => Integer

  validates_presence_of :track_name
  validates_presence_of :laps
  validates_presence_of :racers_count
end
```

Listing 9: Representación en código del modelo Profile.

```
class RacerEntry
  include Mongoid::Document
  include Mongoid::Attributes::Dynamic

  embedded_in :race

  field :car_name, :type => String
  field :position, :type => Integer
  field :username, :type => String
  field :time, :type => String
  field :best_lap, :type => String
  field :finished, :type => Mongoid::Boolean
  field :cheating, :type => Mongoid::Boolean

  validates_presence_of :car_name
  validates_presence_of :position
  validates_presence_of :username
  validates_presence_of :time
  validates_presence_of :best_lap
  validates_presence_of :finished
  validates_presence_of :cheating
end
```

Listing 10: Representación en código del modelo Profile.

```

class Profile
  include Mongoid::Document

  embedded_in :user

  field :about, :type => String
  field :gender, :type => String
  field :public_email, :type => String
  field :location, :type => String
  field :discord, :type => String
  field :github, :type => String
  field :instagram, :type => String
  field :crowdin, :type => String
  field :steam, :type => String
  field :twitter, :type => String
  field :occupation, :type => String
  field :interests, :type => String
end

```

Listing 11: Representación en código del modelo Profile.

```

class Stats
  include Mongoid::Document

  embedded_in :user

  field :race_wins, :type => Integer
  field :race_win_rate, :type => Float
  field :race_podiums, :type => Integer
  field :race_count, :type => Integer
  field :positions_sum, :type => Integer
  field :session_wins, :type => Integer
  field :session_win_rate, :type => Float
  field :session_podiums, :type => Integer
  field :session_count, :type => Integer
  field :average_position, :type => Float
  field :participation_rate, :type => Float
  field :official_score, :type => Float
  field :obtained_points, :type => Integer
end

```

Listing 12: Representación en código del modelo Stats.

```
class Car
  include Mongoid::Document
  include Mongoid::Timestamps

  store_in :database => 'rv_cars'

  belongs_to :season

  field :name, :type => String
  field :speed, :type => Float
  field :accel, :type => Float
  field :weight, :type => Float
  field :multiplier, :type => Float
  field :folder_name, :type => String
  field :category, :type => Integer
  field :stock, :type => Boolean, :default => false

  validates_presence_of :name
  validates_presence_of :speed
  validates_presence_of :accel
  validates_presence_of :weight
  validates_presence_of :multiplier
  validates_presence_of :folder_name
  validates_presence_of :category
  validates_presence_of :stock
end
```

Listing 13: Representación en código del modelo Car.

```
class Track
  include Mongoid::Document
  include Mongoid::Timestamps

  store_in :database => 'rv_tracks'

  belongs_to :season

  field :name, :type => String
  field :short_name, :type => String
  field :difficulty, :type => Integer
  field :length, :type => Integer
  field :folder_name, :type => String
  field :stock, :type => Boolean, :default => false

  validates_presence_of :name
  validates_presence_of :short_name
  validates_presence_of :difficulty
  validates_presence_of :length
  validates_presence_of :folder_name
  validates_presence_of :stock
end
```

Listing 14: Representación en código del modelo Track.

```
class RacerResultEntry
  include Mongoid::Document

  embedded_in :season
  embedded_in :ranking
  embedded_in :session

  field :username, :type => String
  field :country, :type => String
  field :session_count, :type => Integer
  field :race_count, :type => Integer
  field :positions_sum, :type => Integer
  field :average_position, :type => Float
  field :obtained_points, :type => Integer
  field :official_score, :type => Float
  field :participation_multiplier, :type => Float
  field :team, :type => String

