



DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN  
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS  
UNIVERSIDAD DE CHILE  
CC3201-1 BASES DE DATOS

## GAME SPEEDRUN RECORDS - GRUPO 40

---

# HITO N°3

---

Grupo 40: Leonardo Rikhardsson  
Alejandro Mori  
Ignacio Humire  
Adolfo Arenas  
Profesor: Matías Toro I.

Fecha de entrega: 6 de Noviembre de 2023  
Santiago, Chile

## 1. Resumen Hito 0

En el hito anterior se estableció la fuente de datos **Game Speedrun Records** como base de datos para desarrollar el proyecto del curso. Con respecto a esta base de datos, se identificaron los distintos atributos que posee y se describieron los mismos. Adicionalmente, se escogió como problema el desarrollar una aplicación web para visualizar y analizar las estadísticas speedruns en distintos videojuegos, para ayudar al usuario a escoger un videojuego que speedrunear en base a sus gustos, tiempo máximo que está dispuesto a dedicar, género de interés, etc.

## 2. Resumen Hito 1

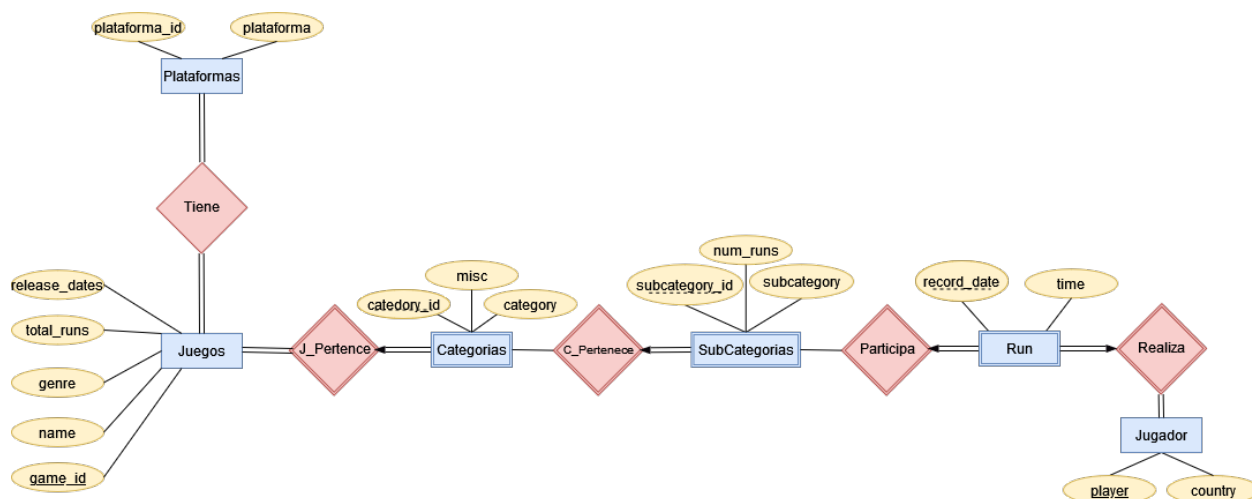


Figura 1: Modelo Entidad Relación

Por cómo está construida la base de datos, las categorías no pueden ser distinguidas unas de otras de forma única mediante sus id's, por ejemplo, la categoría «100 %» se encuentra en la mayoría de los juegos presentes, pero no habría como diferenciar el «100 %» de un juego A con el «100 %» de un juego B solamente con la id de la categoría. Para solucionar esto último, se ha optado por definir a las categorías como entidades débiles, usando las id's de estas (llaves parciales) en conjunto a las id's de los juegos (llaves primarias) para poder diferenciarlas entre ellas. El mismo argumento es válido para justificar a las subcategorías como entidades débiles. Por ejemplo, una Run se puede comparar con otra solamente si y solamente si la otra Run pertenece a la misma tupla (game\_id, category\_id, subcategory\_id). Por otra parte, la idea de relacionar Jugador con Juegos no la consideramos necesaria para nuestro objetivo ya que nuestro foco central está en las Runs. Sin embargo para facilitar la búsqueda de información decidimos hacer para este hito que plataforma sea una entidad y relacionarla con juegos.

Un jugador puede realizar una o más runs para un solo juego u otros. Para diferenciar estas runs de otras en distintos juegos, se construirán como entidades débiles que dependerán de las

id's de una subcategoría, que por transitividad dependerá finalmente de un juego. También, para identificar las runs de otros jugadores, se tomarán los nombres de los jugadores (nicknames).

