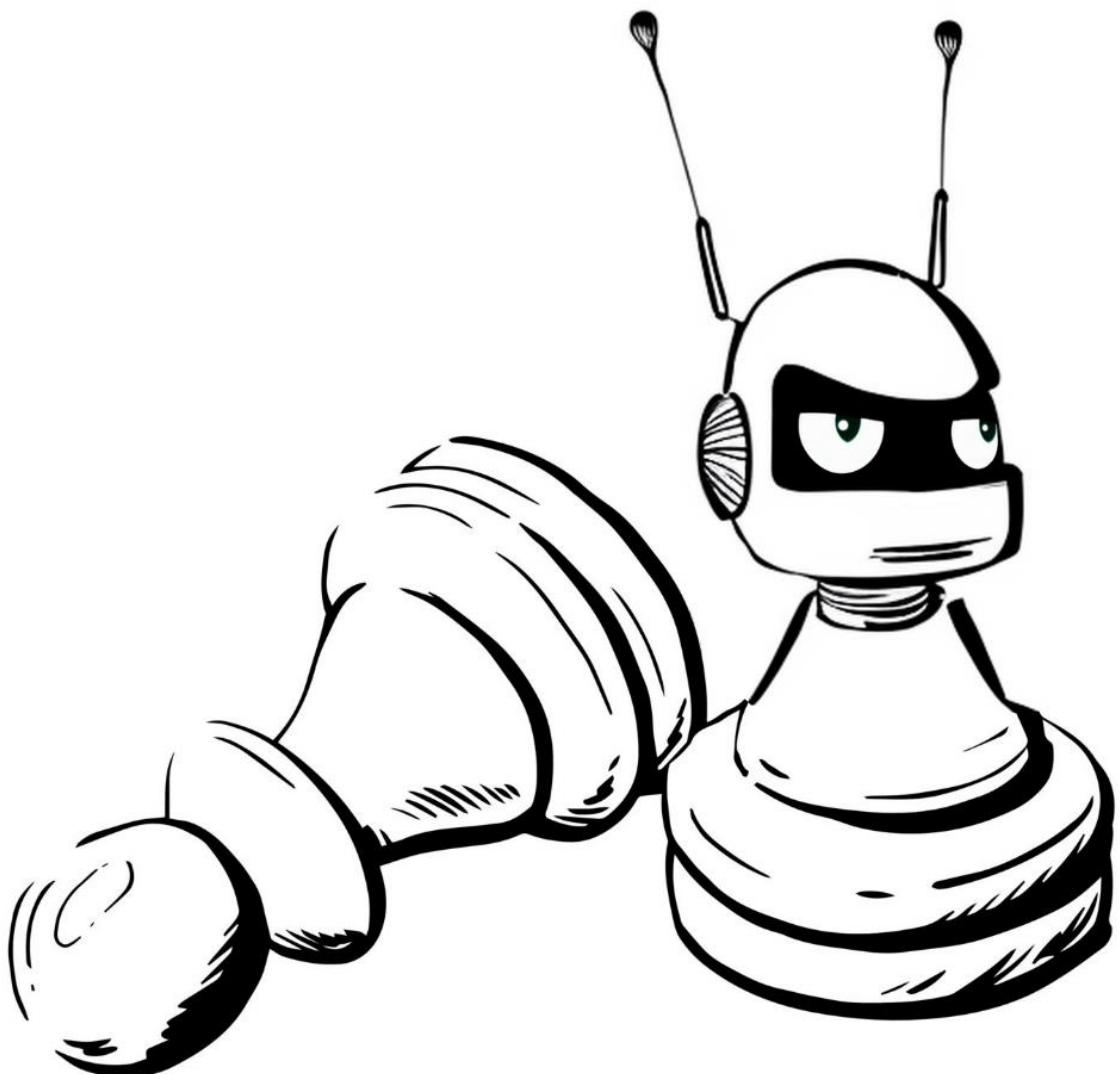


ALPHAUPO

Inteligencia artificial para el ajedrez



Universidad Pablo de Olavide, Ingeniería Informática en Sistemas de Información

Eugenio Menacho de Góngora

21-22-A14

ALPHAUPO

Inteligencia artificial para el ajedrez

Alumno: Eugenio Menacho de Góngora

Tutor: Francisco Martínez Álvarez

Tutora: Alicia Troncoso Lora

Universidad Pablo de Olavide, Escuela Politécnica Superior, Ingeniería Informática en

Sistemas de Información

Contenido

FIGURAS	9
PREÁMBULO	26
1. Resumen.....	28
2. Motivación	29
3. Introducción:	30
ALPHAUPO	33
1. ¿QUÉ ES EL AJEDREZ?	35
2. ¿CÓMO SE MUEVEN LAS PIEZAS?	39
2.1 Peones	39
2.2 Alfiles.....	46
2.3 Caballos:	47
2.4 Torres	48
2.5 Reinas.....	49
2.6 Rey	50
3. REGLAS	51
3.1 Jaque	51
3.2 Jaque Mate	52
3.3 Rey ahogado.....	53
3.4 Repetición de movimientos.....	54
3.5 Acuerdo mutuo.....	54

3.6	Enroque:	55
3.6.1	Enroque corto.....	55
3.6.2	Enroque largo.....	57
4.	HISTORIA DEL AJEDREZ	59
4.1	Origen:	59
5.	TEORÍA EN EL AJEDREZ.....	60
6.	APARICIÓN DE LA INFORMÁTICA EN EL JUEGO	67
6.1	Turochamp	67
6.2	Deep Blue IBM.....	74
6.3	Alphazero.....	78
7.	ESTUDIO DE LOS ALGORITMOS.....	80
7.1	Minimax.....	80
7.2	Turochamp	99
7.3	Minimax con ponderación Alpha y Beta	116
8.	OTRA PERSPECTIVA.....	136
8.1	Redes convolucionales para la predicción de jaques sin conocimiento.....	136
8.2	Redes convolucionales para la clasificación de jaques mate sin conocimiento	
	161	
8.3	Redes convolucionales como función de evaluación.....	164
9.	CONCLUSIONES	168
	ANEXO I – PLAN DE PROYECTO	170

1.	INTRODUCCIÓN	173
2.	OBJETIVOS DEL PROYECTO	174
3.	ORGANIZACIÓN DEL PROYECTO	176
3.1	Diagrama de Organización del proyecto.....	176
3.2	Identificación de los interesados.....	177
3.3	Responsabilidades y funciones de los interesados.....	178
4.	METODOLOGÍA DE GESTIÓN DEL PROYECTO	179
5.	PROGRAMA DE TRABAJO	181
5.1	Alcance y objetivos del programa de trabajo.....	181
5.2	Plan de tareas	181
5.2.1	Investigación	183
5.2.2	Diseño	184
5.2.3	Desarrollo.....	185
5.2.4	Integración	187
5.2.5	Pruebas y Cierre.....	188
5.3	Asignación de recursos	189
5.4	Asignación de tareas	190
6.	EVALUACIÓN Y PLANIFICACIÓN DE RIESGOS.....	193
7.	TEMAS ABIERTOS Y DECISIONES PENDIENTES	195
8.	OTROS ASPECTOS DEL PROYECTO	195
8.1	Ubicación del equipo de desarrollo.....	195

8.2	Entorno de desarrollo y herramientas	195
8.3	Equipos	196
	ANEXO II – PLANTILLA DOCUMENTO ANÁLISIS.....	197
1.	DEFINICIÓN DEL SISTEMA.....	200
1.1	Alcance del sistema.....	200
2.	ESPECIFICACIÓN DE REQUISITOS.....	203
2.1	Catálogo de requisitos.....	203
2.1.1	Requisitos funcionales	203
2.1.2	Requisitos no funcionales	209
2.2	Especificación de los casos de uso.....	212
2.2.1	Identificación y definición de los actores	212
2.2.2	Diagrama de Casos de Uso	214
2.2.3	Especificación de Casos de Uso	215
3.	SUBSISTEMAS DE ANÁLISIS	225
3.1	Identificación de los subsistemas de análisis	225
3.2	Relaciones entre subsistemas de Análisis	226
3.3	Matriz de Trazabilidad	226
4.	ANÁLISIS DE LAS CLASES POR SUBSISTEMAS.....	228
4.1	Paquetes de clases de negocio.....	228
4.1.1	Subsistema “SUB01 – Motor de ajedrez”	228
4.1.2	Subsistema “SUB02 – Base de Datos”	233

4.1.3 Subsistema “SUB03 – “Subsistema de gestión de partidas y usuarios”	234
4.1.4 Subsistema “SUB04 – “Acceso a la aplicación”	236
5. INTERFACES DE USUARIO	238
5.1 Principios de Diseño de la Interfaz de Usuario.....	238
5.2 Interfaces principales aplicación.....	239
5.2.1 Interfaz del módulo IU-0001 – Juega	239
5.2.2 Interfaz del módulo IU-0002 – Base de Datos.....	241
5.2.3 Interfaz del módulo IU-0003 – Analizar partida.....	242
5.2.4 Interfaz del módulo IU-0004 – Gestión de usuarios y partidas	244
5.2.5 Interfaz del módulo IU-0005 – Gestión de usuarios	245
5.2.6 Interfaz del módulo IU-0006 – Editar usuario	246
5.2.7 Interfaz del módulo IU-0007 – Gestionar partidas	247
5.2.8 Interfaz del módulo IU-0008 – Login.....	248
5.2.9 Interfaz del módulo IU-0009 – Register	249
DESARROLLO PÁGINA WEB	251
1. TECNOLOGÍAS SELECCIONADAS.....	253
1.1 Laravel:	253
1.2 XAMPP:.....	254
1.3 Visual Studio Code:	255
2. DISEÑO DEL MOTOR DE AJEDREZ	257
3. DISEÑO DE LOS ALGORITMOS DE AJEDREZ.....	260

4. INTERFAZ BASE DE DATOS	271
5. GESTIÓN DE USUARIOS	274
6. GESTIÓN DE PARTIDAS	274
MANUAL DE USUARIO	275
1. GOOGLE COLABORATORY	277
1.1 Instrucciones	278
2. PÁGINA WEB EN LOCAL	282
3. FUNCIONALIDADES PÁGINA WEB	286
3.1 Portada	286
3.2 Login	289
3.3 Register	289
3.4 Logout	290
3.5 Juega	290
3.6 Base de Datos.....	291
3.7 Gestión de Usuarios y Partidas	294
4. USAR PÁGINA EN LOCAL	296
4.1 Descargar XAMPP.....	296
4.2 Instalar XAMPP.....	297
4.3 Levantar el servidor apache y mysql	298
4.4 Accedemos al panel de phpmyadmin y importamos la base de datos	298
4.5 Levantamos la aplicación.....	299

BIBLIOGRAFÍA	300
1. BIBLIOGRAFÍA	301

FIGURAS

Figure 1. White pawn (classic). Retrieved May 1, 2023, from
<https://www.chess.com/chess-themes/pieces/classic/150/wp.png>

Figure 2. White rook (classic). Retrieved May 1, 2023, from
<https://www.chess.com/chess-themes/pieces/classic/150/wr.png>

Figure 3. White knight (classic). Retrieved May 1, 2023, from
<https://www.chess.com/chess-themes/pieces/classic/150/wn.png>

Figure 4. White bishop (classic). Retrieved May 1, 2023, from
<https://www.chess.com/chess-themes/pieces/classic/150/wb.png>

Figure 5. White queen (classic). Retrieved May 1, 2023, from
<https://www.chess.com/chess-themes/pieces/classic/150/wq.png>

Figure 6. White king (classic). Retrieved May 1, 2023, from
<https://www.chess.com/chess-themes/pieces/classic/150/wk.png>

Figure 7. Black pawn (classic). Retrieved May 1, 2023, from
<https://www.chess.com/chess-themes/pieces/classic/150/bp.png>

Figure 8. Black rook (classic). Retrieved May 1, 2023, from
<https://www.chess.com/chess-themes/pieces/classic/150/br.png>

Figure 9. Black knight (classic). Retrieved May 1, 2023, from
<https://www.chess.com/chess-themes/pieces/classic/150/bn.png>

Figure 10. Black bishop (classic). Retrieved May 1, 2023, from
<https://www.chess.com/chess-themes/pieces/classic/150/bb.png>

Figure 11. Black queen (classic). Retrieved May 1, 2023, from
<https://www.chess.com/chess-themes/pieces/classic/150/bq.png>

Figure 12. Black king (classic). Retrieved May 1, 2023, from
<https://www.chess.com/chess-themes/pieces/classic/150/bk.png>

Figure 13: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 14: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 15: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 16: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 17: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 18: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 19: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 20: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 21: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 22: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 23: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 24: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 25: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 26: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 27: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 28: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 29: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 30: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 31: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 32: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 33: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 34: Paul Morphy versus Johann Jacob Lowenthal. Retrieved May 1, 2023,
from
https://static.wixstatic.com/media/4e6a5e_070ceea60144421b2597f608f4dce71.jpg/v1/fill/w_400,h_391,al_c,q_80,usm_0.66_1.00_0.01,enc_auto/4e6a5e_070ceea60144421b2597f608f4dce71.jpg

Figure 35: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 36: Jose Raul Capablanca vs Frank Marshall. Retrieved May 3, 2023, from
<https://www.chessgames.com/perl/chessgame?gid=1095025>

Figure 37: Ferranti Mark I Chess [Photograph]. Retrieved May 3, 2023, from
<https://images.computerhistory.org/chess/ferranti.mark-i.1953.102645389.jpg?w=600>

Figure 38: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 39: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 40: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 41: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 42: Kasparov y Karpov en Reggio Emilia 1991-92 [Digital image]. Retrieved May 1, 2023, from <https://www.tabladeflandes.com/zenon2006/fotos/Kasparov-y-Karpov-en-Reggio-Emilia-1991-92.jpg>

Figure 43: Garry Kasparov, former world chess champion, plays against the machine [Photograph]. Retrieved from <https://www.washingtonpost.com/wp-apps/imrs.php?src=https://arc-anglerfish-washpost-prod-washpost.s3.amazonaws.com/public/G3HEW6RU6AI6XFUZADJRD4J5FU.jpg&w=916>

Figure 44: Training AlphaZero by self-play. Retrieved from
https://www.researchgate.net/figure/Left-Training-AlphaZero-by-self-play-gives-artificial-intelligence-in-chess-that_fig1_324949983

Figure 45: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 46: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 47: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 48: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 49: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 50: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 51: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 52: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 53: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 53: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 54: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 55: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 56: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 57: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 58: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 59: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 60: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 61: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 62: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 63: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 64: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 65: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 66: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 67: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 68: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 69: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 70: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 71: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 72: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 73: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 74: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 75: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 76: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 77: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 78: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 79: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 80: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 81: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 82: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 83: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 84: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 85: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 86: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 87: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 88: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 89: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 90: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 91: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 92: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 93: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 94: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 95: Chess Explorer. Retrieved May 1, 2023, from
<https://www.chess.com/explorer>

Figure 96: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 97: Convolution Formula. 2023, from
https://wikimedia.org/api/rest_v1/media/math/render/svg/46ca67ae76bc1e6841511aa12fab10aed9cb970d

Figure 98: Convolution Filter. 2023, from
https://media.licdn.com/dms/image/C4E12AQEVhgIM4MWlnQ/article-cover_image-shrink_720_1280/0/1636939462739?e=2147483647&v=beta&t=WUT7QfMDYH0fbT7TRvEABJUzJmzEgIT9LJX-Oflf5s

Figure 99: CNN Structure, 2023, from
https://upload.wikimedia.org/wikipedia/commons/6/63/Typical_cnn.png

Figure 100: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 101: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 102: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 103: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 104: Menacho, E. (2023). Spyder. Retrieved from Spyder IDE