  validates_presence_of :username
  validates_presence_of :country
end
```

Listing 15: Representación en código del modelo Track.

## 6.5. Diseño de Interfaz

El diseño de las interfaces de Re-Volt America ha sido debidamente cuidado al ser implementado en aplicación desarrollada.

En primer lugar, contamos con una página de inicio con el título de la comunidad de Re-Volt America, junto con varias imágenes del juego para que este pueda ser inmediatamente reconocido por quien visite el portal.

En la parte superior, se encuentra la barra de navegación principal, la cual incluye, de izquierda a derecha:

- El logo de RVA.
- Botones para acceder a las secciones más importantes de la web.
- Un botón para poder registrarse, iniciar sesión o visualizar información del perfil en caso de existir una sesión iniciada.

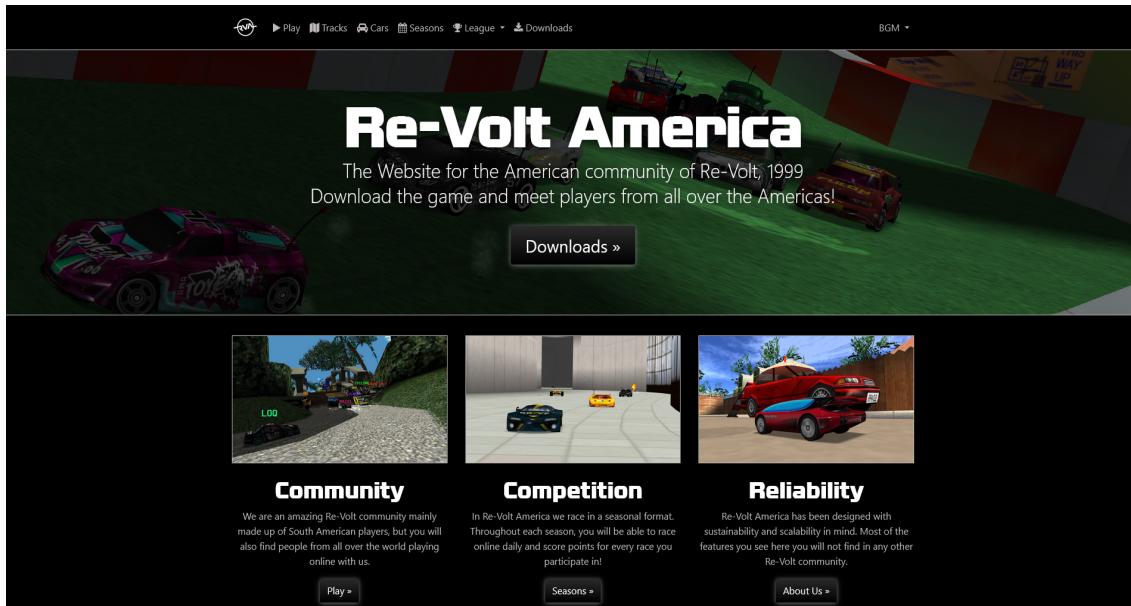


Figura 6.15: Landing page de RVA

Dentro de las secciones de autos, pistas y sesiones se cuenta con una barra de sub-navegación, la cual integra todos los elementos de una temporada, dando acceso al usuario de manera más práctica a toda la información que este podría necesitar.



Figura 6.16: Página de autos de RVA

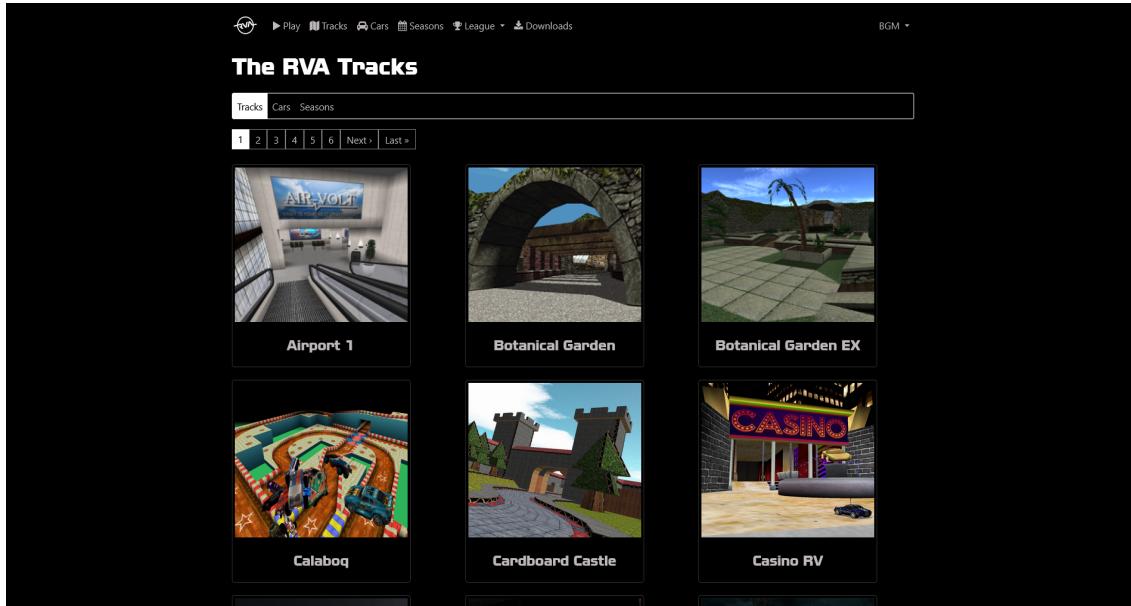


Figura 6.17: Página de pistas de RVA

El diseño de la interfaz de las sesiones respeta fielmente el formato original de RVA.

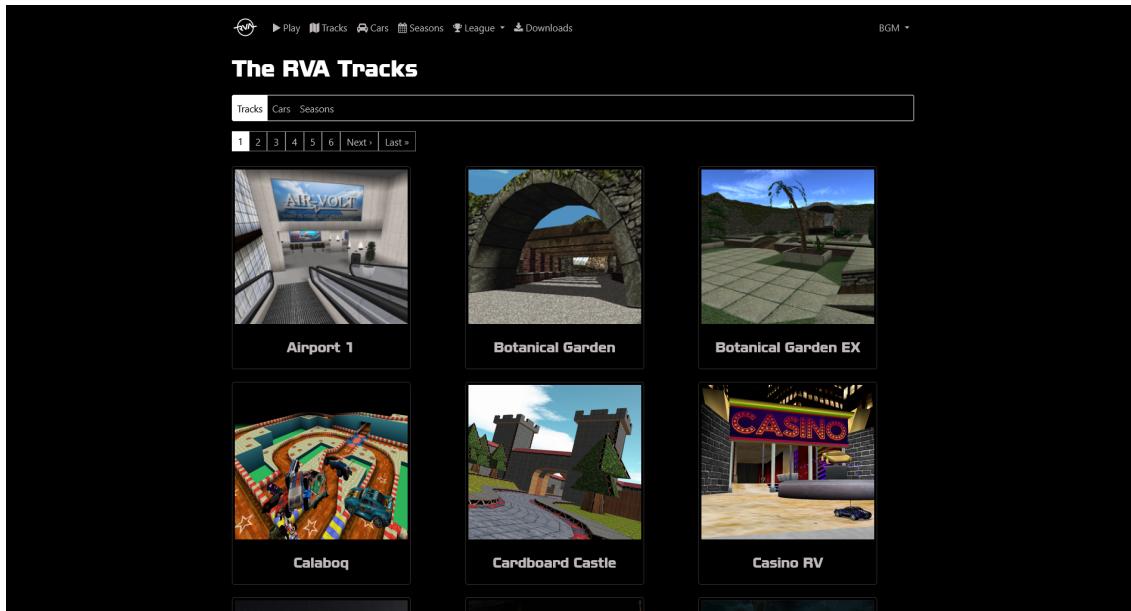


Figura 6.18: Página de sesión de RVA

La el diseño de la página de perfil de los usuarios cuenta con su nombre en grande, un resumen de sus estadísticas, posición en el ranking y un pequeño historial de las últimas sesiones en las que han participado.

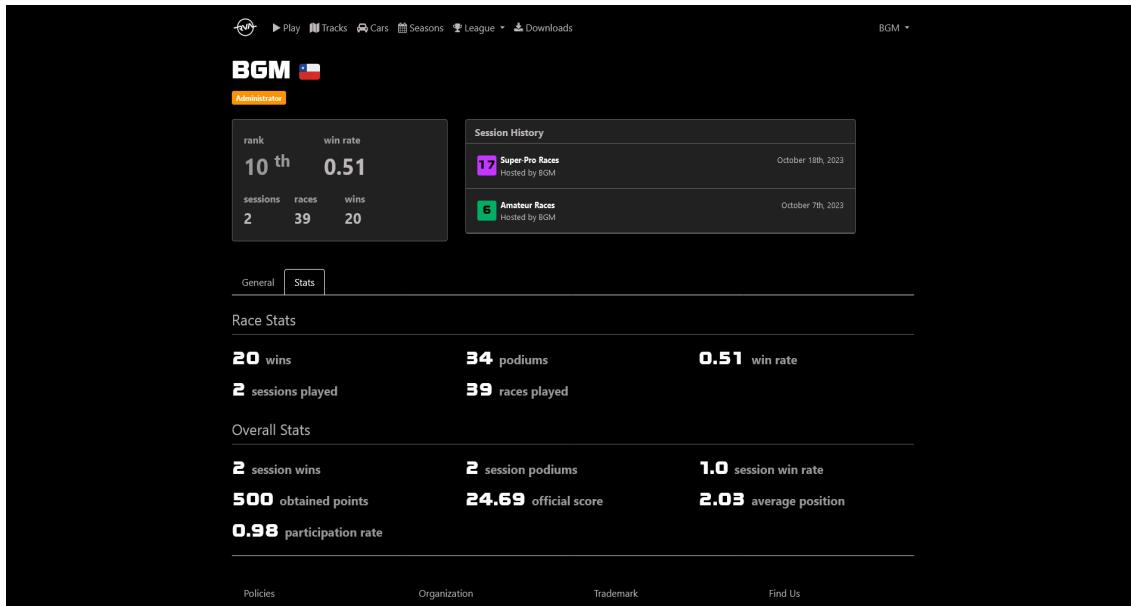


Figura 6.19: Página de perfil de usuario de RVA

Por último, la página de estadísticas sigue un formato simple de tabla de resultados, con un selector que permite filtrar dichos resultados según diversos criterios.