## 2.1. Modelo Relacional

En base al modelo Entidad Relación presentado previamente, construimos el modelo relacional como sigue:

**Juegos**(game\_id: Int, name: String, genre: String, total\_runs: Int, release\_date: Date)

**Categorias**(category\_id: Int, ju\_game\_id: Int, category: String, misc: Bool)  
(ju\_game\_id) REF **Juegos**(game\_id)

**SubCategorias**(subcategory\_id: Int, c\_category\_id: Int, c\_ju\_game\_id: Int, subcategory: String, num\_runs: Int)  
(c\_category\_id, c\_ju\_game\_id) REF **Categorias**(category\_id, ju\_game\_id)

**Run**(record\_date: Date, s\_subcategory\_id: Int, s\_c\_category\_id: Int, s\_c\_ju\_game\_id: Int, j\_player: String, time: Double precision)  
(s\_subcategory\_id, s\_c\_category\_id, s\_c\_ju\_game\_id, j\_player) REF  
**SubCategorias**(subcategory\_id, c\_category\_id, c\_ju\_game\_id),  
j\_player REF **Jugador**(player)

**Jugador**(player: String, country: String)

**Plataformas**(plataforma\_id: Int, plataforma: String)

**Juegos\_plataformas**(ju\_game\_id: Int, p\_plataforma\_id: Int)  
(ju\_game\_id) REF **Juegos**(game\_id),  
(p\_plataforma\_id) REF **Plataformas**(plataforma\_id)

## 3. Hito 2

### 3.1. Implementación relacional

A continuación se muestran las Queries realizadas para crear las tablas de las entidades en la base de datos. Notar que no fue necesario crear tablas para las relaciones puesto que al realizar las tablas para las entidades débiles estas ya relacionan llaves primarias entre tablas.

```
1 create table juegos(  
2     game_id bigint,  
3     name varchar(255) not null,  
4     genre varchar(255) not null,  
5     total_runs bigint not null,  
6     release_date date not null,  
7     primary key (game_id)  
8 );  
9  
10 create table categorias(  
11     category_id bigint not null,  
12     ju_game_id bigint not null,  
13     category VARCHAR(255) not null,  
14     misc boolean not null,  
15     primary key (category_id, ju_game_id),  
16     foreign key (ju_game_id) references juegos(game_id)  
17 );  
18  
19 create table subcategorias(  
20     subcategory_id bigint not null,  
21     c_category_id bigint not null,  
22     c_ju_game_id bigint not null,  
23     subcategory VARCHAR(255) not null,  
24     num_runs bigint,  
25     primary key (subcategory_id, c_category_id, c_ju_game_id),  
26     foreign key (c_category_id, c_ju_game_id) references categorias(  
27         category_id, ju_game_id)  
28 );  
29  
30 create table run(  
31     record_date timestamp,  
32     s_subcategory_id bigint not null,  
33     s_c_category_id bigint not null,  
34     s_c_ju_game_id bigint not null,  
35     j_player varchar(255),  
36     time double precision,  
37     primary key (record_date, s_subcategory_id, s_c_category_id,  
38         s_c_ju_game_id, j_player),  
39     foreign key (s_subcategory_id, s_c_category_id, s_c_ju_game_id)  
40     references subcategorias(subcategory_id, c_category_id, c_ju_game_id),  
41     foreign key (j_player) references jugador(player)  
42 );
```

```
40
41 create table jugador(
42     player VARCHAR(255) not null ,
43     country VARCHAR(255) not null ,
44     primary key (player)
45 );
46
47 create table plataformas(
48     plataforma_id serial ,
49     plataforma varchar(255) not null ,
50     primary key (plataforma_id)
51 );
52
53 create table juegos_plataformas(
54     ju_game_id bigint ,
55     p_plataforma_id bigint ,
56     primary key (ju_game_id, p_plataforma_id),
57     foreign key (ju_game_id) references juegos(game_id),
58     foreign key (p_plataforma_id) references plataformas(plataforma_id)
59 );
```