Figure 105: Menacho, E. (2023). Spyder. Retrieved from Spyder IDE

Figure 106: Menacho, E. (2023). Spyder. Retrieved from Spyder IDE

Figure 107: Menacho, E. (2023). Spyder. Retrieved from Spyder IDE

Figure 108: Menacho, E. (2023). Spyder. Retrieved from Spyder IDE

Figure 109: Menacho, E. (2023). Spyder. Retrieved from Spyder IDE

Figure 110: Menacho, E. (2023). Spyder. Retrieved from Spyder IDE

Figure 111: Menacho, E. (2023). Spyder. Retrieved from Spyder IDE

Figure 112: Menacho, E. (2023). Spyder. Retrieved from Spyder IDE

Figure 113: Menacho, E. (2023). Spyder. Retrieved from Spyder IDE

Figure 114: Menacho, E. (2023). Spyder. Retrieved from Spyder IDE

Figure 115: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 116: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 117: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 118: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 119: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 120: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 121: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 122: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 123: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 124: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 125: Menacho, E. (2023). Google Colab. Retrieved from

<https://colab.research.google.com/>

Figure 126: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 127: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 128: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 129: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 130: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 131: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 132: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 133: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 134: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 135: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 136: Menacho, E. (2023). Google Colab. Retrieved from
<https://colab.research.google.com/>

Figure 137: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 138: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 139: Agile Methodology (2023). Retrieved from

<https://www.nvisia.com/hubfs/agile-methodology-chicago.png>

Figure 140: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 141: Menacho, E. (2023). Microsoft. Retrieved from Microsof Project

Figure 142: Menacho, E. (2023). Microsoft. Retrieved from Microsof Project

Figure 143: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 144: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 145: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 146: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 147: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 148: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 149: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 150: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 151: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 152: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 153: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 154: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 155: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 156: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 157: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 158: Laravel Icon. (2023). Iconduck. Retrieved from <https://static-00.icongduck.com/assets.00/laravel-icon-497x512-uwybstke.png>

Figure 159: Xampp Icon. (2023), Luisllamas. Retrieved from

<https://www.luisllamas.es/wp-content/uploads/2018/01/instalar-servidor-web-xampp.png>

Figure 160: Visual Studio Code. (2023). Wikimedia. Retrieved from https://upload.wikimedia.org/wikipedia/commons/thumb/9/9a/Visual_Studio_Code_1.35_icon.svg/2048px-Visual_Studio_Code_1.35_icon.svg.png

Figure 161: Menacho, E. (2023). Lucid App. Retrieved from <https://lucid.app/>

Figure 162: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 163: Menacho, E. (2023). Visual Studio Code. Retrieved from Visual Studio Code

Figure 164: Menacho, E. (2023). Visual Studio Code. Retrieved from Visual Studio Code

Figure 165: Menacho, E. (2023). Visual Studio Code. Retrieved from Visual Studio Code

Figure 166: Menacho, E. (2023). Visual Studio Code. Retrieved from Visual Studio Code

Figure 167: Menacho, E. (2023). Visual Studio Code. Retrieved from Visual Studio Code

Figure 168: Menacho, E. (2023). Visual Studio Code. Retrieved from Visual Studio Code

Figure 169: Menacho, E. (2023). Visual Studio Code. Retrieved from Visual Studio Code

Figure 170: Menacho, E. (2023). Visual Studio Code. Retrieved from Visual Studio Code

Figure 171: Menacho, E. (2023). Visual Studio Code. Retrieved from Visual Studio Code

Figure 172: Menacho, E. (2023). Visual Studio Code. Retrieved from Visual Studio Code

Figure 173: Menacho, E. (2023). Visual Studio Code. Retrieved from Visual Studio Code

Figure 174: Menacho, E. (2023). Visual Studio Code. Retrieved from Visual Studio Code

Figure 175: Web Workers (2023). Retrieved from <https://cdn-media-1.freecodecamp.org/images/-TZuLkY4dl1q3ngPdl3bAy-95GDku8xF105H>

Figure 176: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 177: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 178: TensorflowJS (2023). Tensorflow. Retrieved from <https://www.tensorflow.org/static/site-assets/images/project-logos/tensorflow-js-logo-social.png>

Figure 179: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 180: Menacho, E. (2023). Google Colab. Retrieved from <https://colab.research.google.com/>

Figure 181: Menacho, E. (2023). Google Colab. Retrieved from <https://colab.research.google.com/>

Figure 182: Menacho, E. (2023). Google Colab. Retrieved from <https://colab.research.google.com/>

Figure 183: Menacho, E. (2023). Google Colab. Retrieved from <https://colab.research.google.com/>

Figure 184: Menacho, E. (2023). Google Colab. Retrieved from <https://colab.research.google.com/>

Figure 185: Menacho, E. (2023). Google Colab. Retrieved from <https://colab.research.google.com/>

Figure 186: Menacho, E. (2023). CMD. Retrieved from cmd

Figure 187: Menacho, E. (2023). XAMPP. Retrieved from XAMPP

Figure 188: Menacho, E. (2023). XAMPP. Retrieved from XAMPP

Figure 189: Menacho, E. (2023). XAMPP. Retrieved from XAMPP

Figure 190: Menacho, E. (2023). CMD. Retrieved from cmd

Figure 191: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 192: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 193: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 194: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 195: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 196: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 197: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 198: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 199: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 200: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 201: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 202: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 203: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 204: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 205: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 206: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 207: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 208: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 209: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 210: Menacho, E. (2023). Alphaupo Website. Retrieved from localhost

Figure 211: Menacho, E. (2023). XAMPP. Retrieved from
<https://www.apachefriends.org/download.html>

Figure 212: Menacho, E. (2023). XAMPP. Retrieved from XAMPP

Figure 213: Menacho, E. (2023). XAMPP. Retrieved from XAMPP

Figure 214: Menacho, E. (2023). XAMPP. Retrieved from XAMPP

Figure 215: Menacho, E. (2023). CMD. Retrieved from cmd

Portada: Menacho, E. (2023). Stable Diffusion. Retrieved from
https://colab.research.google.com/github/TheLastBen/fast-stable-diffusion/blob/main/fast_stable_diffusion_AUTOMATIC1111.ipynb

La portada de este proyecto fue creada usando Stable difusión, un motor de generación de imágenes con inteligencia artificial.

PREÁMBULO

¿Hay que tener inteligencia para jugar?

No: el Ajedrez hace a la gente inteligente.

(Emanuel Lasker, ex campeón del mundo

Polonia, 1868; Nueva York, 1941)

1. Resumen

Objetivos:

El propósito principal del proyecto es realizar una investigación acerca de la interacción de la informática en el juego del ajedrez. En los últimos años, los avances de las técnicas de programación han supuesto una innovación en las formas de evaluar este complejo juego donde la matemática computacional ha irrumpido de manera muy directa, revolucionando el escenario clásico de afrontar la evolución del ajedrez.

Nuevas reglas de análisis han irrumpido en el juego, haciendo que éste cambie de forma radical, y de maneras muy diversas, haciendo que se busquen nuevas formas de usar los ordenadores como métodos de evaluación en el juego.

Esta evolución computacional ha hecho que la informática se convierta en la perfecta aliada para el juego: No sólo los motores de ajedrez han evolucionado, también ha mejorado la forma de acceder a las bases de datos y partidas haciendo la información más accesible tanto para los simples usuarios, como para el juego profesional, usando interfaces web, aplicaciones, promociones..., creando fórmulas de análisis y contraanálisis de manera cada vez más complejas, hasta el punto que su uso resulta indispensable para el jugador en su evolución y aprendizaje de estrategias de juego, íntimamente ligadas al análisis de multitud de datos coincidentes, que deben ser conjugados a la misma vez.

En definitiva, resulta claro que la informática y sus avances en el procesamiento da datos, con la creación de herramientas de programación e Inteligencia Artificial, han tenido y seguirán teniendo, un impacto fuerte en este juego, cuya evolución ya no se entiende si no es de la mano de la tecnología de la información y de la evolución de la matemática computacional.

Los objetivos principales del desarrollo de este proyecto pretenden crear una visión global de estas implicaciones de la evolución informática en las técnicas del juego, y bajo este prisma, consideraremos necesario desarrollar los siguientes mecanismos:

- Entender y comprender como funcionan previas implementaciones informáticas del ajedrez.
- Creación de una interfaz con el juego del ajedrez.
- Creación de varios motores de ajedrez.
- Investigar acerca de la forma de evaluar las posiciones usando aprendizaje profundo supervisado.
- Extracción de conclusiones y elaboración de informes pertinentes.
- Desarrollo de una aplicación web donde mostrar los resultados.
- Crear una base de datos con partidas para reproducirlas.
- Lanzamiento de la aplicación en la red.

2. Motivación

La motivación principal que me llevó a proponer esta temática es la clara ventaja que tiene un ordenador sobre un jugador de ajedrez, haciendo prácticamente imposible una victoria incluso para los jugadores más curtidos en el juego, pero la gracia está cuando compiten dos motores de ajedrez, ya que se ganan entre ellos como si de humanos se tratase. Al ser un juego con un número de posibilidades casi imposible de calcular esto hace que aparezcan motores nuevos de ajedrez y vengan a los anteriores. Esto nos quiere decir que un humano podría ganar a un motor de ajedrez, simplemente es menos probable ya que explora una cantidad de posibilidades menor de las que lo hace un ordenador. Me parece apasionante que todavía no se haya calculado por completo el juego y es por esto que decidí realizar esta investigación.

3. Introducción:

Para entender más sobre la informática en el ajedrez hemos realizado una investigación sobre los algoritmos más influyentes en el ajedrez a lo largo de la historia, para ello hemos codificado algunos algoritmos de búsqueda de árbol en problemáticas por turnos, esto lo hicimos con el objetivo de entender las previas resoluciones en el juego para poder sacar nuestras propias conclusiones a posteriori, para ello hemos creado una investigación en la que hemos codificado las siguientes funcionalidades.

- **Minimax:**

El algoritmo de Minimax es un método utilizado en la teoría de juegos para determinar la mejor estrategia en un juego de suma cero entre dos jugadores, considerando todas las posibles jugadas y minimizando las pérdidas máximas.

- **Minimax Alpha y Beta:**

El algoritmo de Minimax con poda alfa y beta optimiza la búsqueda en un árbol de juego al reducir el número de nodos evaluados, manteniendo los límites de los valores máximos y mínimos encontrados en cada nivel.

- **Turochamp:**

Turochamp fue un software de ajedrez desarrollado por Alan Turing en los años 1950. Aunque no era muy poderoso en términos de juego, fue uno de los primeros intentos de crear un programa de ajedrez utilizando una computadora, sentando las bases para el desarrollo de programas más avanzados en el futuro. Para ello hemos desarrollado la heurística que propone Turing para resolver calcular una posición.

- **Aprendizaje de reglas sin conocimiento previo:**

Para esta investigación hemos programado una red convolucional que aprende a reconocer jaques y jaques mate sin conocimiento alguno sobre el ajedrez. Para esto hemos entrenado una red convolucional que sea capaz de clasificar estas reglas.

- **Redes Convolucionales como función heurística:**

La parte final de la investigación contiene el uso de redes convolucionales con el objetivo de predecir el valor de una posición aprendiendo de un motor de ajedrez, para esto usamos el motor de ajedrez como función de evaluación para las posiciones y la red convolucional entrenará en base a estas evaluaciones. De esta forma estamos demostrando el aprendizaje reforzado desde una perspectiva diferente.

Para ejecutar los cuadernos que contienen las investigaciones acceder a:

<https://github.com/nenomg/ALPHAUPO-INVESTIGATION>

Con fin de exponer algunos de los algoritmos de ajedrez y la investigación realizada hemos creado una página web en la que podremos jugar contra varios motores de ajedrez, una base de datos con una red convolucional como heurística y un sistema de gestión para los usuarios y las partidas.

Para esto hemos preparado el siguiente enlace:

<https://e1fe-81-36-11-101.ngrok-free.app/public/>

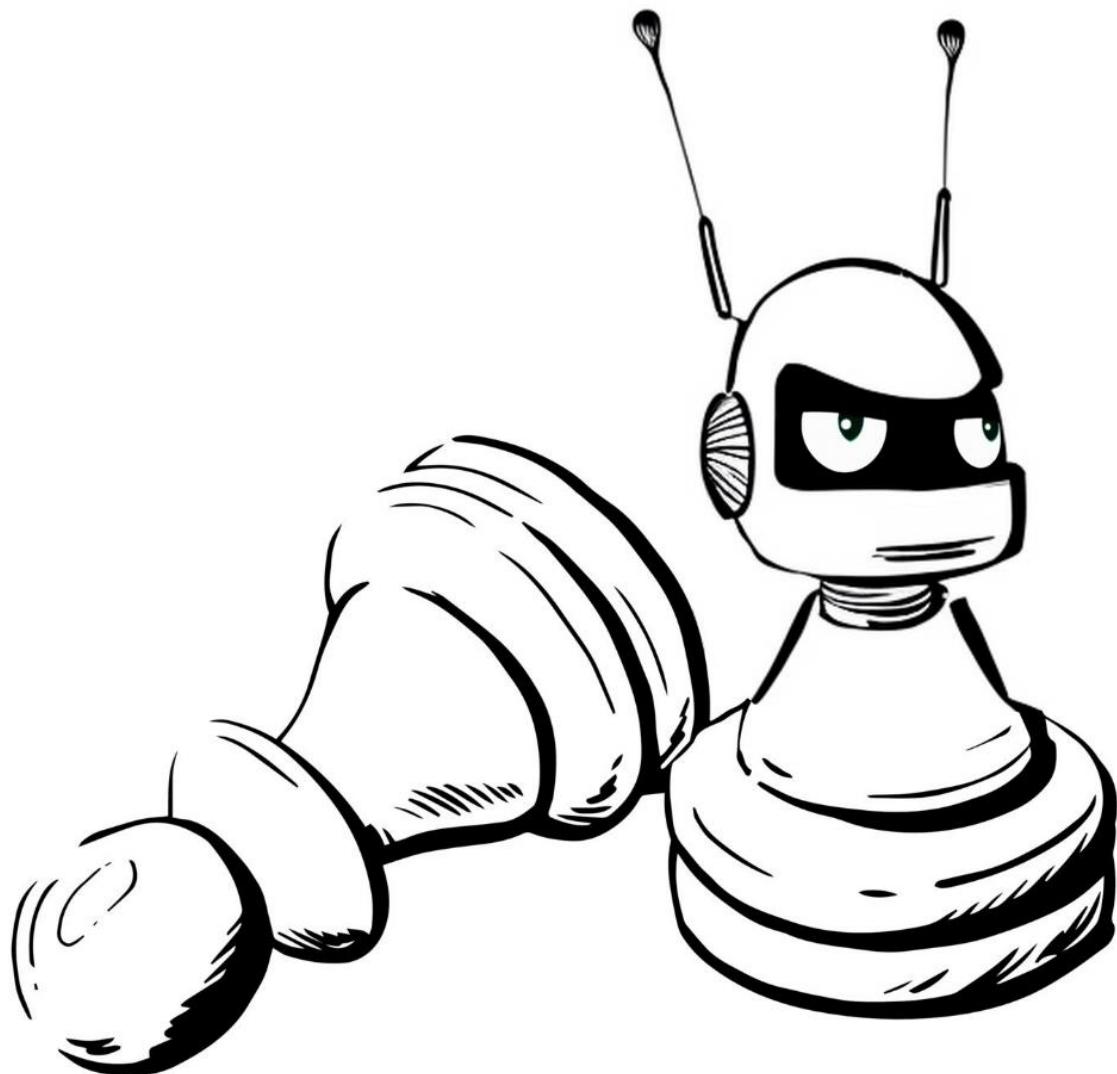
Si hubiera algún problema con el enlace proporcionado, el siguiente enlace contiene el vínculo actualizado:

https://github.com/nenomg/ALPHAUPO_WEBSITE

Hemos preparado los siguientes usuarios de administrador para acceder a esta y probar las funcionalidades:

- **Usuario:** pablodeolavide@alphaupo.es
Contraseña: Pablodeolavide0
Nombre: UPO
- **Usuario:** pablodeolavide1@alphaupo.es
Contraseña: Pablodeolavide1
Nombre: UPO1
- **Usuario:** pablodeolavide2@alphaupo.es
Contraseña: Pablodeolavide2
Nombre: UPO2

ALPHAUPO



**INVESTIGACIÓN SOBRE LA INFORMÁTICA
EN EL AJEDREZ
“ALPHAUPO”**

1. ¿QUÉ ES EL AJEDREZ?