**Global Stats**

Sorted by: Wins

#	Country	Racer	Wins	Podiums	Win Ratio	Average Pos.	Race Count	Obtained Points	Official Score
1	YUN	90	197	0.29	3.35	315	3372	129.71	
2	MACACOSKY	52	131	0.2	3.83	255	2697	79.78	
3	ROG	24	94	0.08	5.24	308	2075	45.94	
4	LOQ	21	58	0.08	5.96	253	1845	34.09	
5	ESTEBANMZ	21	76	0.12	4.41	177	1481	37.26	
6	BGM	20	34	0.51	2.03	39	500	24.69	
7	GFORCE	19	36	0.25	4.1	77	1093	30.16	
8	WALLIIGIMB	17	61	0.07	5.68	242	2236	41.17	
9	CEMANUEL	16	69	0.08	4.82	193	1727	38.44	
10	MIGHTY	11	45	0.08	5.67	143	1088	21.47	
11	BURDANG	10	30	0.09	5.83	106	634	13.96	

Figura 6.20: Página de estadísticas globales de RVA

### 6.5.1. Paleta de Colores y Tipografía

Como preámbulo principal en cuanto a la paleta de colores y tipografía de RVA, debemos recordar que esta comunidad nace de un grupo de jugadores de Re-Volt, no de una empresa o compañía con estándares definidos. Por lo tanto, para comprender los orígenes de estos elementos, debemos remontarnos a los inicios de Re-Volt America por el año 2017.

Durante los inicios de RVA, se diseña su logo principal, el cual puede apreciarse en la figura 6.21.



Figura 6.21: Logotipo de RVA

Del logo de RVA, se derivan los colores blanco y negro (absolutos), los cuales son utilizados como paleta principal para la comunidad.

Así como los colores, la comunidad ha utilizado una fuente muy similar a la empleada en el logotipo a lo largo de los años. Esta fuente se titula "Nissan Opti", desarrollada originalmente por Castcraft Software. La caracterización de esta compañía tipográfica puede verse a continuación en la figura 6.22.



Figura 6.22: Arte de Castcraft Software

De la misma forma, nacen las paletas de colores para las tablas de resultados de sesiones y rankings acumulados. Estas últimas pueden apreciarse a continuación.