Código 1: Queries usadas para crear las tablas en la DB

## 3.2. Carga de datos

### 3.2.1. Código

A continuación se muestra el código que se escribió para cargar los datos correctamente a la base de datos. Cabe destacar que nuestro caso el *.csv* que escogimos tenía pocos errores por lo que fue innecesario hacer *parse* a los datos. Además, no separamos los jugadores cuando las runs son grupales pues no lo consideramos relevante para la creación de la DB.

```

1 from binascii import a2b_base64
2 import psycopg2
3 import psycopg2.extras
4 import csv
5 import re
6
7 conn = psycopg2.connect(
8     host="cc3201.dcc.uchile.cl",
9     database="cc3201",
10    user="cc3201",
11    password="a",
12    port="5540"
13 )
14
15 cur = conn.cursor()
16
17 def findOrInsert(name):
18     cur.execute("select plataforma_id from plataformas where plataforma=%s
19                 limit 1", [name])
20     r = cur.fetchone()
21     if(r):
22         return r[0]
23     else:
24         cur.execute("insert into plataformas (plataforma) values (%s)
25                     returning plataforma_id", [name])
26         return cur.fetchone()[0]
27
28 with open('speedrun.csv', encoding="utf8") as csvfile:
29     reader = csv.reader(csvfile, delimiter=',', quotechar='"')
30     i = 0
31     for row in reader:
32         i+=1
33         if i==1:
34             continue
35
36         #tablas: juegos, categorias, juego_categoria, subcategorias,
37         #categorias_subcategorias, run, subcategory-run, jugador, run-jugador
38
39         #columnas: Game_Id,Category_Id,Subcategory_Id,Name,Genre,Platforms
40         #Total_Runs,Release_Date,Misc,Category,Num_Runs,Subcategory,Time_0,
41         #Country_0,Players_0,Record_Date_0
42
43         #datos para insertar en "juegos"
```

```

39     game_id = row[0]
40     name = row[3]
41     genre = row[4]
42     total_runs = row[6]
43     release = row[7]
44
45     #buscamos si ya está el juego en la tabla
46     cur.execute("select game_id from juegos where game_id=%s limit 1",
47 [game_id])
48     r = cur.fetchone()
49     if(not r): #si no existe
50         cur.execute("insert into juegos (game_id, name, genre,
51 total_runs, release_date) values (%s, %s, %s, %s, %s)", [game_id, name,
52 genre, total_runs, release])
53
54     #separamos las plataformas
55     platforms = [m.strip() for m in row[5].split(',')]
56
57     #buscamos o creamos sus id's
58     for p in platforms:
59         findOrInsert(p)
60
61     #datos para crear un "jugador"
62     player = row[14]
63     country = row[13]
64     position = 1
65     #buscamos si ya está el player en la tabla
66     cur.execute("select player from jugador where player=%s limit 1",
67 [player])
68     r = cur.fetchone()
69     if(not r):
70         cur.execute("insert into jugador (player, country) values (%s,
71 %s)", [player, country])
72     conn.commit()
73 conn.close()
74
75 conn = psycopg2.connect(
76     host="cc3201.dcc.uchile.cl",
77     database="cc3201",
78     user="cc3201",
79     password="a",
80     port="5540"
81 )
82
83 cur = conn.cursor()
84 with open('speedrun.csv', encoding="utf8") as csvfile:
85     reader = csv.reader(csvfile, delimiter=',', quotechar='"')
86     i = 0
87     for row in reader:
88         i+=1

```

```
86         if i==1:
87             continue
88
89         #empezamos llenando la tabla de "juegos_plataformas"
90         game_id = row[0] #volvemos a sacar el game_id
91
92         #separamos las plataformas
93         platforms = [m.strip() for m in row[5].split(',')]
94
95         #buscamos o creamos sus id's
96         platforms_ids = []
97         for p in platforms:
98             a = findOrInsert(p)
99             platforms_ids.append(a)
100
101         #buscamos si ya está el par en la tabla
102         cur.execute("select (ju_game_id, p_plataforma_id) from
juegos_plataformas where (ju_game_id, p_plataforma_id)=(%s, %s) limit 1
", [game_id, a])
103         r = cur.fetchone()
104         if(not r):
105             cur.execute("insert into juegos_plataformas (ju_game_id,
p_plataforma_id) values (%s, %s)", [game_id, a])
106
107         #datos para insertar en "categorias"
108         category_id = row[1]
109         category = row[9]
110         misc = row[8]
111         player = row[14] #volvemos a sacar al player
112
113         #buscamos si ya está la categoría en la tabla
114         cur.execute("select (category_id, ju_game_id) from categorias
where (category_id, ju_game_id)=(%s, %s) limit 1", [category_id,
game_id])
115         r = cur.fetchone()
116         if(not r):
117             cur.execute("insert into categorias (category_id, ju_game_id,
category, misc) values (%s, %s, %s, %s)", [category_id, game_id,
category, misc])
118
119         #datos para insertar en "subcategorias"
120         sub_id = row[2]
121         sub_category = row[11]
122         num_runs = row[10]
123
124         #buscamos si ya está la subcategoria en la tabla
125         cur.execute("select (subcategory_id, c_category_id, c_ju_game_id)
from subcategorias where (subcategory_id, c_category_id, c_ju_game_id)
=(%s, %s, %s) limit 1", [sub_id, category_id, game_id])
126         r = cur.fetchone()
127         if(not r):
```



```

128         cur.execute("insert into subcategorias (subcategory_id,
c_category_id, c_ju_game_id, subcategory, num_runs) values (%s, %s, %s,
%s, %s)", [sub_id, category_id, game_id, sub_category, num_runs])
129
130     #datos para crear "run"
131     runtime = row[12]
132     recordate = row[15]
133
134     #buscamos si ya está la run en la tabla y chequeando si está vacío
135     if(recordate != '' and recordate != None):
136         cur.execute("select (record_date, s_subcategory_id,
s_c_category_id, s_c_ju_game_id) from run where (record_date,
s_subcategory_id, s_c_category_id, s_c_ju_game_id)=(%s, %s, %s, %s)
limit 1", [recordate, sub_id, category_id, game_id])
137         r = cur.fetchone()
138         if(not r):
139             cur.execute("insert into run (record_date, s_subcategory_id
, s_c_category_id, s_c_ju_game_id, j_player, time) values (%s, %s, %s,
%s, %s, %s)", [recordate, sub_id, category_id, game_id, player, runtime
])
140     conn.commit()
141 conn.close()