El ajedrez es, desde un punto de vista material o físico, un juego de mesa que consta de un tablero cuadrado con 64 casillas, es decir: un tablero de 8x8 casillas. El tablero normalmente se distingue por dos colores, aunque es indiferente para la jugabilidad.

Se juega entre dos oponentes, con piezas de diferentes colores, que generalmente son blancas y negras, siendo indiferente, salvo que se trate de competiciones oficiales, su tamaño, volumen y peso.

Cada jugador comienza con un número idéntico de piezas colocadas de forma simétrica en el tablero, en total son 16 piezas para cada jugador y se colocan en las primeras dos filas desde el punto de vista de cada uno, con el cuadrado blanco en el flanco derecho de cada jugador.

El objetivo final es que, con el movimiento tasado y reglamentario de cada una de las piezas del tablero, crear estrategias de apoyo, defensa y ataque simultáneamente que permitan cercar y derrotar a la figura central del oponente: El Rey.

Las piezas que se usan son las siguientes:

Peones:



Figure 1. White pawn (classic)



Figure 7. Black pawn (classic)

Torres:



Figure 2. White rook (classic)



Figure 8. Black rook (classic)

Caballos:



Figure 3. White knight (classic)



Figure 9. Black knight (classic)

Alfiles:

Figure 4. White bishop (classic)



Figure 10. Black bishop (classic)

Reinas:

Figure 5. White queen (classic)



Figure 11. Black queen (classic)

Reyes:

Figure 6. White king (classic)



Figure 12. Black king (classic)

Cada una de las piezas se mueve de forma diferente sobre el tablero. Esta sería la posición inicial de una partida de ajedrez:

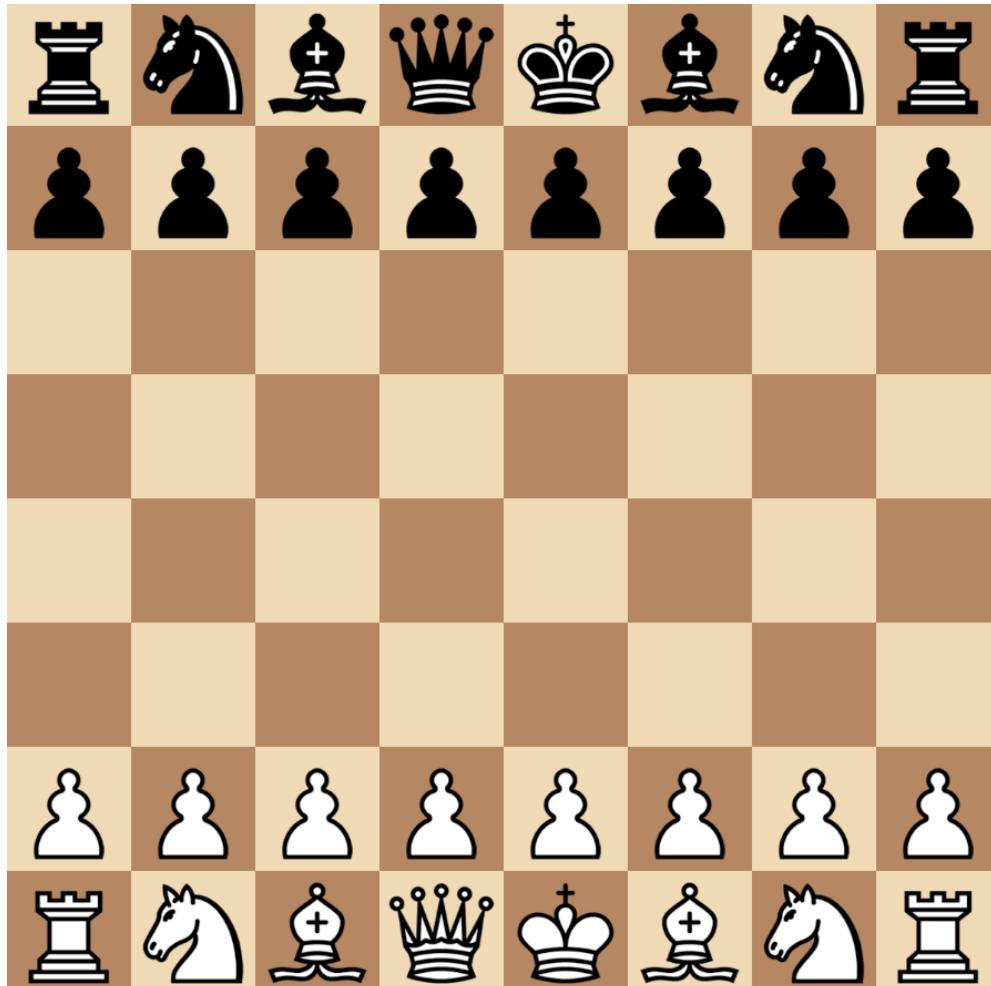


Figure 13: Chess Explorer

Así es como se iniciaría una partida, al inicio, el primer movimiento lo realizan las piezas blancas y en su turno mueve una de las piezas y termina el turno, seguido el jugador de piezas negras mueve una pieza y termina su turno y así hasta que termine la partida.

2. ¿CÓMO SE MUEVEN LAS PIEZAS?

2.1 Peones

Los peones en su primer movimiento podrán decidir si moverse dos casillas hacia delante o simplemente una, pero solo podrán moverse hacia delante dos casillas en el primer turno.

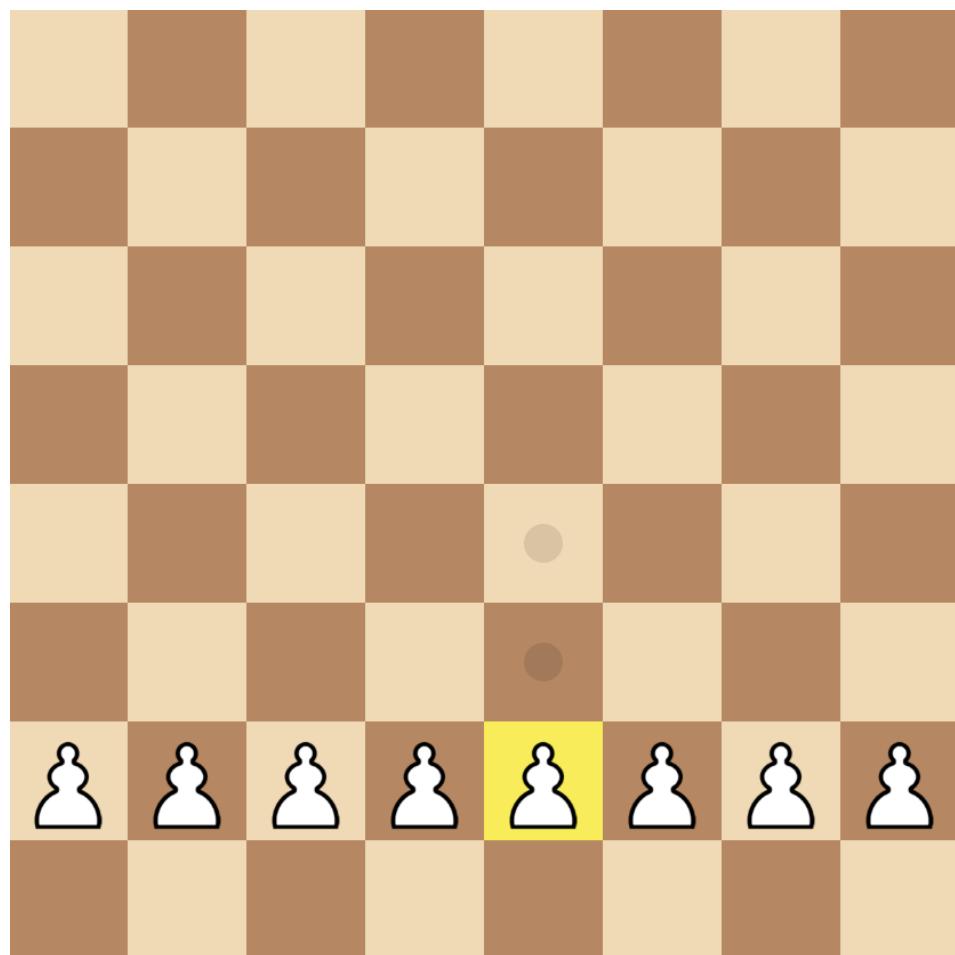


Figure 14: Chess Explorer

El resto de turnos solo podrá moverse hacia delante una vez y nunca podrá ir hacia detrás el peón:

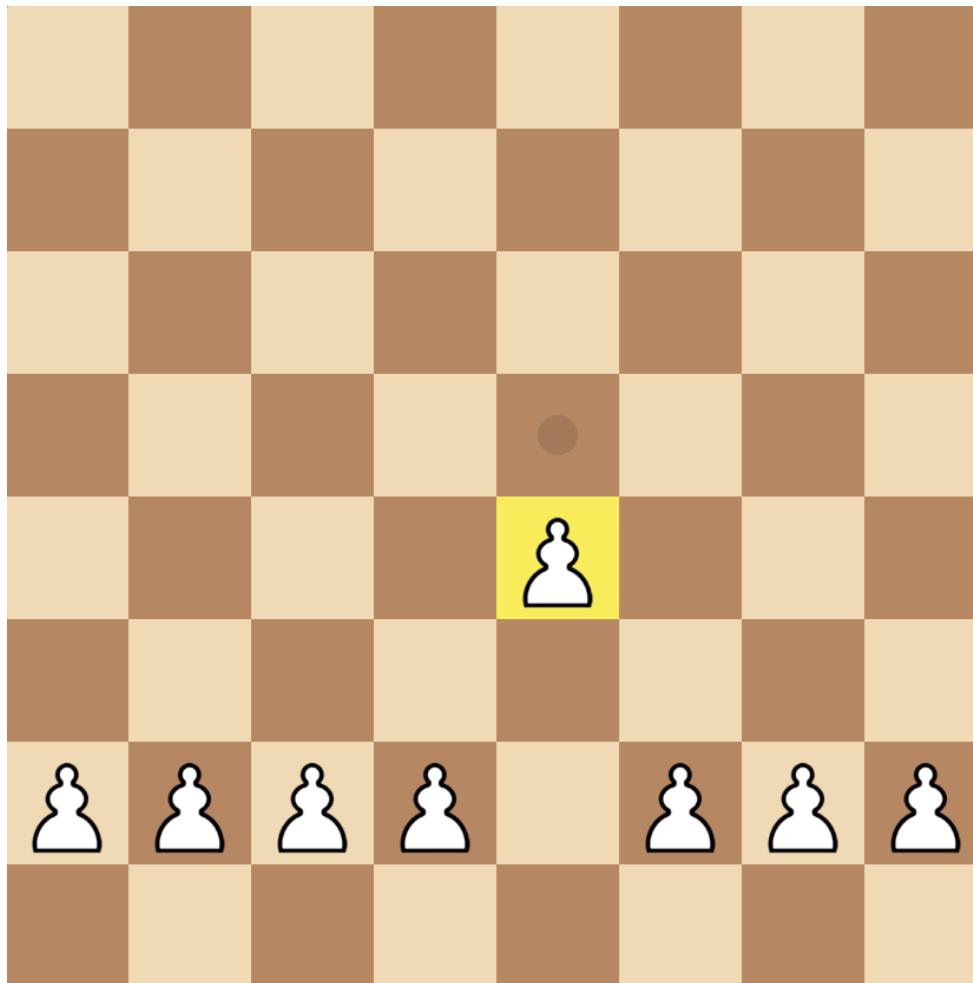


Figure 15: Chess Explorer

Cuando queremos comer la pieza del rival con un peón solo lo podremos hacer de manera diagonal una casilla hacia delante, nunca hacia atrás.

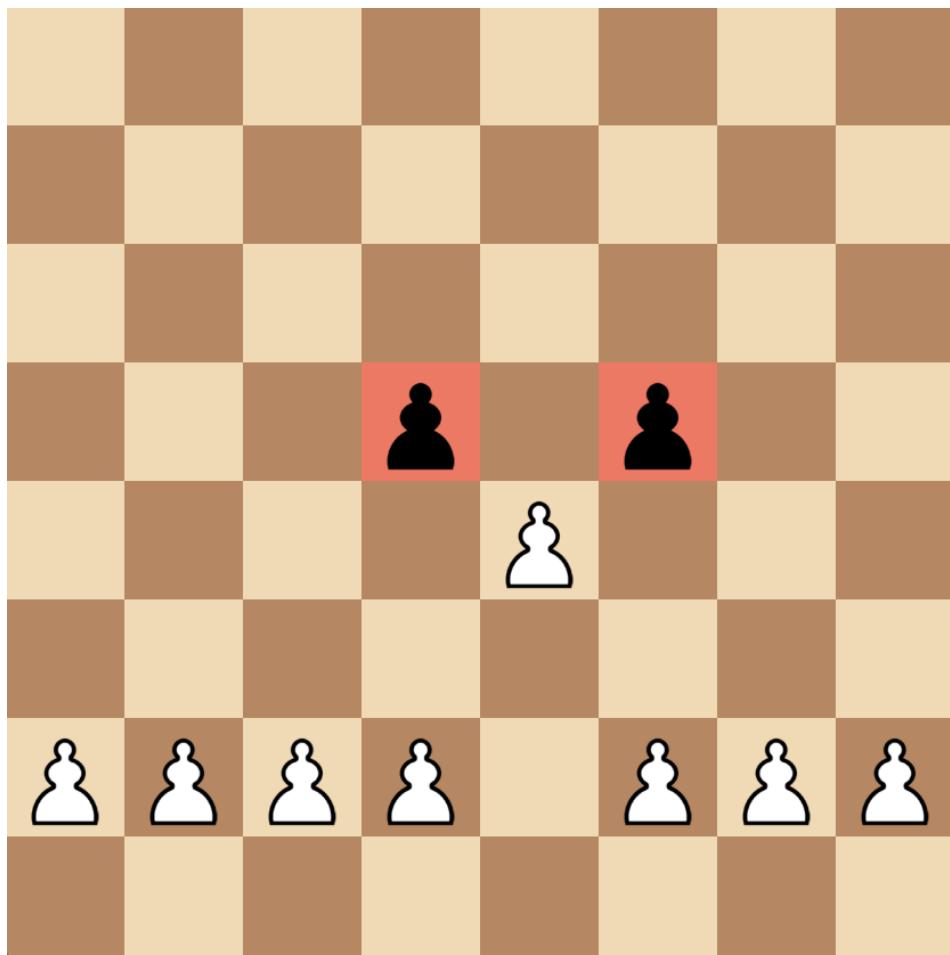


Figure 16: Chess Explorer

Otra peculiaridad del peón es la captura al paso (en francés en passant), un movimiento bastante enrevesado que no suele conocer las personas que no están relacionadas con el ajedrez, fue añadido en el siglo XV. Para realizar este movimiento nuestro peón debe de estar colocado tres casillas por delante de la posición inicial de un peón, el último movimiento del contrincante ha sido dos casillas hacia delante y el peón del rival se encuentra en la casilla izquierda o derecha de nuestro peón.