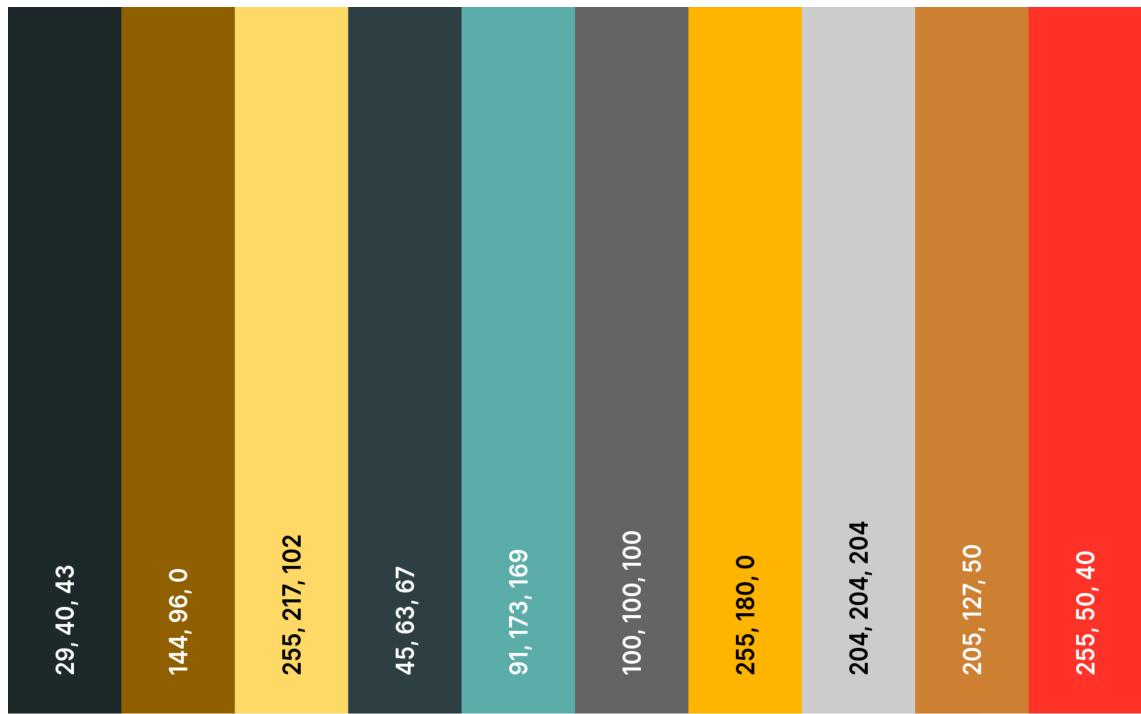
Figura 6.23: Tabla de resultados de sesión de RVA



		April	Ultima fecha jugada:				1
	Corredor	PP	PA	MP	PO	CC	FJ
1º		GForce	2,150	239	1,000	11,12	20 1
2º		Yun	2,850	188	1,000	6,60	20 1
3º		Iurac	3,750	125	1,000	3,33	20 1
4º		Loq	4,750	136	1,000	2,86	20 1
5º		Rog_97	4,450	100	1,000	2,25	20 1
6º		Burdang	5,800	96	1,000	1,66	20 1
7º		Tiorotti	5,240	90	0,850	1,46	17 1
8º		Night	8,150	30	1,000	0,37	20 1
9º		Zon	6,000	30	0,400	0,20	8 1
10º		Antelocazo	5,000	10	0,100	0,02	2 1

Figura 6.24: Tabla de ranking de RVA

La paleta de colores de las tablas de resultados, junto a su definición en RGB (Red, Green, Blue), puede encontrarse en la figura 6.25 a continuación.



RVA Results Table Colour Palette

Figura 6.25: Paleta de colores de sesiones de RVA

## 6.6. Diseño de Arquitectura

El proyecto, en su estado actual, hace uso de un servidor propio, el cual contiene los servicios web, bases de datos y caché. Todo en una sola máquina. Independientemente de donde se termine alojando, la aplicación web estará disponible en "https://rva.lat/".

Además de esto, la planificación contempla dos servicios externos, que actualmente son proveídos por GitHub pages, los cuales sirven como repositorios de almacenamiento de datos masivos. Dichos repositorios se encargan actualmente de servir información y assets como las imágenes de las pistas y autos que la web ofrece a los usuarios:

- <https://tracks.rva.lat/>: Repositorio de pistas de RVA.
- <https://cars.rva.lat/>: Repositorio de autos de RVA.

A continuación, se muestra un diagrama que ilustra todo el proceso de interacción entre servicios y usuarios:

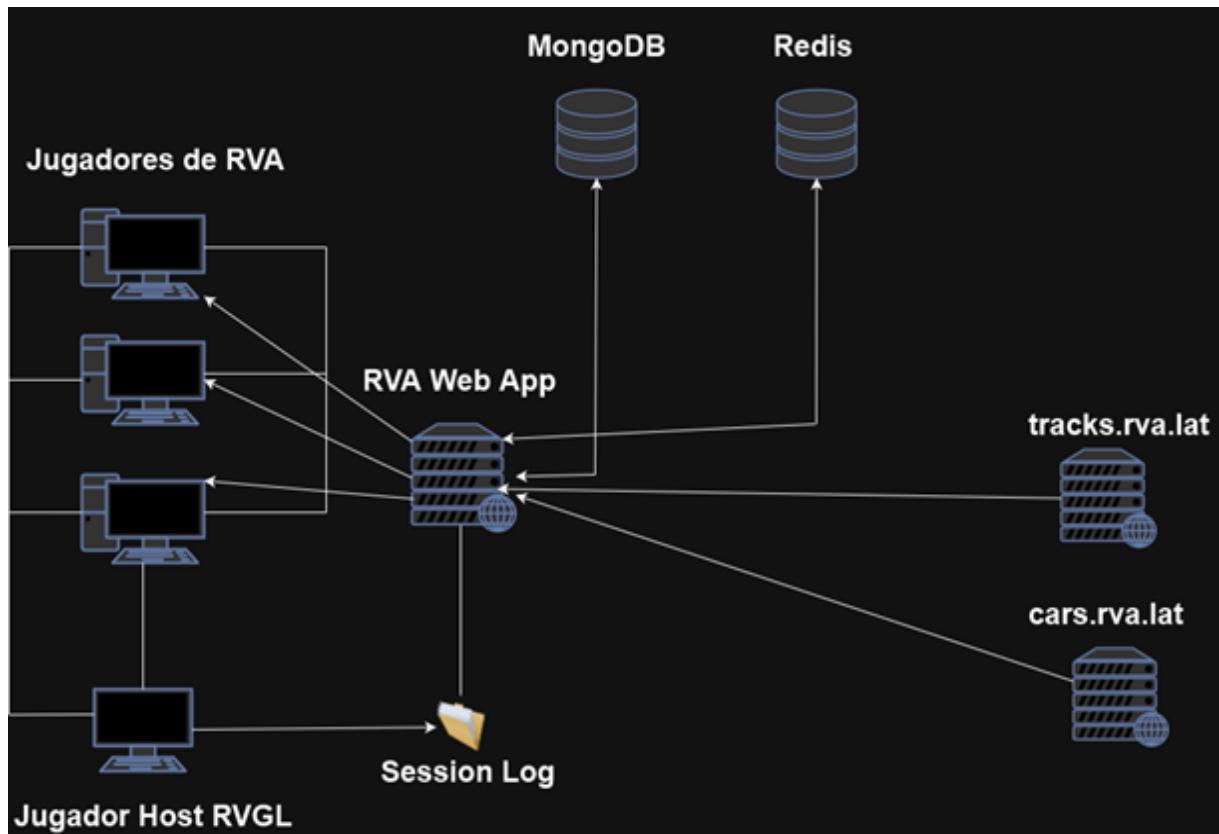


Figura 6.26: Diagrama de arquitectura de RVA