```

Código 2: Código de Python para cargar datos en las tablas de la DB

### 3.2.2. Evidencia DB

game_id	name	genre	total_runs	release_date
0	Super Mario 64	3D Platformer	9174	1996-06-23
1	Minecraft: Java Edition	Sandbox	6869	2011-11-18
2	Super Mario Odyssey	3D Platformer	4284	2017-10-27
3	Celeste	2D Platformer	3815	2018-01-25
4	Mario Kart 8 Deluxe	Racing	7660	2017-04-28
5	Seterra	Misc	7763	1997-01-01
6	Portal	Puzzle	3676	2007-10-09
7	Super Metroid	Action	2826	1994-03-19
8	Getting Over It With Bennett Foddy	2D Platformer	2561	2017-10-06
9	Super Mario Sunshine	3D Platformer	2575	2002-07-19
10	ROBLOX: Speed Run 4	Parkour	5086	2013-01-01
11	Super Mario World	2D Platformer	3150	1990-11-21
12	Super Mario Bros.	2D Platformer	2277	1985-09-13
13	The Legend of Zelda: Ocarina of Time	Action-Adventure	3065	1998-11-21
14	The Legend of Zelda: A Link to the Past	Action-Adventure	2891	1991-11-21
15	Minecraft: Bedrock Edition	Sandbox	2147	2011-08-16
16	SpongeBob SquarePants: Battle for Bikini Bottom	3D Platformer	1084	2003-10-31
17	Resident Evil 2 (2019)	Horror	2880	2019-01-25
18	Pou	Misc	3930	2013-02-14
19	ROBLOX: Piggy	Puzzle	4081	2020-01-23

Figura 2: Tabla de juegos

category_id	ju_game_id	category	misc
0	0	120 Star	f
1	0	70 Star	f
2	0	16 Star	f
3	0	1 Star	f
4	0	0 Star	f
0	1	Any% Glitchless	f
1	1	Any%	f
2	1	All Achievements	f
3	1	All Advancements	f
4	1	Any% Random Seed Glitchless Co-op	t
5	1	All Advancements Co-op	t
6	1	Any% Glitchless (Demo)	t
7	1	Any% Glitchless (Peaceful)	t
8	1	Any% (Peaceful)	t
9	1	Combined Any% Glitchless	t
10	1	Any% (Time Travel)	t
0	2	Any%	f
1	2	World Peace	f
2	2	Dark Side	f
3	2	Darker Side	f
4	2	All Moons	f
5	2	100%	f
0	3	Any%	f
1	3	All Red Berries	f
2	3	True Ending	f
3	3	All Cassettes	f
4	3	All Hearts	f
5	3	All Chapters	f
6	3	100%	f
7	3	All A-Sides	t
8	3	All B-Sides	t
9	3	All C-Sides	t
0	4	48 Tracks	f
1	4	Nitro Tracks	f
2	4	Retro Tracks	f
3	4	Bonus Tracks	f
4	4	Nitro Cups	t
5	4	Retro Cups	t
6	4	Bonus Cups	t
7	4	DLC Cups	t
8	4	32 Tracks	t
0	5	Europe: Countries	f