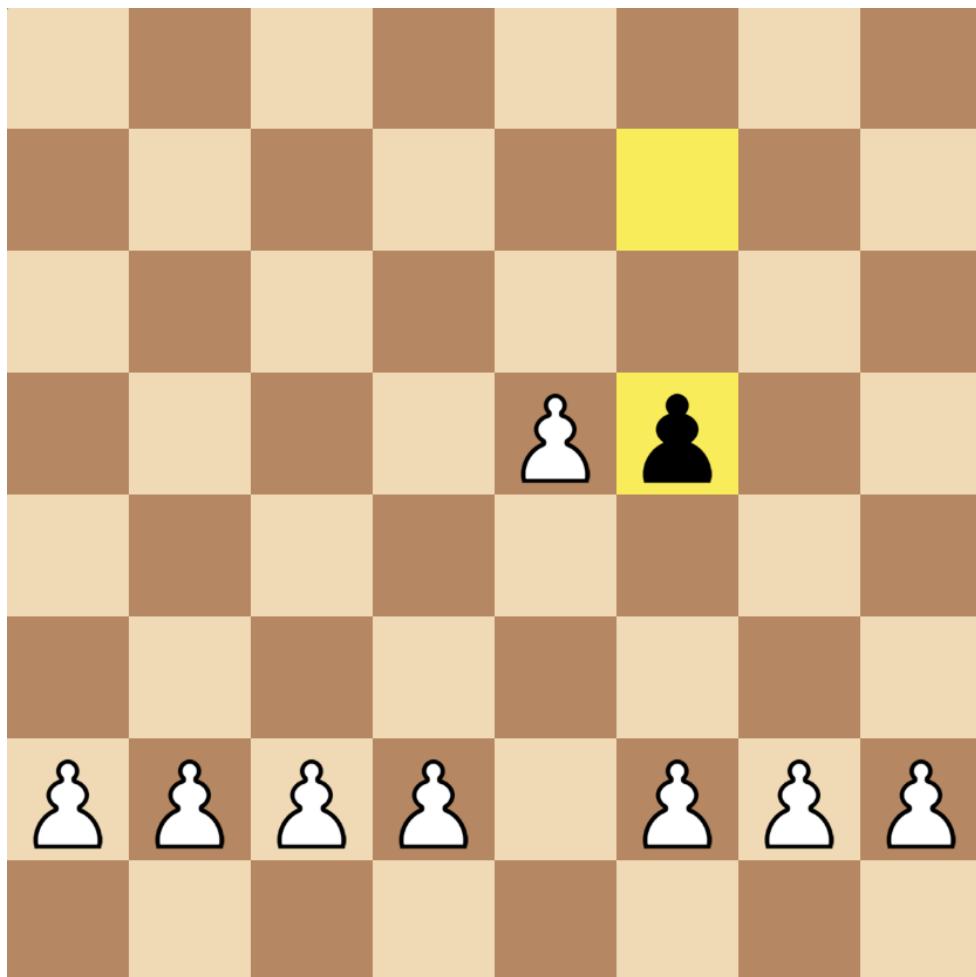


Figure 17: Chess Explorer

El último movimiento fue dos casillas hacia delante de las negras por lo que podremos comerlo como si hubiese movido una casilla hacia delante y lo mismo sucede si el peón negro estuviera colocado en la izquierda de nuestro peón y su último movimiento hubiera sido dos casillas hacia delante.

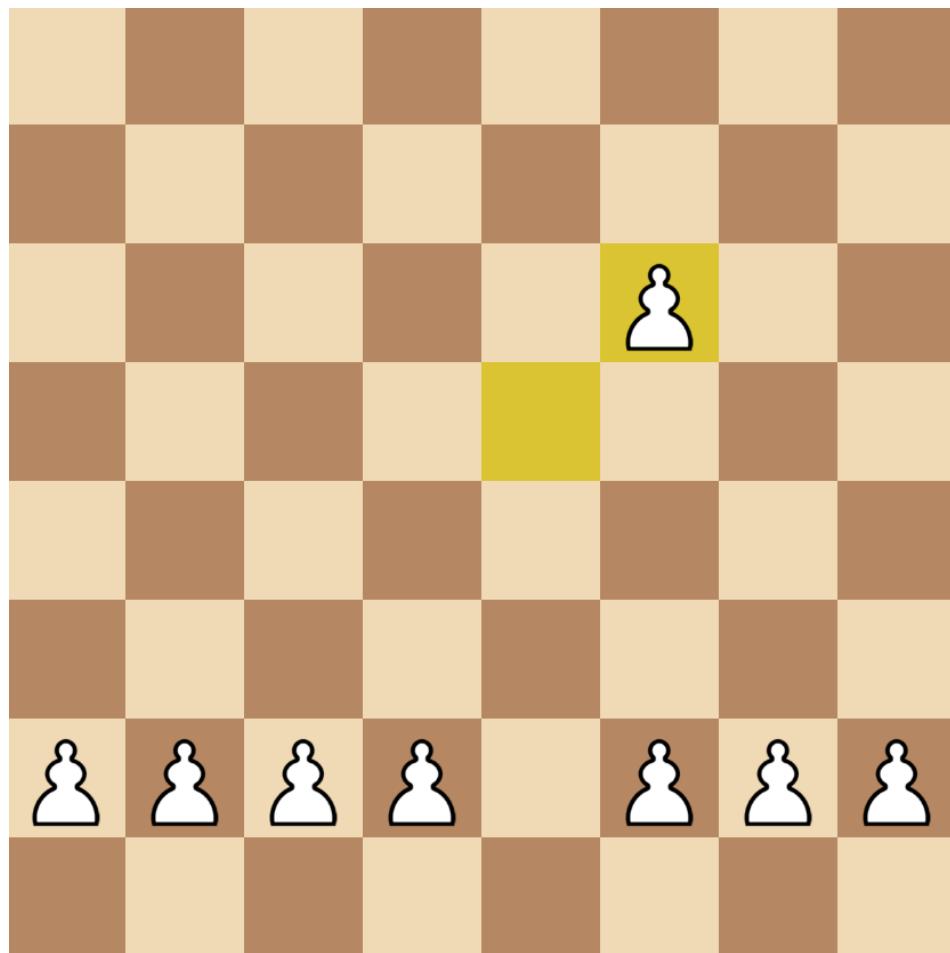


Figure 18: Chess Explorer

Por último, los peones al llegar a la última fila podrán coronar, es decir, este podrá convertirse en la pieza que desee el jugador exceptuando el rey.

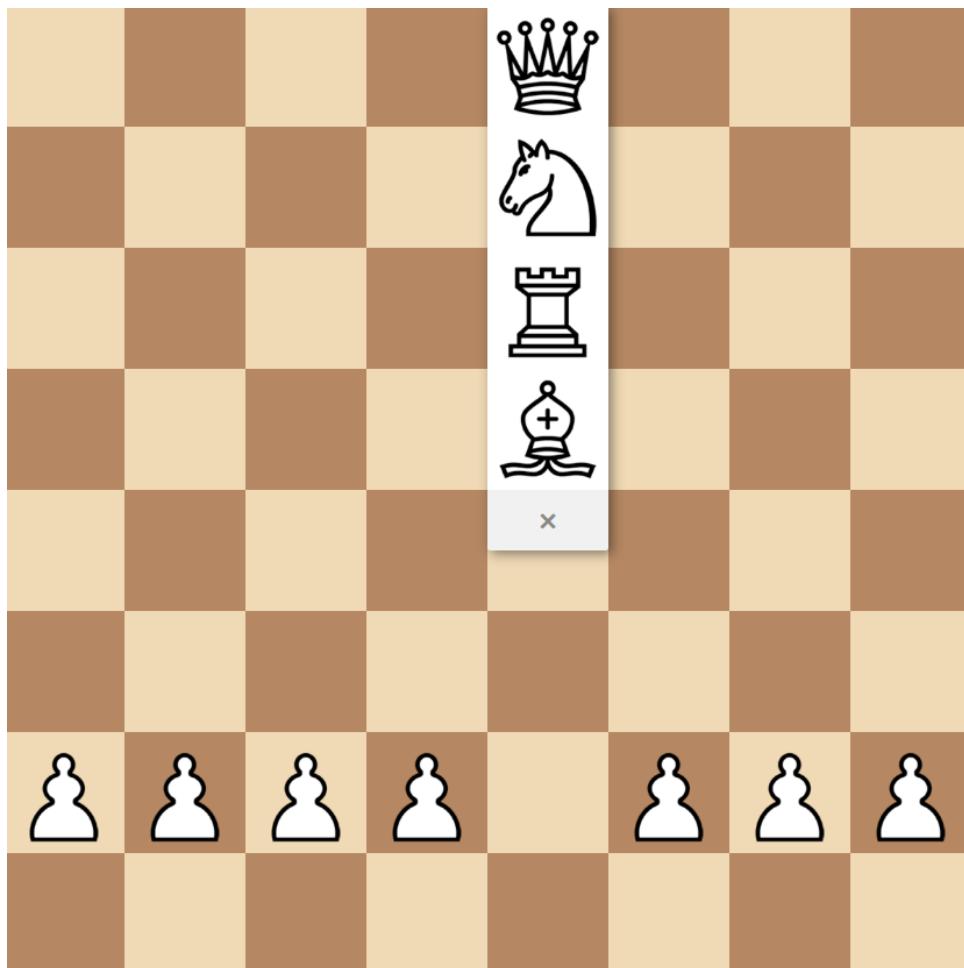


Figure 19: Chess Explorer

Al llegar a la última casilla podemos decidir en qué pieza convertir nuestro peón, en este caso lo haremos con una reina.

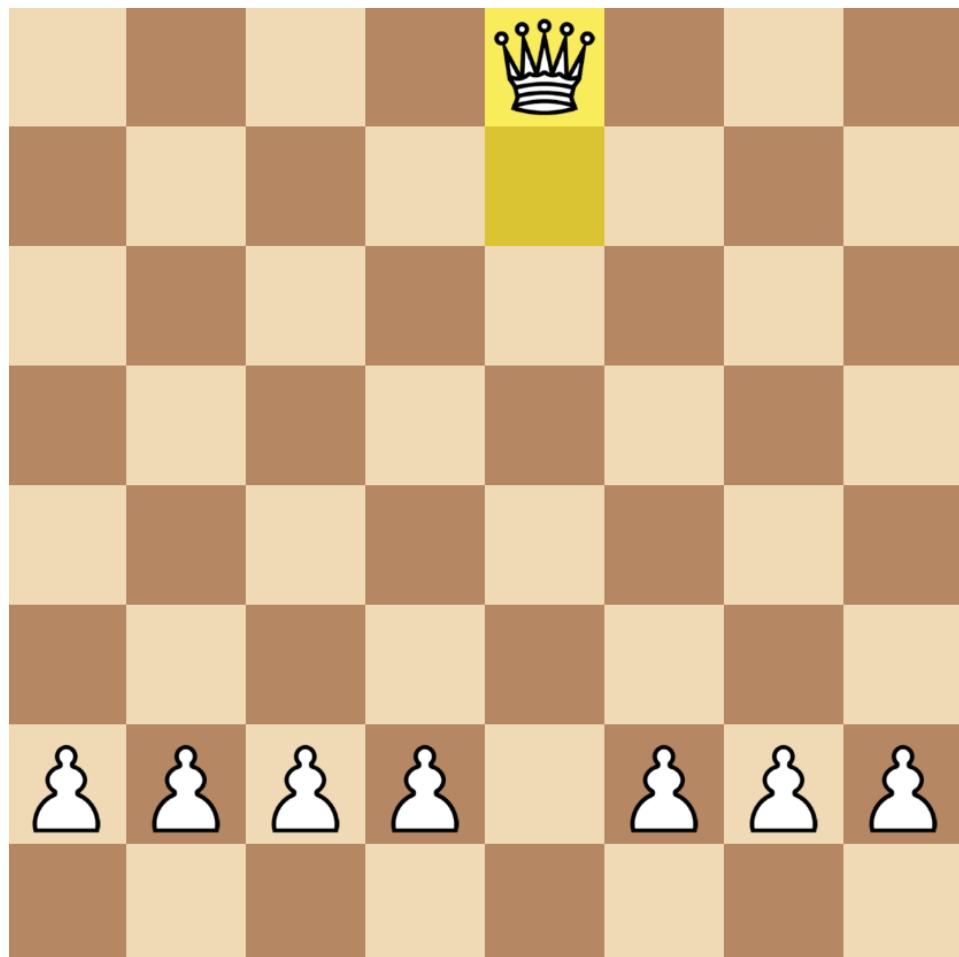


Figure 20: Chess Explorer

2.2 Alfiles

Los alfiles podrán desplazarse de manera diagonal por todo el tablero exceptuando si existe una pieza de nuestro color o una contraria, en ese caso podremos comerla y retirar la pieza del rival.

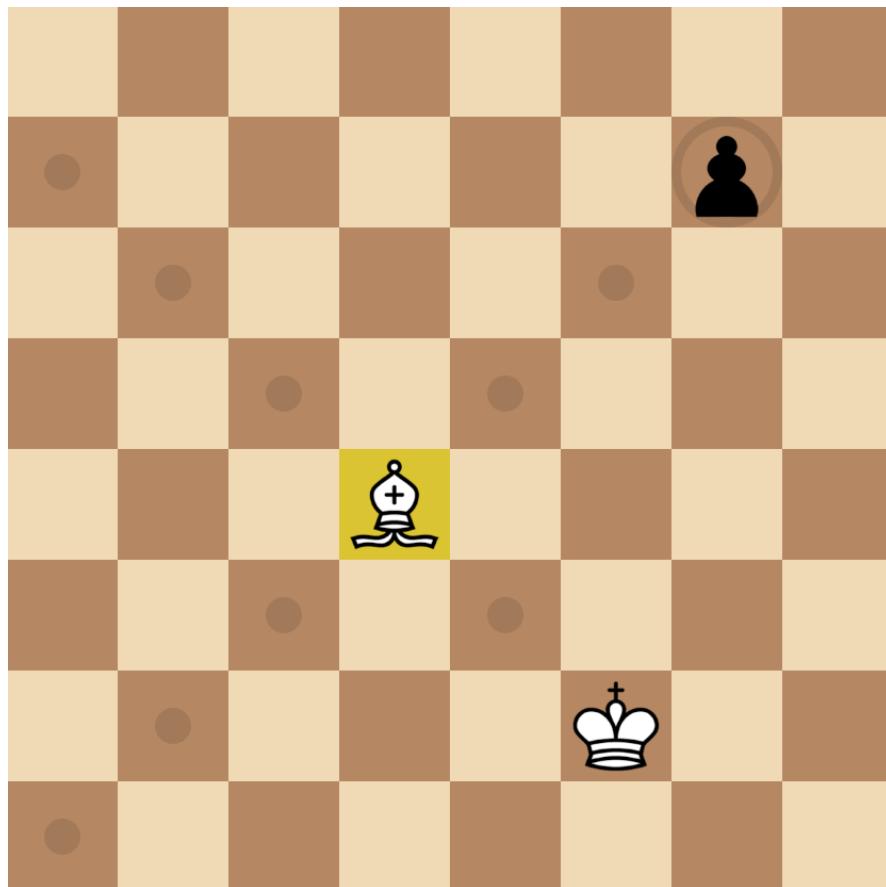


Figure 21: Chess Explorer

2.3 Caballos:

Los caballos son una de las piezas más peculiares, estos pueden saltar a las piezas y se mueven en forma de L, con dos casillas hacia una dirección y otra en la perpendicular. Un ejemplo de posibles casillas a las que mover el caballo sería el siguiente:

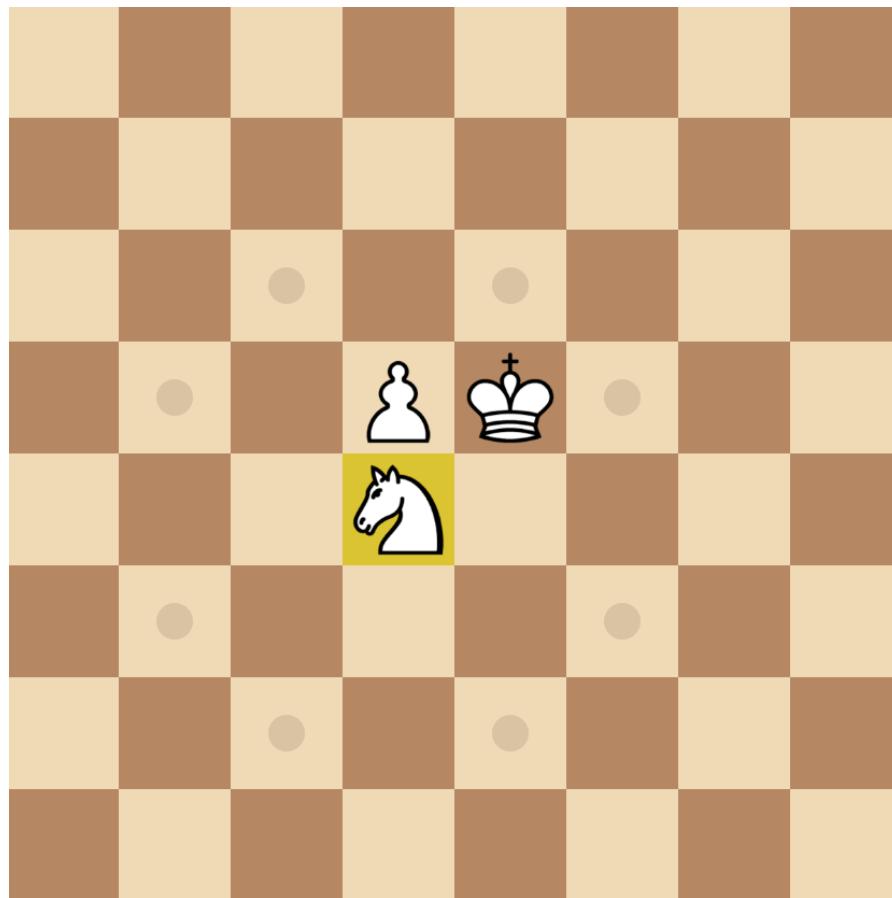


Figure 22: Chess Explorer

Si se encontrara una pieza rival en una de las casillas podríamos comer la pieza, si fuera de nuestro color no podríamos mover hacia allí.