Tal como se puede apreciar en la ilustración anterior, la arquitectura que da soporte a la aplicación web de RVA se concentra en un servidor, con dos almacenes de datos. Podemos ver que los usuarios juegan la sesión, el host de la sesión sube el Session Log a la web, y los usuarios pueden visitar la misma web para revisar los resultados.

## 6.7. Estructura del Código

El proyecto, al ser una aplicación hecha en el framework de Ruby on Rails, sigue el patrón de MVC (Model View Controller) o modelo, vista, controlador. El árbol de directorios del proyecto se ve de la siguiente manera.

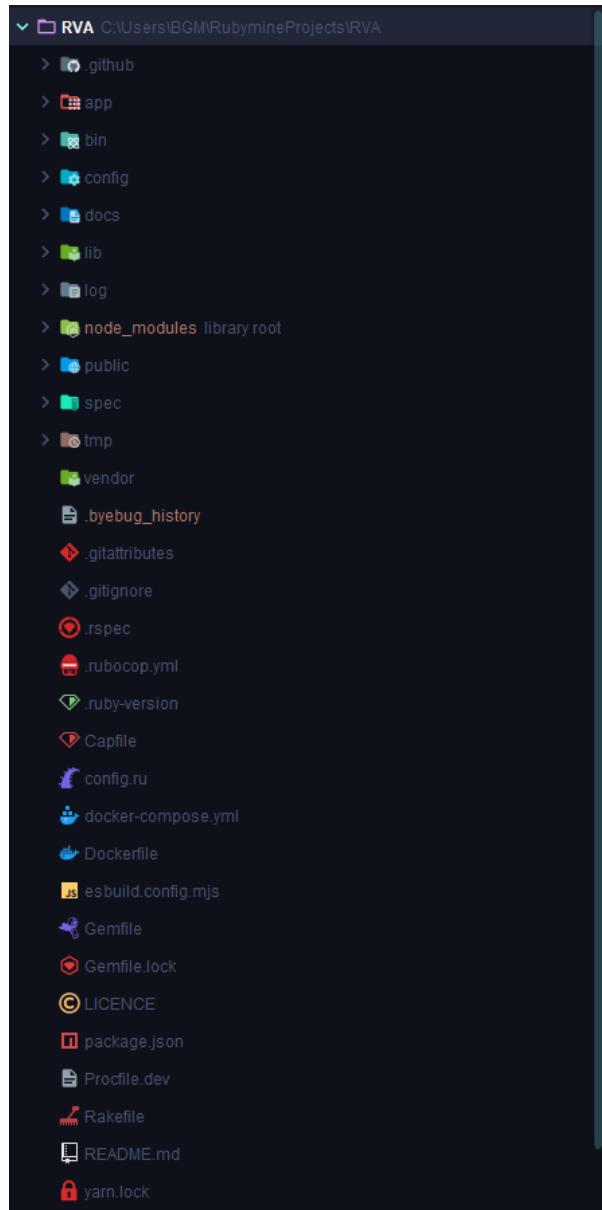


Figura 6.27: Árbol de directorios del proyecto

Dentro los archivos relevantes para la estructura del código, tenemos aquellos archivos en los que se definen dependencias y módulos de despliegue.

Archivo	Detalle
<b>Capfile</b>	Contiene la carga de módulos para el despliegue de la aplicación.
<b>Gemfile</b>	Contiene todas las dependencias del proyecto, con sus respectivas versiones.

### 6.7.1. Estándares de Codificación

El código de este proyecto sigue las recomendaciones estándar para la programación con Ruby, y otros estándares de manera consistente:

- Toda la base de código y documentación debe estar escrita en inglés.
- En general, todo el proyecto sigue las convenciones de nombrado y otros lineamientos útiles del sitio de RubyStyle.
- Se utilizan linebreaks (EOL - End of Line) CRLF.
- Se utilizan dos espacios para la indentación del código, no tabulaciones.
- La codificación del proyecto es en UTF-8.

### 6.7.2. Caché con Redis

Las llaves o entradas del caché de Redis son nombradas en minúscula, y los espacios que pudieran existir en ellas son reemplazados por guiones bajos. En caso de llaves compuestas, se utiliza el siguiente formato:

- object\_type:id
- object\_type:id#field
- object\_type:id#embedded\_object:id
- object\_type:id#embedded\_object:id#field