Figura 3: Tabla de categorías

subcategory_id	c_category_id	c_ju_game_id	subcategory	num_runs
0	0	0	N64	456
1	0	0	VC	135
2	0	0	EMU	302
0	1	0	N64	928
1	1	0	VC	291
2	1	0	EMU	907
0	2	0	N64	1511
1	2	0	VC	588
2	2	0	EMU	3318
0	3	0	N64	299
1	3	0	VC	55
2	3	0	EMU	222
0	4	0	N64	105
1	4	0	VC	10
2	4	0	EMU	47
0	0	1	Set Seed,Pre 1.9	218
1	0	1	Set Seed,1.9-1.15	149
2	0	1	Set Seed,1.16+	989
3	0	1	Random Seed,Pre 1.9	299
4	0	1	Random Seed,1.9-1.15	648
5	0	1	Random Seed,1.16+	3174
0	1	1	Set Seed,Pre 1.9	31
6	1	1	Set Seed,1.9-1.13	16
7	1	1	Set Seed,1.14+	11
3	1	1	Random Seed,Pre 1.9	31
8	1	1	Random Seed,1.9-1.13	9
9	1	1	Random Seed,1.14+	10
10	2	1	1.0-1.6,SS	6
11	2	1	1.0-1.6,SSG	4
12	2	1	1.0-1.6,RS	3
13	2	1	1.0-1.6,RSG	5
14	2	1	1.8-1.11,SS	11
15	2	1	1.8-1.11,SSG	3
16	2	1	1.8-1.11,RS	3
17	2	1	1.8-1.11,RSG	3
18	3	1	1.12,SS	1
19	3	1	1.12,SSG	1
20	3	1	1.12,RS	0
21	3	1	1.12,RSG	4
22	3	1	1.13,SS	0
23	3	1	1.13,SSG	0
24	3	1	1.13,RS	0

Figura 4: Tabla de subcategorías

record_date time	s_subcategory_id	s_c_category_id	s_c_fu_game_id	j_player
2022-03-01 19:41:53	0	0	0	cheese
2021-03-23 00:47:15	1	0	0	Paracusia
2021-03-10 18:47:17	2	0	0	Hebula
2021-11-11 16:05:03	0	1	0	Meegee
2021-06-14 23:29:08	1	1	0	Aleph64
2022-01-28 13:40:07	2	1	0	tailhou
2021-12-31 14:20:50	0	2	0	Suigi
2022-03-18 20:40:21	1	2	0	Finnii
2022-01-18 23:50:23	2	2	0	Kaimatix
2021-12-08 14:20:47	0	3	0	KANNO
2021-11-11 07:20:10	1	3	0	Finnii
2020-07-24 08:09:50	2	3	0	KANNO
2022-02-15 06:00:46	0	4	0	KANNO
2021-01-07 05:00:48	1	4	0	Finnii
2020-07-21 12:48:07	2	4	0	KANNO
2021-09-07 01:43:33	0	0	1	Galacticafro
2021-10-24 02:02:24	1	0	1	Spropl
2021-10-25 23:27:51	2	0	1	Rayoh
2021-12-28 07:51:20	3	0	1	whatevermarco
2021-07-01 03:16:39	4	0	1	Korbanoes

Figura 5: Tabla de run

player	country
cheese	Spain
Paracusia	Australia
Hebula	Australia
Meegee	United States
Aleph64	Germany
tailhou	Japan
Suigi	Canada
Finnii	Germany
Kaimatix	England
KANNO	Germany
Galacticafro	Germany
Spropl	United States
Rayoh	United States
whatevermarco	Brazil
Korbanoes	United States
Cobol377a	Finland
Tommydavis	Antarctica
TheMizzler	Unknown
Achuck	Singapore
Elizium	Canada
Siriuslaser	United States
Utopia21	Poland
Schmöl	Switzerland
Orangutan	Finland
Majeston	United States
DougTuff4	United Kingdom
Cavallero4	Canada

Figura 6: Tabla de jugador

plataforma_id	plataforma
1	Nintendo 64
2	Wii Virtual Console
3	Wii U Virtual Console
4	Switch
5	PC
6	PlayStation 4
7	Xbox One
8	Xbox One X
9	Xbox One S
10	Google Stadia
11	PlayStation 5
12	Xbox Series X
13	Xbox Series S
14	Android
15	iOS
16	Web
17	PlayStation 3
18	Xbox 360
19	Super Nintendo
20	New Nintendo 3DS Virtual Console
21	SNES Classic Mini
22	MiSTer
23	Analogue Super Nt
24	GameCube
25	Wii
26	Macintosh
27	Nintendo 3DS Virtual Console
28	Nintendo Entertainment System
29	Game Boy Advance
30	Famicom Disk System