2.4 Torres

Se mueven por todo el tablero como los alfiles, pero estas solo podrán moverse en línea recta por las filas y las columnas del tablero.

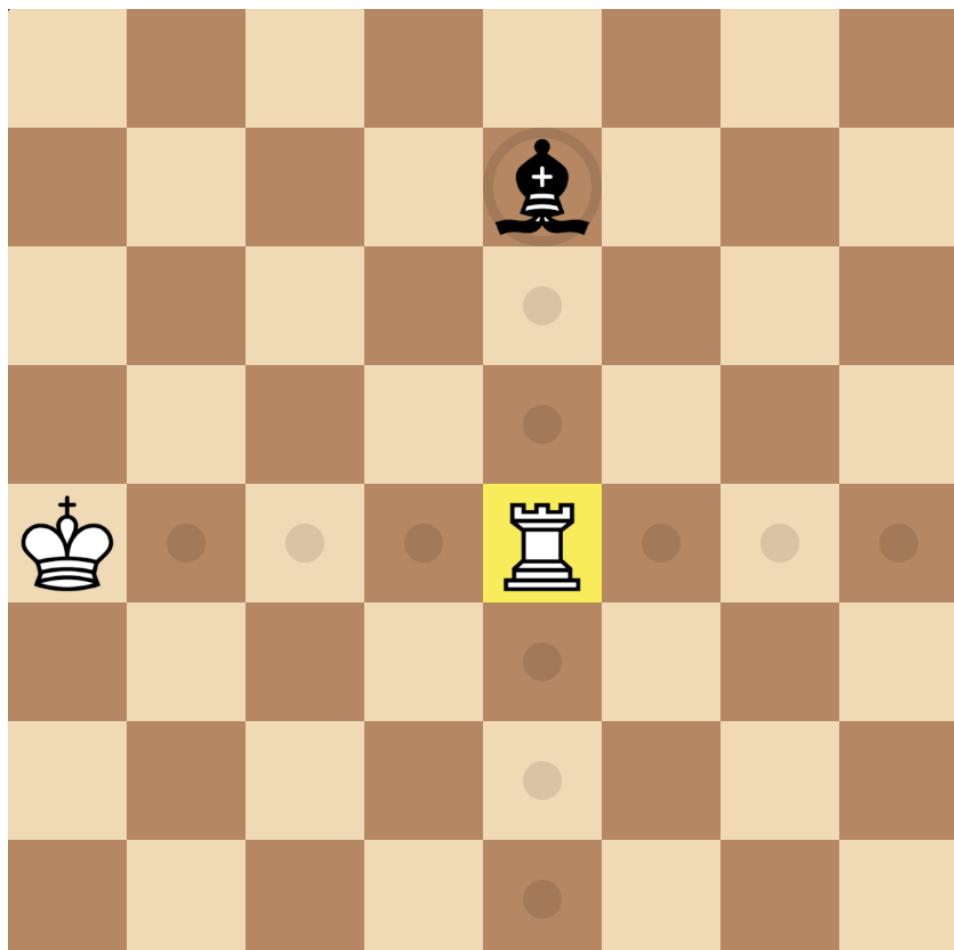


Figure 23: Chess Explorer

2.5 Reinas

La reina es la pieza que puede ocupar más casillas sobre el tablero, ya que esta se mueve como un alfil y una torre al mismo tiempo, por lo que podrá moverse por las diagonales, las verticales y las horizontales del tablero.

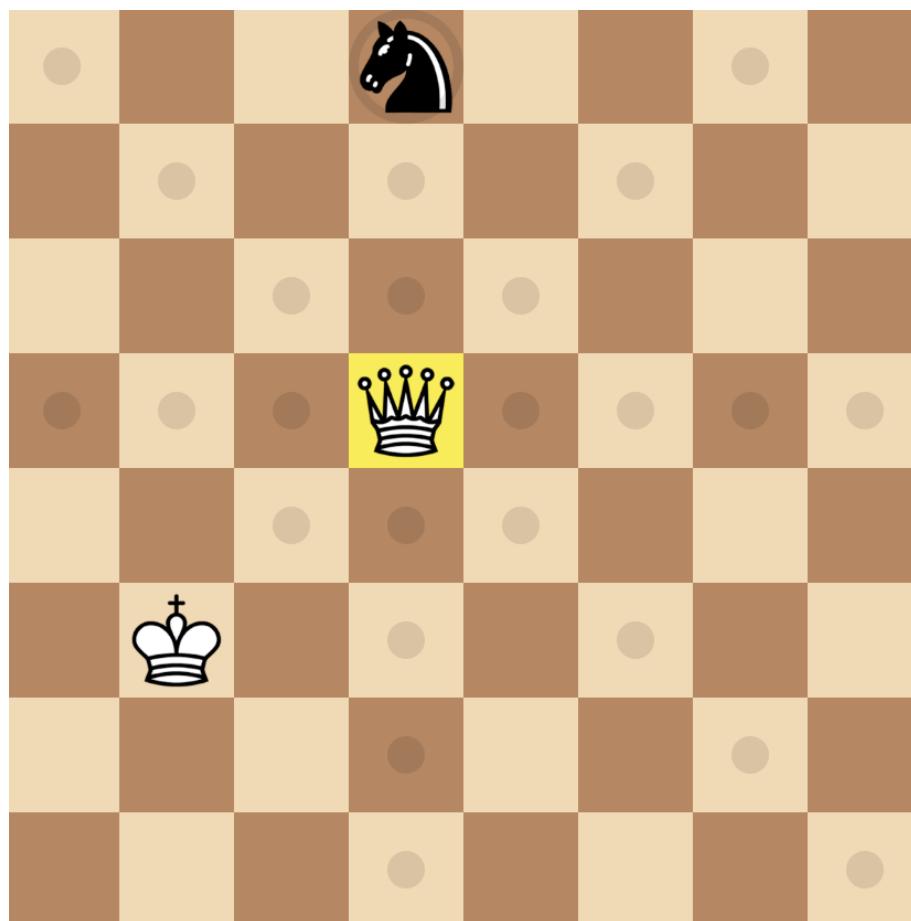


Figure 24: Chess Explorer

2.6 Rey

El rey es la pieza más importante, ya que si nos dan jaque mate hemos perdido la partida, explicaré lo que es un jaque mate más adelante. El rey puede moverse únicamente una casilla en la diagonal, horizontal y vertical de la posición en la que se encuentre.

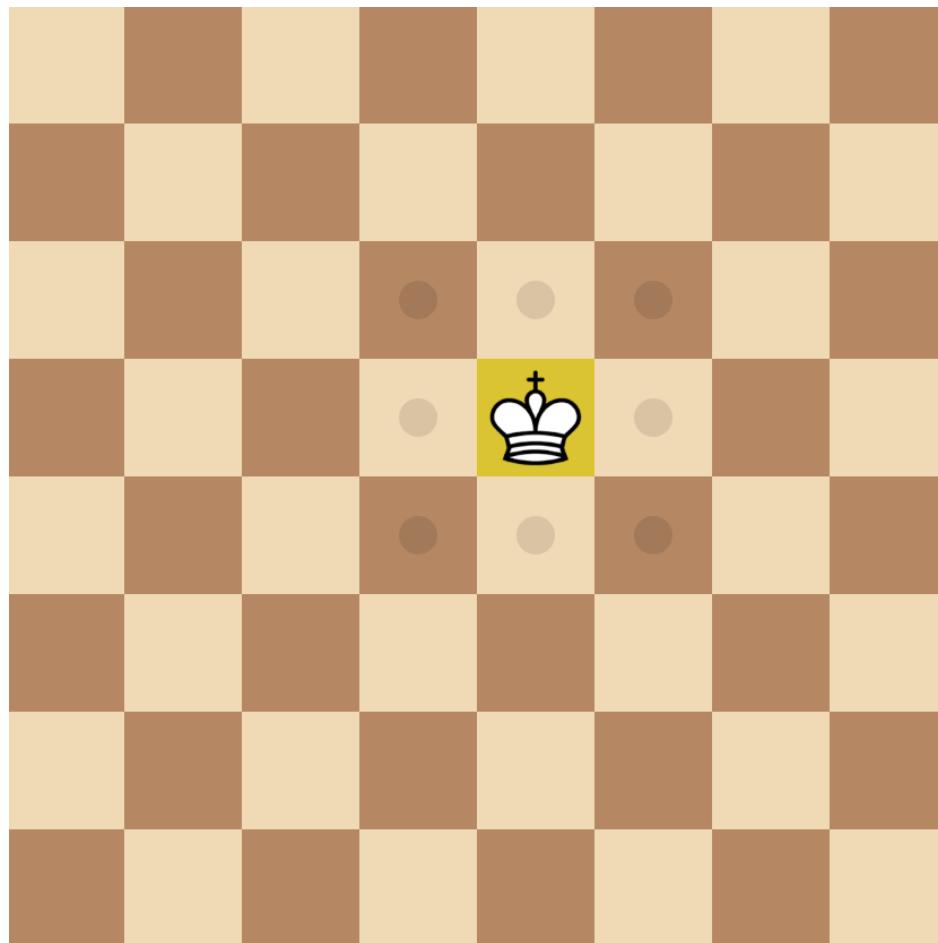


Figure 25: Chess Explorer

3. REGLAS

Están instituidas por la Federación Internacional de Ajedrez (FIDE), creando un marco normativo internacional universal.

El Reglamento de Ajedrez de la FIDE rige el juego sobre el tablero y consta de dos partes: 1. Las reglas básicas de juego y 2. Las reglas de competición.

El vigente Reglamento de Ajedrez, se adoptó en el 88º Congreso de la FIDE celebrado en Göynük (Turquía), y entró en vigor el 1º de enero de 2018.

3.1 Jaque

Ocurre cuando el rey está amenazado por alguna de las piezas contrarias, pero este puede escapar de la amenaza en el siguiente turno, un ejemplo de jaque:

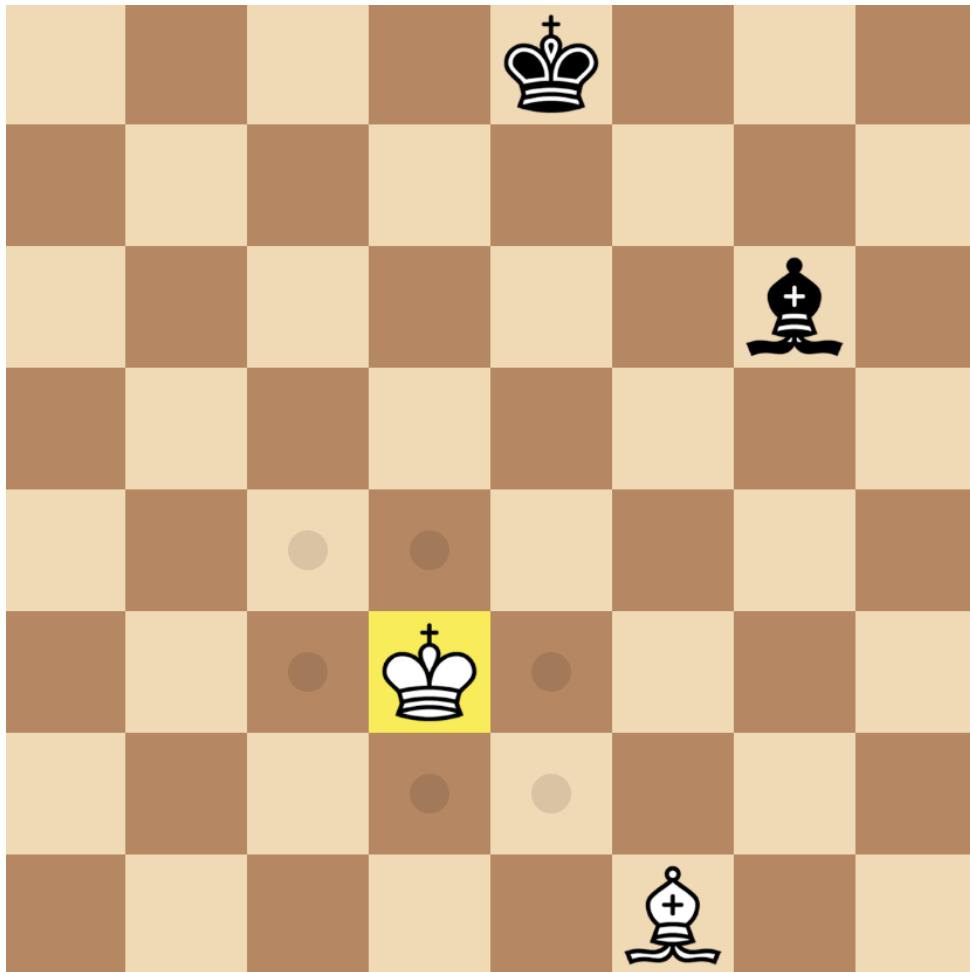


Figure 26: Chess Explorer

Nuestro rey no podrá quedar en jaque para el siguiente movimiento por lo que debemos moverlo para el siguiente turno, por lo tanto, no podremos mover el alfil.

3.2 Jaque Mate

Es el objetivo del juego, el que de jaque mate al rival es el jugador ganador, cuando damos jaque mate el rey contrario debe estar amenazado (Jaque) por alguna de nuestras piezas y el rey rival no puede tener escapatoria alguna.

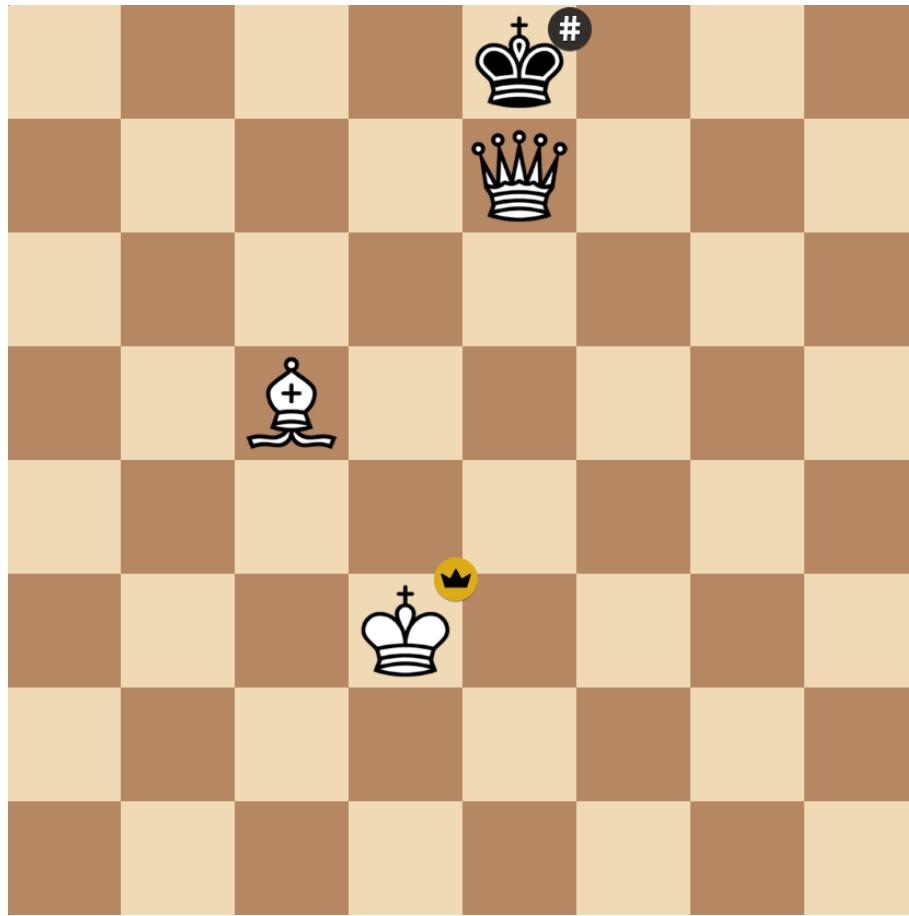


Figure 27: Chess Explorer

El rey negro no tiene escapatoria y está amenazado, luego la victoria será para las piezas blancas y ganan la partida.