```

1 def show
2   require 'rva_calculate_results_service'
3
4   @count = 0
5   @rva_results = Rails.cache.fetch("Session:#{@session.id}") do
6     RvaCalculateResultsService.new(@session).call
7   end
8
9   respond_with @session do |format|
10    format.json { render :layout => false }
11  end
12 end

```

Listing 16: Implementación de caché con Redis en Rails (*sessions\_controller.rb*)

### 6.7.3. Backend

Directorio	Detalle
<b>controllers</b>	Contiene todos los controladores de Rails, los cuales manejan todas las peticiones web del usuario.
<b>helpers</b>	Contiene todas las clases utilitarias, las cuales pueden ser utilizadas para asistir a las clases de modelos, vistas y controladores.
<b>javascript</b>	Todo el código de JavaScript utilizado por la aplicación a nivel web se almacena en este directorio.
<b>models</b>	Contiene todas las clases que modelan y envuelven los datos almacenados en la base de datos de la aplicación.
<b>services</b>	Contiene todas las clases de servicio de la aplicación. Las clases de servicio son aquellos procesos complejos compuestos de mucho código, los cuales se abstraen en servicios para no polucionar los controladores que los requieren.
<b>uploaders</b>	Contiene todas las clases relacionadas con la subida de archivos.
<b>config</b>	Contiene toda la configuración de la aplicación, como las rutas, inicializadores de librerías, configuración de despliegue, declaración de los entornos de Rails (development, production), configuración de la base de datos y archivos de idioma para la localización del proyecto.

### 6.7.4. Frontend

Directorio	Detalle
<b>views</b>	Contiene todas las vistas de la aplicación. Todo lo que el usuario ve en la aplicación web tiene su archivo de vista que lo representa.
<b>assets</b>	Todas las imágenes, hojas de estilo y demás artefactos requeridos por la aplicación para su correcto funcionamiento y diseño.
<b>public</b>	Todos los archivos estáticos servidos por la página pueden encontrarse aquí. Cosas como las páginas de error (404, 422, 500, etc.), estilos compilados, entre otros.

# **Capítulo 7**

## **Plan de Capacitación, Implantación y Puesta en Marcha**

### **7.0.1. Estado del Proyecto**

Actualmente, el proyecto se encuentra finalizado. Esto quiere decir que todos los módulos, requerimientos funcionales y objetivos propuestos han sido alcanzados con éxito.

Si bien el software del proyecto está completamente finalizado, éste aún no pasa a ser utilizado por el público, por lo que sigue estando en una etapa de puesta en marcha cerrada sólo para la administración y miembros del staff de RVA.

### **7.0.2. Implantación y Puesta en Marcha**

Como se mencionó anteriormente en el estado del proyecto, la implantación del software está tomando un ruta gradual, en la que primero se le ha permitido sólo al staff de RVA interactuar con la aplicación web para que puedan familiarizarse con el nuevo sistema.

Eventualmente, para la puesta en marcha oficial, la página será lanzada para el público general.