Figura 7: Tabla de plataformas

ju_game_id	p_plataforma_id
0	1
0	2
0	3
0	4
1	5
2	4
3	6
3	7
3	5
3	4
3	8
3	9
3	10
3	11
3	12
3	13
4	4
5	5
5	14
5	15
5	16
6	17
6	18
6	7
6	5
7	19
7	2
7	3
7	20
7	4
7	21
7	22
7	23
8	5
8	14
8	15
9	24
9	25
9	4
10	5
10	14
10	15
10	26
11	19
11	2

Figura 8: Tabla de id's de juegos y plataformas

## 4. Hito 3

A continuación se muestra el trabajo hecho para el hito tres, lo que incluye las consultas pedidas y la creación de una página web usando Apache2 más PHP.

### 4.1. Consultas

1. Todos los juegos, cuya categoria sea “input“, tal que su runner sea del pais “input“.

```
1 SELECT name FROM speed.juegos WHERE game_id IN
2     (SELECT s_c_ju_game_id FROM speed.run
3     WHERE (s_c_ju_game_id, s_c_category_id) IN
4         (SELECT ju_game_id, category_id FROM speed.categorias
5         WHERE category LIKE :valor1)
6     AND j_player IN
7     (SELECT player FROM speed.jugador
8     WHERE country = :valor2))
9 ;
```

En :valor1 debe ir un string como input, un ejemplo de categoría es 'Any'. Por otro lado, para el :valor2 también deben ir strings como input, en este caso al ser país, un ejemplo sería: 'United States'. Es posible dejar el :valor1 vacío para elegir cualquier categoría.

2. Todos los juegos que sean de determinada plataforma y tengan sub-1 minuto

```
1 SELECT name FROM speed.juegos
2 WHERE game_id IN
3     ((SELECT s_c_ju_game_id FROM speed.run
4     WHERE time < 60)
5     INTERSECT
6     (SELECT ju_game_id FROM speed.juegos_plataformas
7     WHERE p_plataforma_id IN
8         (SELECT plataforma_id FROM speed.plataformas
9         WHERE plataforma LIKE :valor1)))
10 ;
```

En :valor1 deben ir strings como input, un ejemplo de plataforma sería 'PlayStation 5'. Se puede dejar este campo en blanco para elegir un juego de cualquier plataforma.

3. Todos los juegos usando determinados inputs

```
1 SELECT name FROM speed.juegos
2 WHERE game_id IN
3     (SELECT DISTINCT s_c_ju_game_id FROM speed.run
4     WHERE time < :valor2
5     AND j_player IN
6     (SELECT player FROM speed.jugador
7     WHERE country = :valor3)
8     AND s_c_ju_game_id IN
9     (SELECT game_id FROM speed.juegos
10     WHERE name LIKE :valor1))
```

11 ;

En :valor1 deben ir strings como input, un ejemplo de franquicia sería 'Call of Duty'. Por otro lado, para el :valor2 deben ir un número entero mayor a cero como input, donde un ejemplo sería 20. Finalmente, en el :valor3 deben ir strings como input, en este caso al ser país, un ejemplo sería: 'United States'. Dejar en blanco el campo de franquicia permite ver juegos de cualquier saga.

## 4.2. Optimización

### 4.2.1. Índices

1. B+ Tree sobre time en la tabla run, ya que dentro de las consultas posibles podríamos buscar las runs cuyo tiempo sea mayor o menor a un tiempo deseado, o sea, haríamos una búsqueda por rango.
2. Hash sobre country en la tabla jugador, ya que acá nunca se realizarán búsquedas por rangos, solo se utilizarán igualdades para buscar los países buscados.
3. B+ Tree sobre record\_date en la tabla run. Utilizando el mismo argumento para time, podríamos buscar todas las runs cuya fecha sea más reciente que la fecha especificada.

## 4.3. Seguridad

Utilizar PDO::prepare() junto con PDOStatement::execute() para preparar y ejecutar declaraciones que se usarán repetidamente con diversos parámetros mejora la eficiencia de la aplicación. Además, contribuye a prevenir ataques de inyección SQL al eliminar la necesidad de realizar manualmente la manipulación de comillas alrededor de los parámetros.

```
try {
    $pdo = new PDO('pgsql:
        host=localhost;
        port=5432;
        dbname=cc3201;
        user=cibernavegante;
        password=XxAntionioX');
    $variable1='% ' . $_GET['input_categoria'] . ' %';
    $variable2=$_GET['input_pais1'];
    $stmt = $pdo->prepare("SELECT name FROM speed.juegos WHERE game_id IN (SELECT s_c_ju_game_id FROM speed.run WHERE (s_c_ju_game_id, s_c_category_
    $stmt->execute(['valor1' => $variable1, 'valor2' => $variable2]));
    $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);

    foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
        echo $v;
    }
}
```

Figura 9: Consulta.php

## 4.4. Interfaz

El link de acceso para la aplicación web: <https://grupo40.cc3201.dcc.uchile.cl>  
A continuación presentaremos pantallazos para evidenciar el correcto uso de la página.