3.3 Rey ahogado

Ocurre cuando el jugador no tiene ningún movimiento posible y el rey no está en ninguna amenaza, esto significa que la partida termina en tablas, es decir empate.

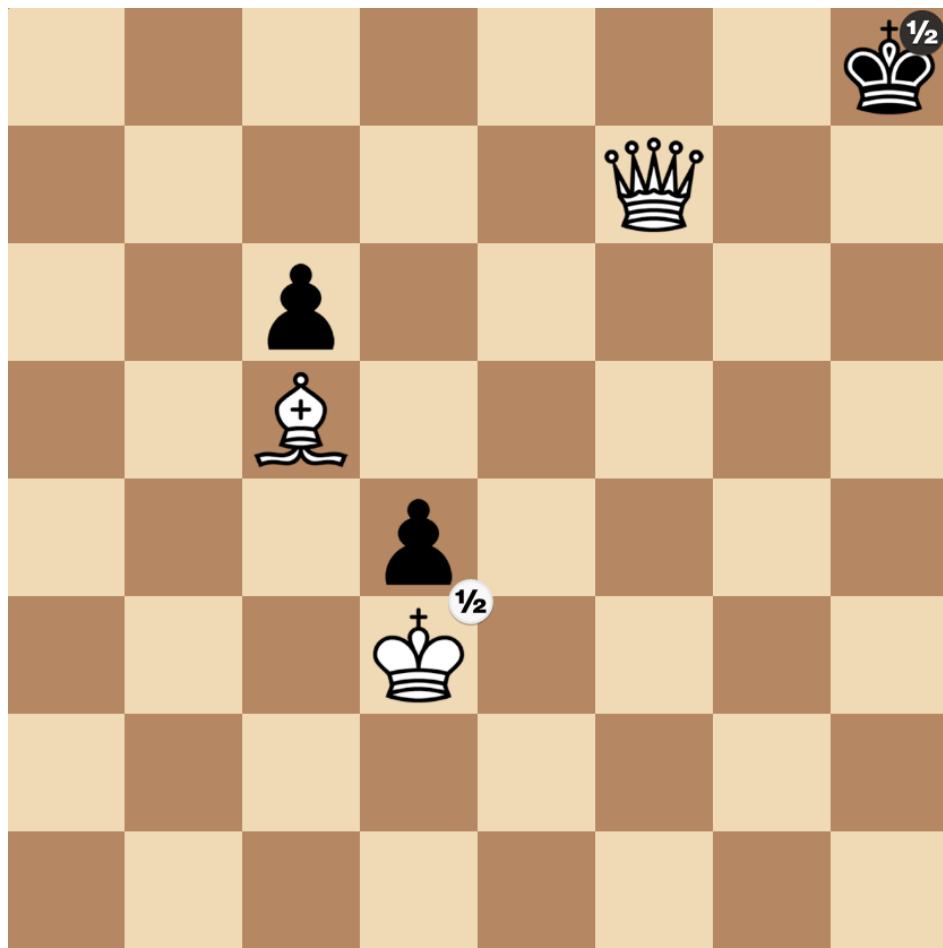


Figure 28: Chess Explorer

3.4 Repetición de movimientos

Cuando se repite la misma posición 3 veces en la partida sucede una repetición de movimientos y finaliza la partida.

3.5 Acuerdo mutuo

Una partida puede finalizar por un acuerdo pactado entre ambos de los jugadores.

3.6 Enroque:

El enroque involucra al rey y a la torre y solo puede suceder si ambos no se han movido ninguna casilla desde el comienzo del juego. Para que suceda un enroque, las casillas entre el rey y la torre deben estar descubiertas y las casillas en las que vamos a colocar nuestro rey no pueden estar atacadas por ninguna pieza rival, existen dos tipos de enroque, el enroque corto y el enroque largo, ya que hay una casilla más de diferencia entre una de las dos torres y el rey al comenzar la partida.

3.6.1 Enroque corto

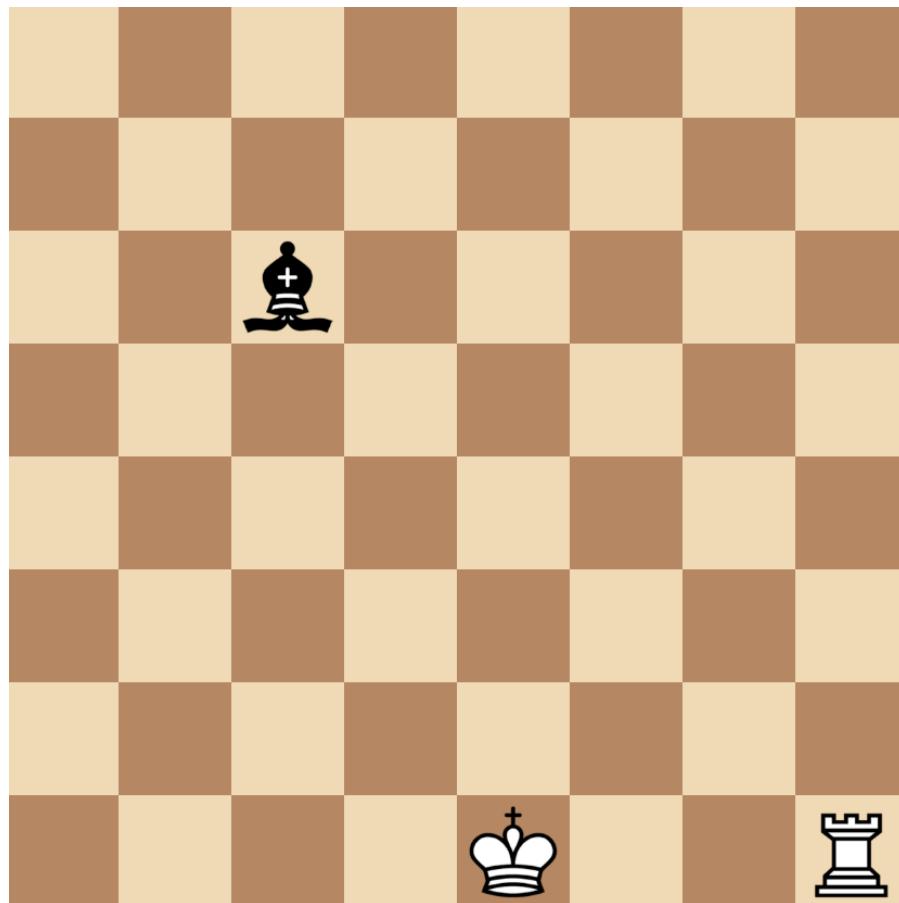


Figure 29: Chess Explorer

En esta posición podríamos realizar un enroque corto ya que no hay ninguna pieza entre el rey y la torre y ninguna de las casillas en las que se coloca el rey y la torre están amenazadas, luego el resultado de hacer un enroque corto es el siguiente:

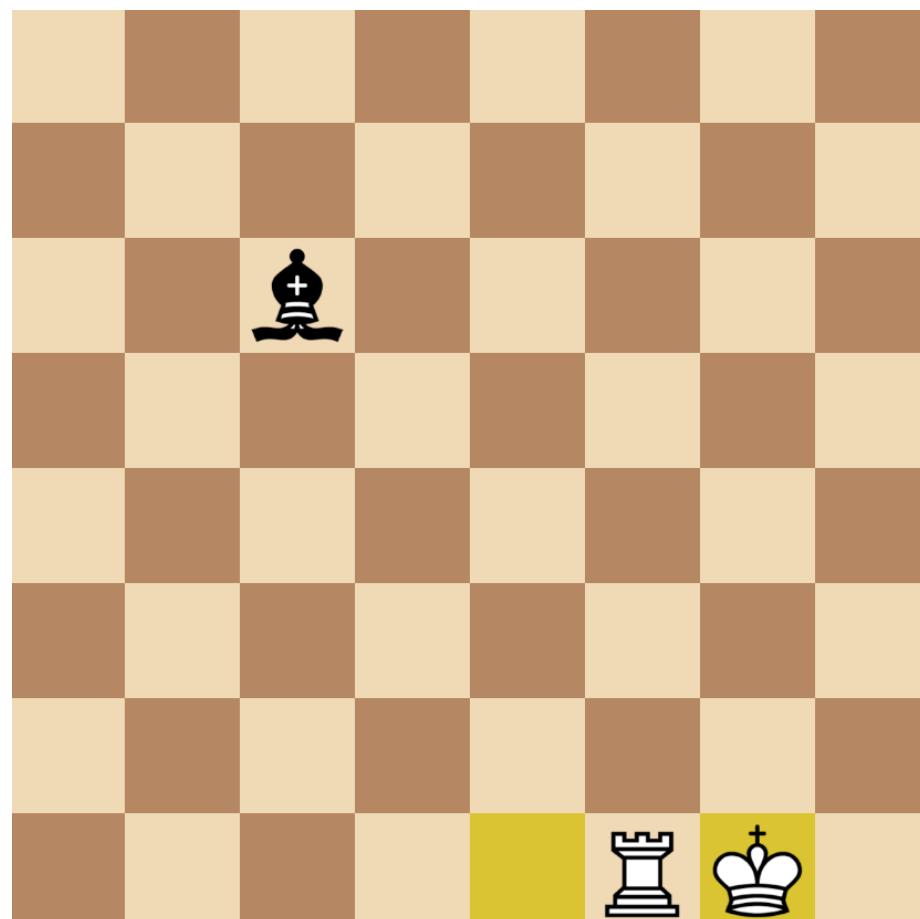


Figure 30: Chess Explorer

3.6.2 Enroque largo

Para el enroque largo sucede lo mismo, pero en este caso colocaremos al rey dos casillas hacia la izquierda y la torre una casilla a la derecha del rey:

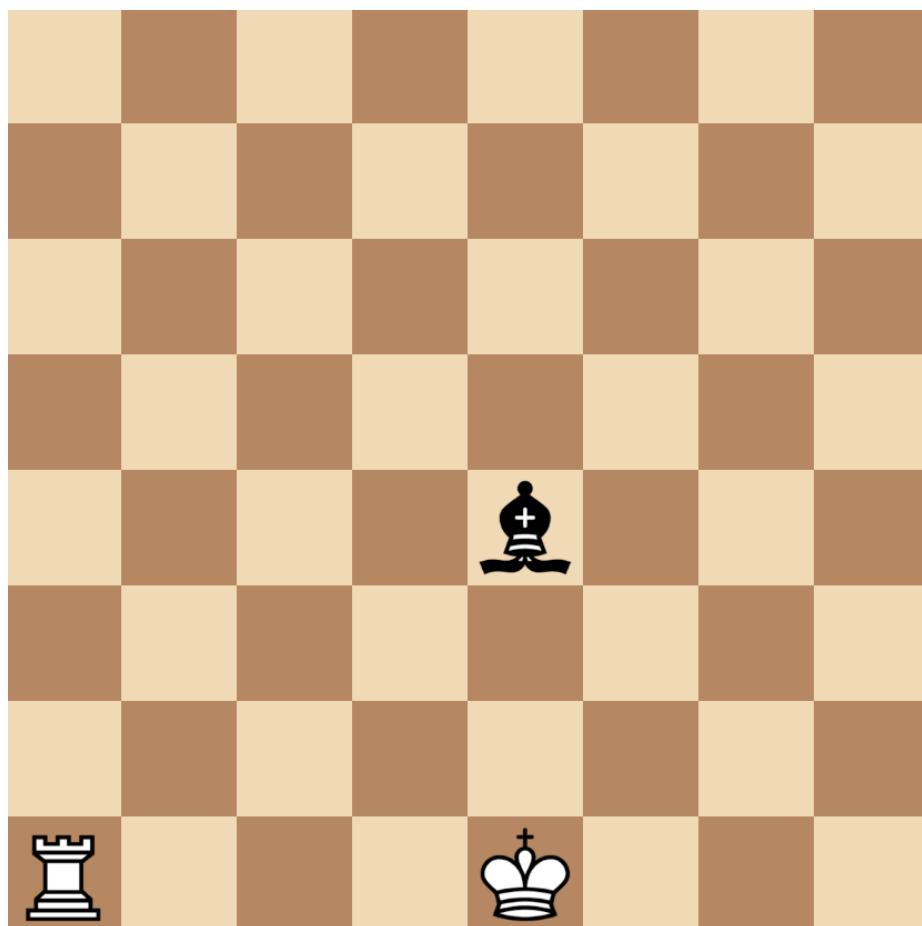


Figure 31: Chess Explorer

En esta posición podemos realizar el enroque largo y el resultado sería el siguiente:

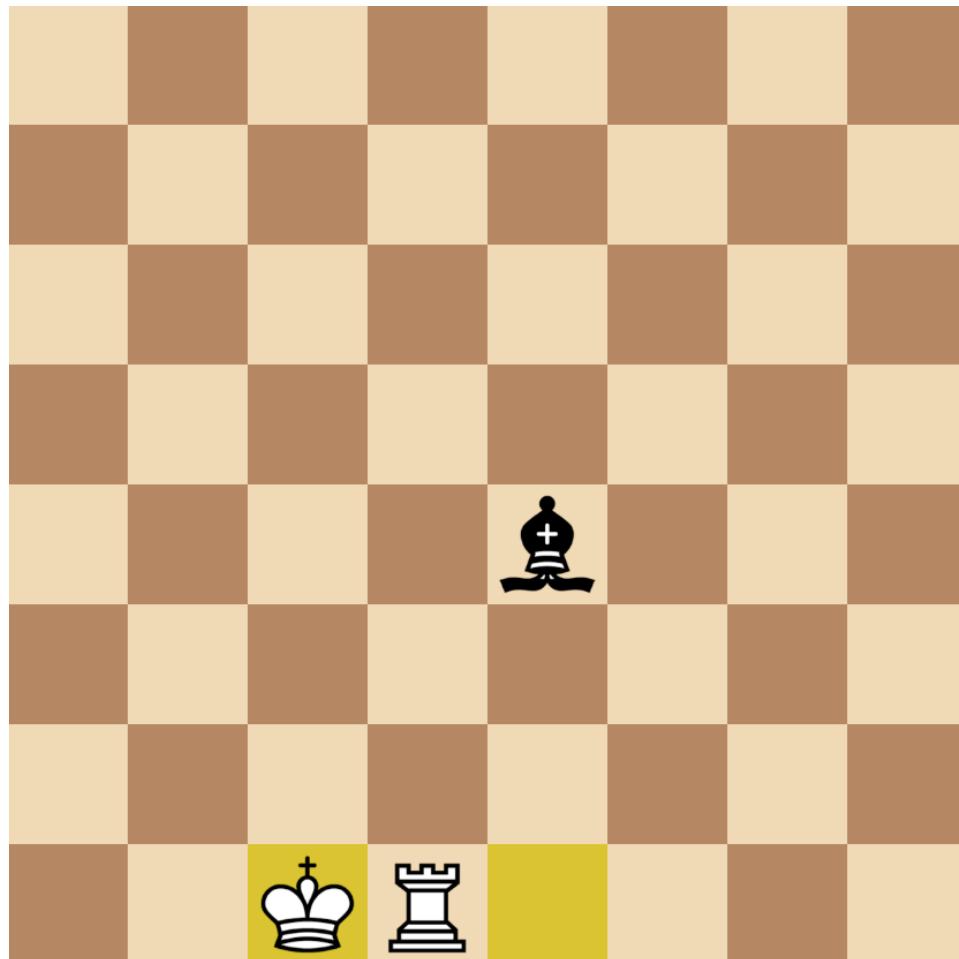


Figure 32: Chess Explorer

4. HISTORIA DEL AJEDREZ

Para comprender la trascendencia del juego a lo largo de los siglos, debemos hacer una breve sinapsis de su Historia, e hitos antiguos y modernos más relevantes:

4.1 Origen:

Incluso a día de hoy es imposible ubicar con certeza los orígenes de este juego de mesa, pero las investigaciones históricas apuntan a que se originó en India alrededor del siglo VI d.C.: Dicho juego se llamaba “Chaturanga” y ya por ese entonces tenía muchas similitudes con el juego actual.