### **7.0.3. Plan de Capacitación**

Para este proyecto, se ha considerado un plan de capacitación que consiste en un video explicativo simple, en el cual se indica como utilizar la plataforma en tiempo real, entrando en detalle en cada una de las funcionalidades relevantes del software.

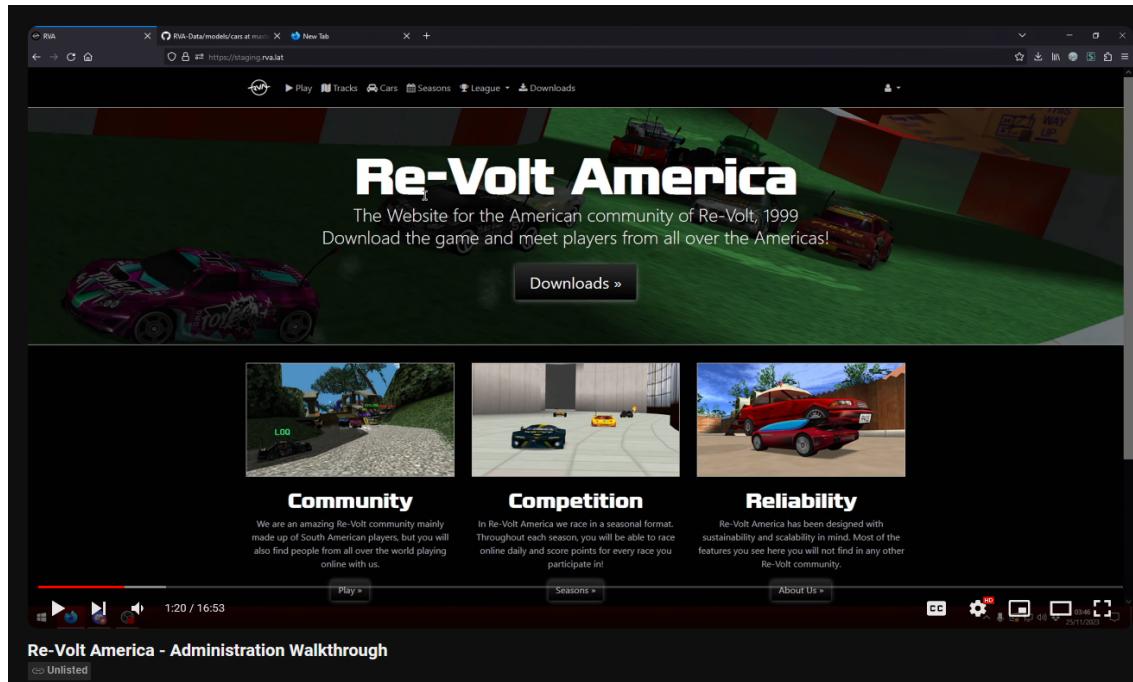


Figura 7.1: Video de capacitación para administradores

# Capítulo 8

## Conclusión del Proyecto

Para finalizar este informe, a continuación, se plantean una serie de conclusiones a las que se llegó luego de llevar a cabo el desarrollo e implementación del software:

- En relación con elaborar una propuesta que consiguiera atender las necesidades y problemas de los usuarios de Re-Volt America, con respecto al almacenamiento y visualización de resultados de partidas online, podemos decir que se logró cumplir dicho objetivo gracias al software desarrollado.
- De acuerdo con el objetivo que hablaba de diseñar una solución de software de procesamiento de datos de sesiones multijugador oficiales de las sesiones de carreras en línea, además de estadísticas personales, podemos decir que es un objetivo que fue cumplido completamente ya que, tanto el procesamiento de datos como las estadísticas por usuario fueron exitosamente implementadas.
- En relación con el objetivo de implementar una aplicación web que permitiera a los organizadores de sesiones de carreras en línea subir y publica los resultados de dichas carreras podemos decir que, en conclusión, dicho objetivo fue cumplido con éxito, ya que el software, incluso en su estado actual, ya puede realizar la importación de Session Logs, el cálculo y procesamiento de resultados en el formato de RVA, y finalmente mostrar dichos resultados por pantalla a quien los solicite desde la web.
- El proyecto ha permitido demostrar las competencias que se esperan de un ingeniero de ejecución en computación e informática, ya que se han aplicado la identificación de necesidades, análisis y el diseño de soluciones informáticas para Re-Volt America, logrando desarrollar una solución que le permite a la comunidad tener un mejor manejo de sus procesos internos, registros de datos más fiables y una experiencia de usuario mejorada en comparación con su sistema original.
- El proyecto ha sido realmente enriquecedor desde el punto de vista del desarrollo de software puesto que, dentro del área en que fue realizado, fue posible aplicar diversas tecnologías, técnicas de diseño de software, despliegue de aplicaciones e implementación de estándares de calidad, todo en un mismo contexto que cierra

de manera redonda el ciclo de desarrollo que se espera pueda ser alcanzado por un ingeniero de software.

# Capítulo 9

## Anexos

### 9.1. Anexo Estimación de Casos de Uso

Technical Complexity Factor (TCF)			
Technical Factor	Multiplier	Relevancia Percibida	Resultado Multiplicación
Distributed System	2	2	4
Application performance objectives, in either response or throughput	1	2	2
End-user efficiency (on-line)	1	3	3
Complex internal processing	1	3	3
Reusability, the code must be able to reuse in other applications	1	3	3
Installation ease	0,5	1	0,5
Operational ease, usability	0,5	2	1
Portability	2	3	6