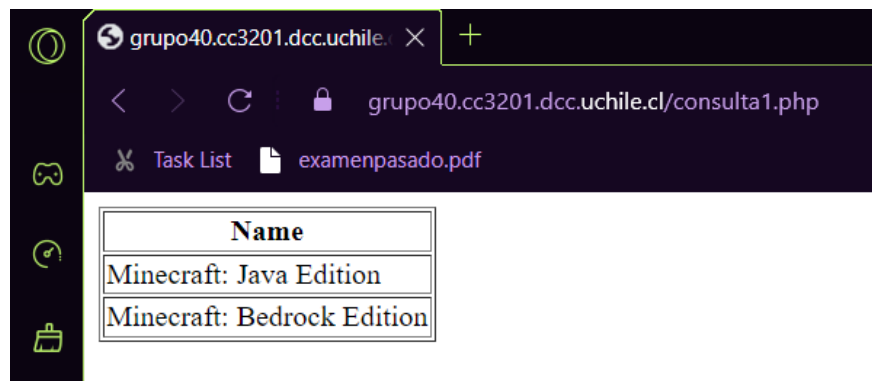
#### 4.4.1. Consulta 1

**Todos los juegos, cuya categoria sea "input",  
tal que su runner sea del pais "input"**

Ingrese una categoria

Ingrese un pais

Figura 10: Consulta 1 con ejemplo de input

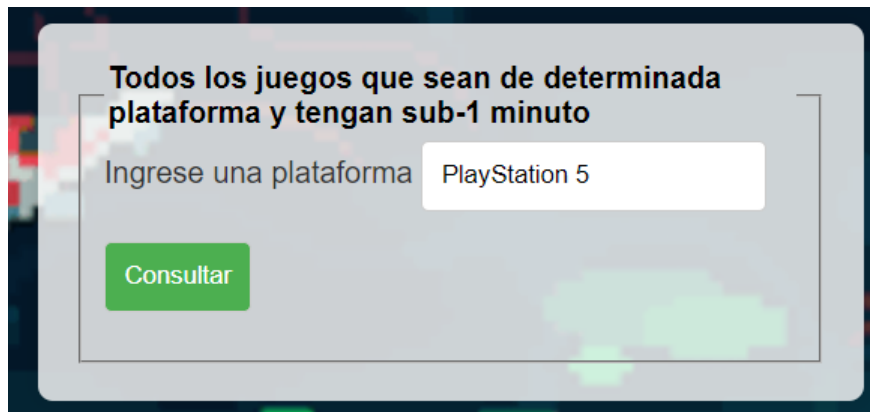


Name
Minecraft: Java Edition
Minecraft: Bedrock Edition

Figura 11: Tabla de resultado consulta 1



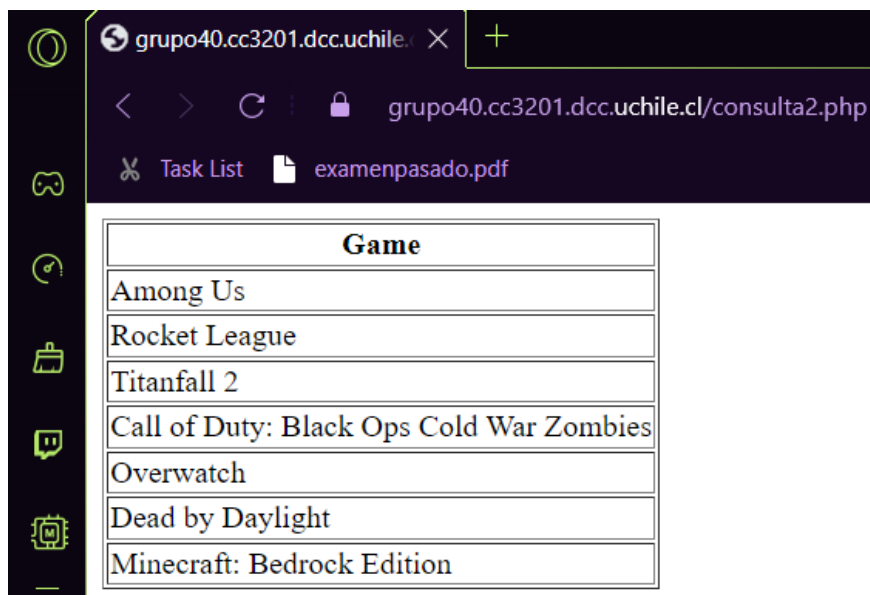
#### 4.4.2. Consulta 2



Todos los juegos que sean de determinada plataforma y tengan sub-1 minuto

Ingrese una plataforma

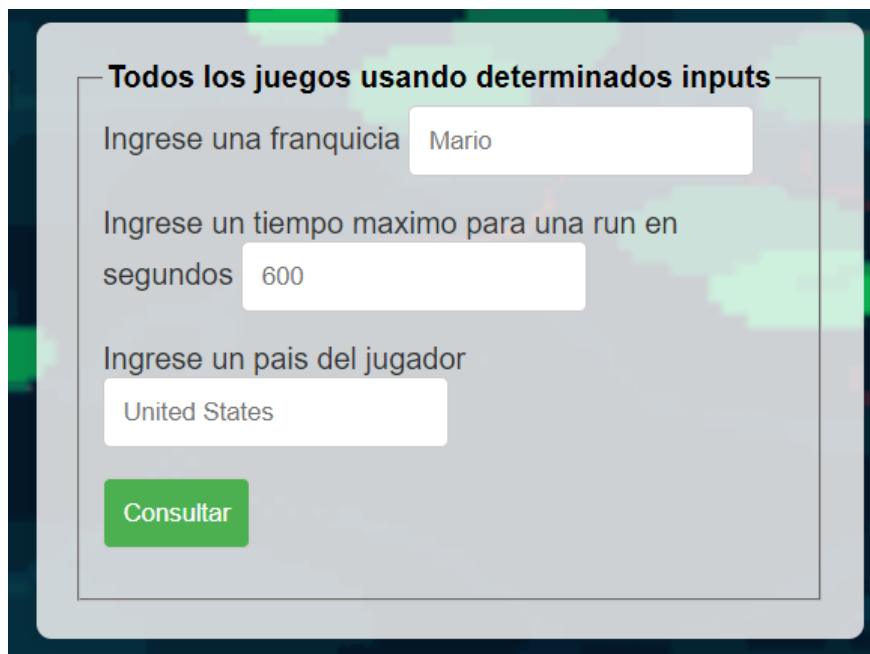
Figura 12: Consulta 2 con ejemplo de input