“Chaturanga” es una palabra cuyas raíces vienen del sánscrito y significa “cuatro partes.”: Al decir cuatro partes hace referencia al ejército indio de aquella época, ya que estaba compuesto por elefantes, caballos, carros y soldados a pie. Estas eran las piezas del “Chaturanga” y se movían de forma muy parecida a la que lo hacen las piezas actuales.

Se estima que alrededor del siglo VII el ajedrez se extendió por Persia y desde allí se introdujo al mundo Islámico. Durante la Edad Media, avanzó su expansión hasta llegar a Europa donde se hizo muy popular y se hicieron diversos cambios tanto en las reglas como en el diseño de las piezas hasta llegar a las actuales.

En el siglo XIX, su popularidad se expandió de manera exponencial y se convirtió en un deporte profesional celebrándose los primeros torneos internacionales, hasta alcanzar su actual nivel, donde es uno de los juegos de mesa más practicados, si no es el que más.

5. TEORÍA EN EL AJEDREZ

La teoría o los estudios acerca del ajedrez comienzan aproximadamente en el siglo XV, si bien no fue hasta el siguiente siglo cuando empezaron a aparecer los verdaderos teóricos que analizaban las aperturas, estructuras y más factores relevantes a la hora de evaluar una posición. En esa época, aparecieron increíbles jugadores como el sacerdote español Ruy López (Zafra, 1530, Lima 1580/1590), creador de una de las aperturas con mayor índice de victorias incluso a día de hoy., o Gioacchino Greco (1600 - 1634) que destacó tras la desaparición de Ruy Lopez, y que fue uno de los pioneros en el estudio de la teoría, nacido cerca de Cosenza en la región de Calabria.

Como podemos comprobar, España fue una de las naciones pioneras en la introducción de la teoría de aperturas, habiendo introducido la llamada “apertura española”, jugada por grandes maestros a nivel de competición repetidas veces y bastante popular en los torneos de la última década: Una apertura que fue introducida por el maestro Ruy López antes citado y del que dejamos la siguiente muestra:

Apertura Ruy López: 1. e4 e5 2. Nf3 Nc6 3. Bb5

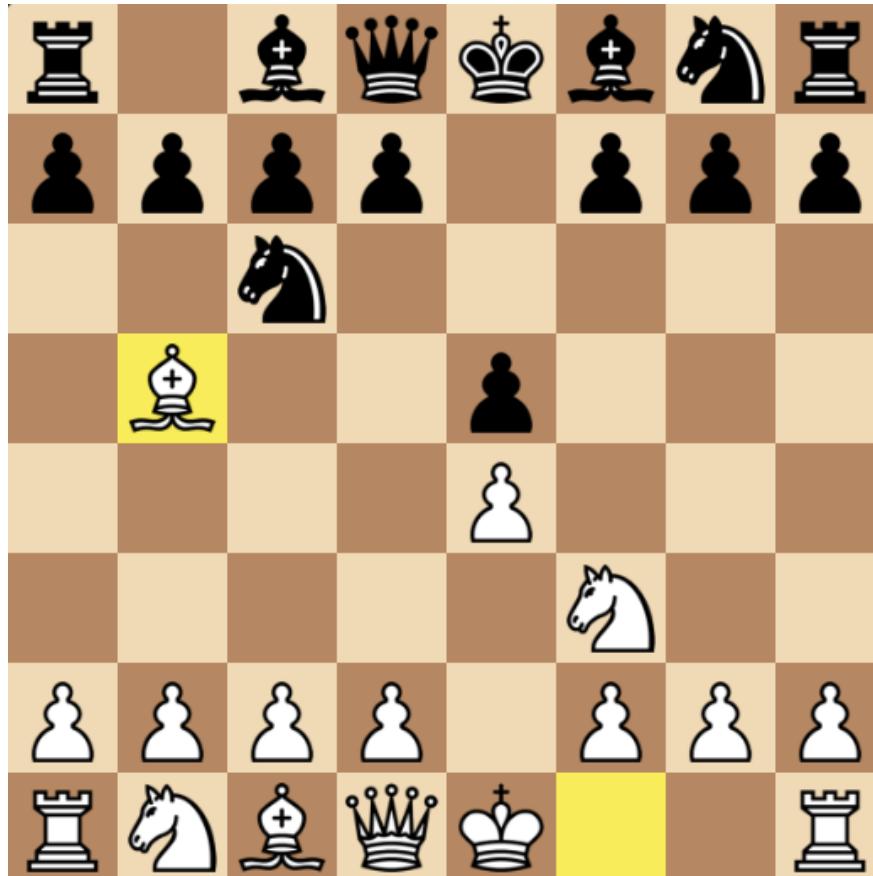


Figure 33: Chess Explorer

Ya en el siglo XIX resulta obligado mencionar al norteamericano Paul Morphy (1837 – 1884), sin duda uno de los padres del ajedrez moderno. Nace en Nueva Orleans, el 22 de junio de 1837, y desde los 8 años jugó innumerables partidas. No sólo destacaba en ajedrez Morphy también era abogado y su alta capacidad mental le permitía hacer cosas como memorizar por completo el Código Civil de Luisiana, su Estado natal.

Su alta comprensión del juego y habilidad fueron completamente notorias, haciéndolo ver como un hombre adelantado a su época: Y así era, ya que en sus partidas era cuando por primera vez empezamos a ver a un jugador hacer estructuras modernas y calcular tácticas complejas.

Su estilo de juego era completamente agresivo, haciendo sacrificios de múltiples piezas en muchas de sus jugadas.

Es notorio que tenía un nivel de compresión superior en aquel entonces y es nombrado para muchos como uno de los padres del ajedrez moderno.

Creó la “Defensa de Morphy” y utilizaba reiteradamente la apertura española. Para Richard Rèti, fue el primer jugador en entender la base posicional de ataque y Viswanathan Anand y Bobby Fischer colocan a este jugador como uno de los 10 mejores de la historia. Fischer, de hecho, llegó a afirmar que Paul Morphy era “El jugador más preciso que jamás haya existido”. Gary Kasparov el reconocido excampeón del mundo colocaba a Morphy como el “preursor del ajedrez moderno”.

En 10 de julio de 1884, con tan sólo 47 años, y habiendo participado en campeonatos en Europa, donde demostró su agilidad de pensamiento, falleció en su baño a causa de una apoplejía.



Figure 34: Paul Morphy versus Johann Jacob Lowenthal

Paul Morphy era y aún es todo un ídolo para muchos: Y es uno de mis jugadores favoritos: su manera de pensar era diferente y nos dejó problemas tan enigmáticos que aún se siguen planteando a día de hoy. Por ejemplo, este problema que plantea Morphy en el que las blancas dan jaque mate en dos, y que revela su enorme capacidad de análisis:

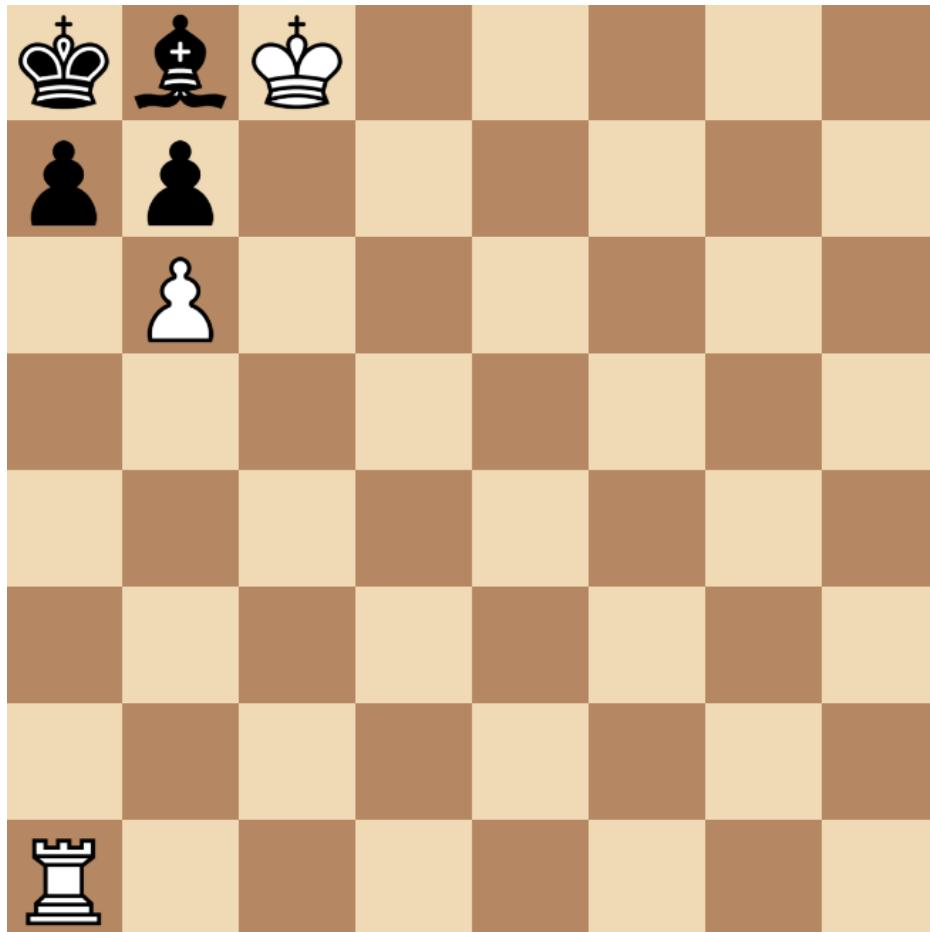


Figure 35: Chess Explorer

La solución es mover la torre a a6, ya que si comen con el peón negro de b7 ($bxa6$) avanzamos el peón blanco de b6 y damos jaque mate ($b7++$), si no decide comer la torre solo nos queda el alfil negro que mueva a donde se mueva no puede frenar a la torre para dar jaque mate en a7. Más adelante, llegaron más perfeccionistas del juego y teóricos del juego que mejoraron los conceptos actuales: Estamos hablando del siglo XX, que fue cuando el ajedrez se empezó a jugar con mayor fundamento. De esta época debe destacarse a José Raúl Capablanca, un jugador cubano que en su época fue apodado como “El Mozart del ajedrez”, y cuyo impacto en la teoría fue evidente y se demuestra en partidas como la que jugó contra Frank James Marshall otro de los mejores jugadores de la época:

Esta partida se jugó en Nueva York en 1918 y se planteaba el “Ataque Marshall” la famosa variante ofensiva del jugador contrario a Capablanca, y cuya apertura es una rama de la variante “Ruy Lopez”.

En esta partida, Capablanca realizó una extraordinaria defensa con la que logró contrataracar y vencer al famoso jugador. Capablanca jugaba con las piezas blancas.

Por su relevancia, y para los más puristas, adjunto la partida en notación algebraica

Frank James Marshal vs José Raúl Capablanca – Nueva York – 23/10/1918

1.e4 e5 2.Nf3 Nc6 3.Bb5 a6 4.Ba4 Nf6 5.O-O Be7 6.Re1 b5 7.Bb3 O-O 8.c3 d5 9.exd5 Nxd5 10.Nxe5 Nxe5 11.Rxe5 Nf6 12.Re1 Bd6 13.h3 Ng4 14.Qf3 Qh4 15.d4 Nxf2 16.Re2 Bg4 17.hxg4 Bh2+ 18.Kf1 Bg3 19.Rxf2 Qh1+ 20.Ke2 Bxf2 21.Bd2 Bh4 22.Qh3 Rae8+ 23.Kd3 Qf1+ 24.Kc2 Bf2 25.Qf3 Qg1 26.Bd5 c5 27.dxc5 Bxc5 28.b4 Bd6 29.a4 a5 30.axb5 axb4 31.Ra6 bxc3 32.Nxc3 Bb4 33.b6 Bxc3 34.Bxc3 h6 35.b7 Re3 36.Bxf7+

1-0

Jose Raul Capablanca vs Frank Marshall

Esta partida es una de las partidas más famosas de la historia. Es sencillamente brillante la defensa de Capablanca, el incesante ataque de Marshall sacrificando piezas forzaba jugadas muy complicadas difíciles de ver incluso para algunos ordenadores, a pesar de esto Capablanca se mantuvo firme y fue capaz de encontrar las mejores jugadas en los momentos decisivos. Terminó con un maravilloso jaque con la intención de dar jaque mate en 5 jugadas.

La razón de anotar todas estas apariciones de grandes jugadores y de mostrar gráfica o algebraicamente sus posiciones y jugadas, no es otra que la de poner de manifiesto, cómo el pensamiento humano evolucionaba exponencialmente con el constante análisis y contraanálisis de cálculo de posiciones, en una manera similar a lo que realiza el procesador de un Ordenador. De ahí que la aparición de la Informática supusiera todo un hito en la evolución del pensamiento del ajedrez.

6. APARICIÓN DE LA INFORMÁTICA EN EL JUEGO

6.1 Turochamp

Probablemente el motor de ajedrez más famoso sea el DeepBlue de IBM's, pero pocos conocen “Turochamp” posiblemente el primer prototipo de ordenador calculando el juego. Fue desarrollado por Alan Turing, para muchos historiadores conocido como el “padre de la informática moderna”. Este algoritmo fue escrito a finales de 1940 y se ejecutó en una máquina por primera vez a principios de 1950 en Inglaterra.

Se diseñó para ejecutarse en la computadora Ferranti Mark I, utilizaba una búsqueda de árbol para calcular la mejor jugada. Una búsqueda de árbol es un algoritmo utilizado para representar los estados posibles de un problema. Si estuviéramos hablando de ajedrez sería una forma de representar las posibles posiciones para cada jugada. Aunque “Turochamp” era muy básico, fue el que sentó las bases computacionales del juego.