Changeability	1	3	3
Concurrency	1	2	2
Special security features	1	3	3
Provide direct access for third parties	1	0	0
Special user training facilities	1,5	4	6

Environment Factors (EF)			
Environmental Factor	Multiplier	Relevancia Percibida	Resultado Multiplicación
Familiar with Iterative Methods	0,5	5	2,5
Application experience	1	5	5
Object Oriented experience	0,5	5	2,5
Analyst capability	1	5	5
Motivation	2	5	10
Stable requirements	-1	0	0
Difficult programming language	-1	3	-3

## 9.2. Anexos de Recopilación de Información

Para la recopilación de información de este proyecto, se realizaron varias reuniones en modalidad remota con la administración de RVA. En las 3 reuniones realizadas, se le hicieron preguntas a la administración tanto para conocer su opinión acerca de la facilidad de uso del software que estaba en desarrollo, como para que tuvieran la oportunidad de sugerir cambios o mejoras de interfaz o usabilidad del sitio.

Además de las reuniones por llamada, también se creó un chat de texto en línea con los administradores, organizadores y moderadores que contaban con el tiempo para poder participar de él. En este chat, continuarían entregando sus opiniones a lo largo del desarrollo del software propuesto, mientras que también se les mostraban avances del proyecto.

## 9.3. Anexo Aspectos de Gestión de Proyectos

### 9.3.1. Anexo Carta Gantt

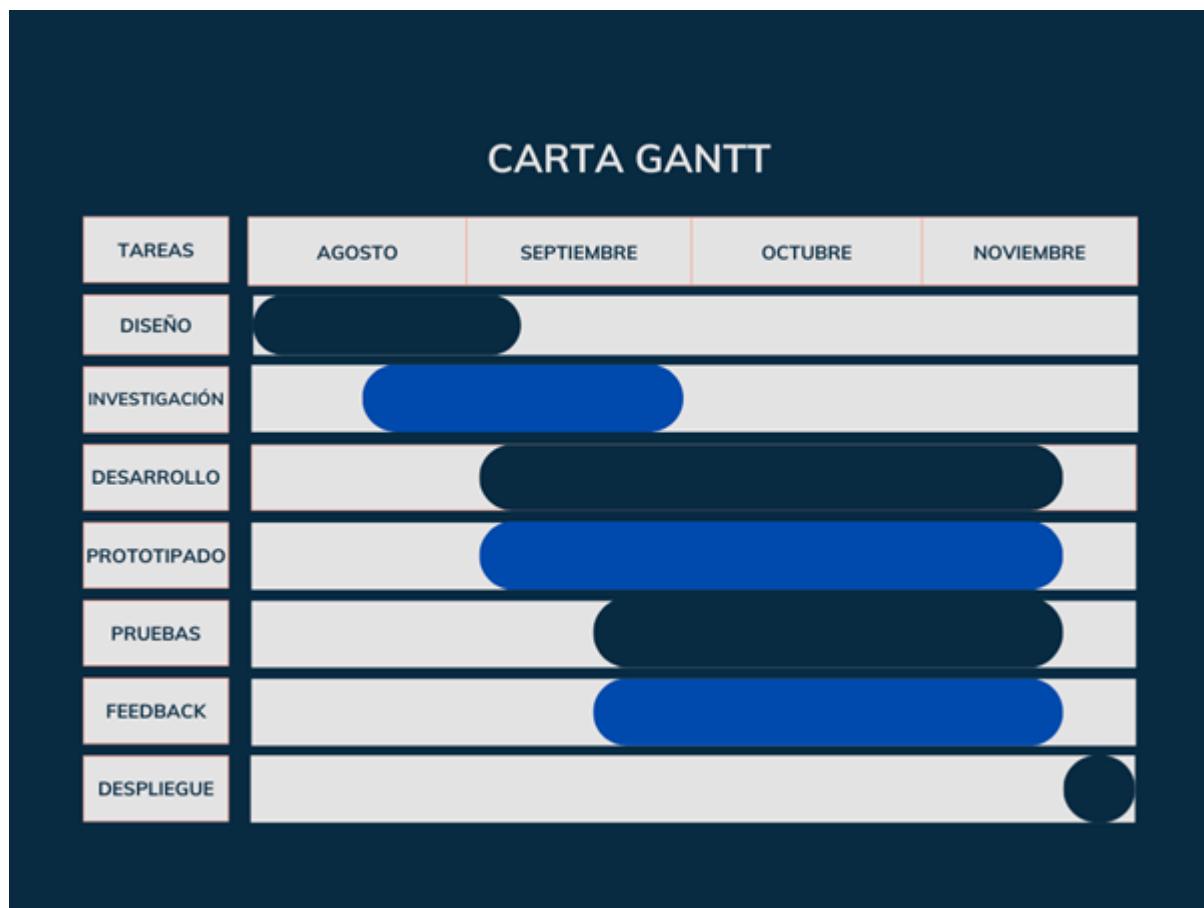


Figura 9.1: Carta Gantt del proyecto

### 9.3.2. Anexo Resumen de Esfuerzo

Actividad	Número de Horas
Preparación del proyecto	20
Desarrollo del módulo de autos	50
Desarrollo del módulo de pistas	50
Desarrollo del módulo de sesiones	80
Desarrollo del módulo de temporadas	80
Corrección de errores de código	40
Despliegue de la aplicación	27
Control de versiones	37
<b>Total</b>	<b>384</b>

## 9.4. Anexos Retrospectiva del Proyecto

### 9.4.1. Anexo Iteraciones en el Desarrollo

Funcionalidad	Fecha	Retroalimentación
Módulo de Autos (1)	12/11/2023	Añadir visualización para todos los autos y un sólo auto.
Módulo de Autos (2)	12/11/2023	Agregar los autos tanto a la navegación como a la sub-navegación de la aplicación
Módulo de Autos (3)	21/11/2023	Corregir problemas internos del modelo de autos
Módulo de Pistas (1)	11/11/2023	Corregir problemas de validación en el modelo de Pista
Módulo de Pistas (2)	12/11/2023	Agregaron las pistas tanto a la navegación como a la sub-navegación de la aplicación
Módulo de Pistas (3)	28/11/2023	Mejorar la relación a modelos de Pista desde la visualización del modelo de Sesión.
Módulo de Temporadas (1)	19/11/2023	Generar modelos de Ranking asociados a la temporada automáticamente después de crearla
Módulo de Temporadas (2)	30/11/2023	Eliminar entradas de resultado de jugador asociadas a la temporada si estas tienen todos sus parámetros en cero
Módulo de Sesiones (1)	19/11/2023	Determinar el número de cada sesión automáticamente una vez creada
Módulo de Sesiones (2)	28/11/2023	Añadir botón de descarga para el Session Log asociado a cada sesión
Módulo de Sesiones (3)	29/11/2023	Añadir acciones de administrador a la vista de sesión, tales como eliminar