Game
Among Us
Rocket League
Titanfall 2
Call of Duty: Black Ops Cold War Zombies
Overwatch
Dead by Daylight
Minecraft: Bedrock Edition

Figura 13: Tabla de resultado consulta 2

#### 4.4.3. Consulta 3



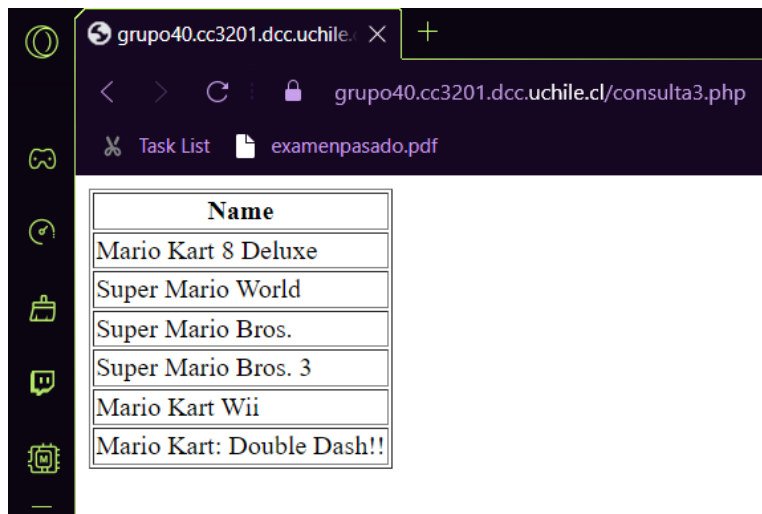
**Todos los juegos usando determinados inputs**

Ingrese una franquicia

Ingrese un tiempo maximo para una run en segundos

Ingrese un pais del jugador

Figura 14: Consulta 3 con ejemplo de input



Name
Mario Kart 8 Deluxe
Super Mario World
Super Mario Bros.
Super Mario Bros. 3
Mario Kart Wii
Mario Kart: Double Dash!!

Figura 15: Tabla de resultado consulta 3