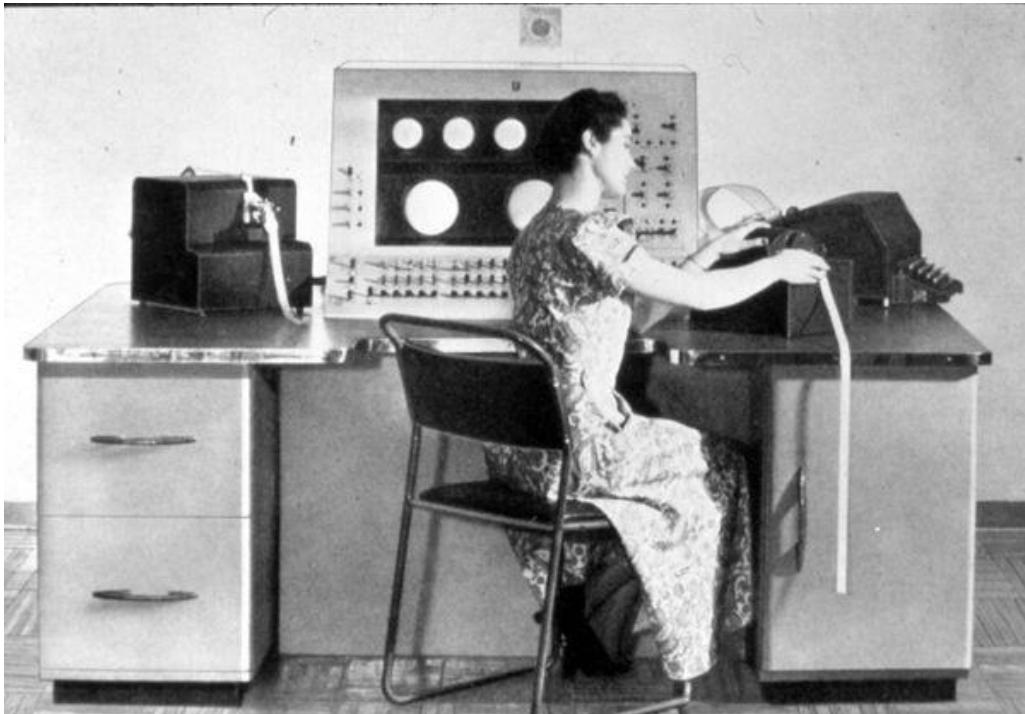


Figure 37. Ferranti Mark I Chess [Photograph]

No solo estaba Turing en el proyecto, hubo un destacado personaje que ayudó en la construcción del algoritmo, el economista y matemático inglés David Gaewen Champernowne. Juntos desarrollaron una de las maneras más acertadas de evaluar una posición a nivel computacional. La función heurística que propusieron es la siguiente:

1. Valor de las piezas: Darle un valor para cada pieza que se encuentre en el tablero, Turin y Champernowne proponían 1 para los peones, 3 para los caballos, 3.5 para los alfiles, 5 las torres y 10 para la reina.
2. Evaluar la movilidad: Para la reina, las torres, los caballos y los alfiles contamos como valor de movilidad la raíz cuadrada del número de posibles movimientos que tiene cada pieza, si tenemos una captura la contamos como 2 en vez de 1.
3. Seguridad de las piezas: Para la torre, el alfil y el caballo añadimos 1 punto si está defendido y 1.5 si está defendido por lo menos dos veces.
 - 1.
4. Mobilidad del rey: Hacemos lo mismo que en la movilidad de las piezas.
5. Seguridad del rey: Para el rey asumimos como si una reina estuviera en nuestro lugar y calculamos su movilidad, después substraemos ese resultado.
6. Enroque: Añadimos 1 punto si es posible hacer un enroque y añadimos otro si fuera posible realizarlo en el siguiente movimiento.
7. Valor para los peones: En este caso añadimos 0.2 puntos por cada rango avanzado por cada peón y 0.3 si estuviera defendido.
8. Mates y jaques: Añadimos 1 punto por cada amenaza de mate y jaque.

Un ejemplo de cómo evaluaría Turochamp la posición inicial del juego sería el siguiente:

Para el cálculo de los valores consideramos:

- P = Peón
- R= Torre
- N= Caballo
- B= Alfil
- Q= Reina
- K= Rey

Para las piezas negras consideramos la misma notación, pero con letras minúsculas.

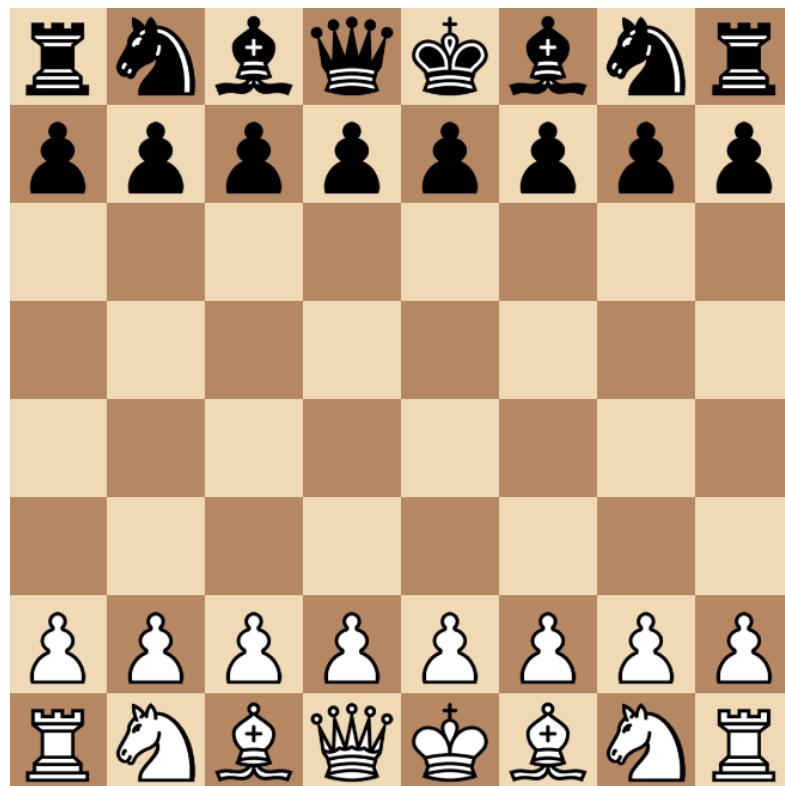


Figure 38: Chess Explorer

Valor de las piezas:

$$\begin{aligned} \text{valorPiezas} &= (8 * P + 2 * N + 2 * B + 2 * R + 2 * Q) - (8 * p + 2 * n + 2 * b + 2 * r + 2 * q) = \\ &= (8 * 1 + 2 * 3 + 2 * 3.5 + 2 * 5 + 2 * 10) - (8 * 1 + 2 * 3 + 2 * 3.5 + 2 * 5 + 2 * 10) = 0 \end{aligned}$$

Movilidad:

En este caso pueden hacer movimientos los caballos nada más de las piezas mayores, luego existen 4 posibles movimientos para N, B, R y Q. Como no hay capturas no hay movimientos que puntúen doble.

$$\text{movilidad} = \sqrt{\text{numMovimientos}} = \sqrt{4} = 2$$

Seguridad de las piezas:

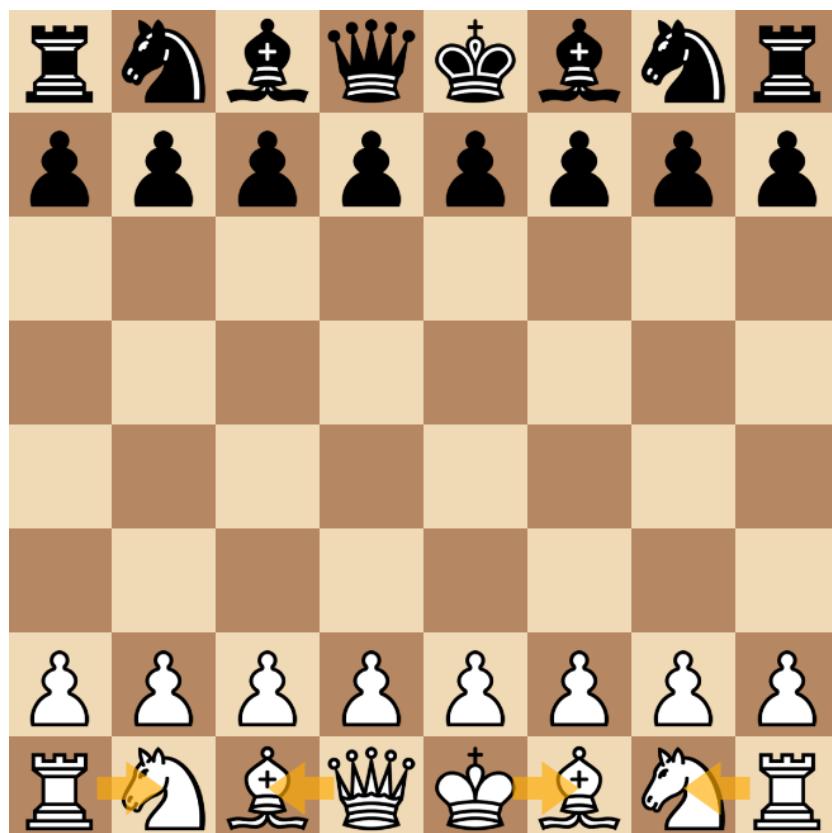


Figure 39: Chess Explorer

Observamos que existen 4 posibles defensas para R, B, N, luego el resultado de será el número de posibles defensas.

$$\text{Seguridad} = 4$$

Movilidad del rey:

Hacemos lo mismo que para la movilidad y como no existen posibles movimientos para el rey el resultado es 0.

`movilidadRey = 0`

Seguridad del rey:

La forma de calcular la seguridad del rey es bastante curiosa, ya que colocamos una reina en el lugar del rey, calculamos su movilidad y substraemos el resultado al valor final, como si quisiéramos calcular lo descubierto que está nuestro rey.

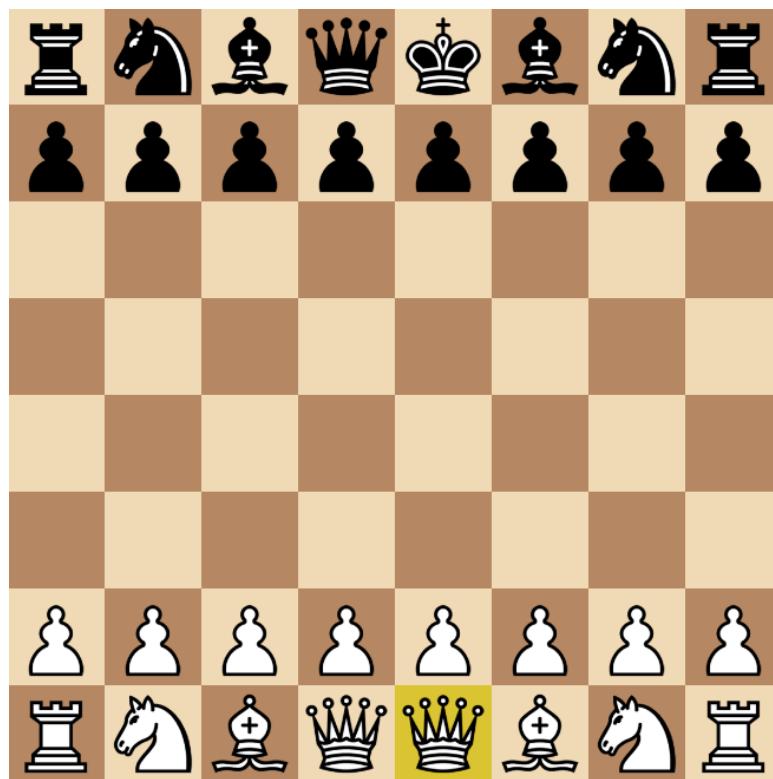


Figure 40: Chess Explorer

En este caso la reina en la casilla e4 no tiene posibles movimientos luego el resultado es 0.

`seguridadRey = -(0)`

Enroque:

En este caso podemos todavía enrocar hacia ambas direcciones, ya que no hemos movido ninguna pieza. Pero no podemos hacer ningún enroque en el siguiente movimiento.

$$\begin{aligned} \text{Enroque} &= \text{Posibilidad_Enroque_Largo} + \text{Posibilidad_Enroque_corto} + \text{Posibles_Enroques} \\ &= 1 + 1 + 0 = 2 \end{aligned}$$

Valor de los peones:

En este caso sumamos los valores de cada peón defendido ya que ninguno ha avanzado una fila.

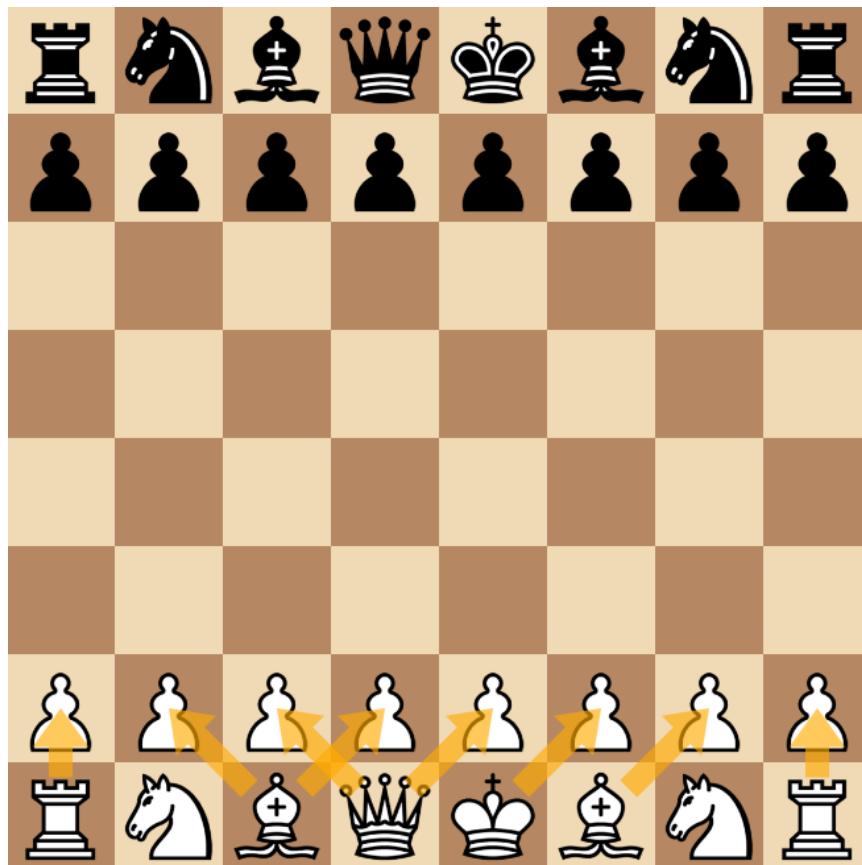


Figure 41: Chess Explorer

$$\text{Peones} = 0.3 * 8 = 2.4$$

Amenazas de jaque y mate:

En el siguiente turno no existen amenazas de jaque y de jaque mate luego el resultado es 0.

Jaques = 0

Resultado:

valorPiezas + Movilidad + Seguridad + movilidadRey + seguridadRey + Enroques + Peones +

$$\text{Jaques} = 0 + 2 + 4 + 0 - 0 + 2 + 2.4 = 10.4$$

Según Turochamp 10.4 es el resultado de evaluar la posición inicial de una partida. Solo quedaría introducir una búsqueda de árbol para poder crear un algoritmo capaz de jugar al juego.

Es impresionante la comprensión que tenía Turing para su época sobre la algorítmica y algoritmos como este lo demuestran, claramente un adelantado a su época en cuanto a computación.

6.2 Deep Blue IBM

Sin duda el ordenador que más impactó a los seguidores del juego era el superordenador desarrollado por el equipo de IBM, llamado “Deep Blue”. El nombre fue elegido como una referencia a “Deep Blue Sea” que se refiere a un “mar profundo azul” como si se tratase de algo imponente, pero el color azul solo hacía referencia a la franquicia.

Originalmente el nombre del proyecto era “Chips”, pero el invertir recursos en el superordenador hizo que el equipo renombrara al mismo. Así fue como se eligió el nombre de “Deep Blue”, en su momento el ordenador más fuerte con absoluta diferencia sobre el resto.

El primer prototipo se construyó en 1989, pero éste no poseía la destreza para hacerle competencia a un maestro de ajedrez, aunque si a un jugador casual del juego. No fue hasta 1996 cuando se presentó una versión del ordenador que se enfrentó al campeón mundial de aquel entonces, Gary Kasparov, una auténtica bestia del juego, y considerado para muchos como uno de los tres mejores de la historia. Tenía un estilo agresivo y preciso, haciéndolo uno de los jugadores más temidos. Destacamos los duelos que tenía con Anatoly Karpov, otro jugador con una base ajedrecística excepcional, tenía un estilo más pausado y posicional. Aún con el eclipse que le hacía Kasparov, Karpov es considerado para muchos como uno de los mejores, incluso comparándolo con jugadores actuales, y esto hacía que fuera una auténtica explosión cada vez que jugaban ambos.



Figure 42: Kasparov y Karpov en Reggio Emilia 1991-92

Kasparov fue el primero en enfrentarse a un superordenador de ajedrez. En el primer enfrentamiento Gary fue capaz de ganar con un marcador de 4 – 2 partidas que jugaron y una de ellas fue victoria para “Deep Blue” lo que supuso la primera victoria de un ordenador sobre un gran maestro de ajedrez.

Hay que hacer notar que muchísimos jugadores del mundo no serían capaces de ganarle a este prodigo una de seis partidas y estaríamos hablando de jugadores muy fuertes, incluyendo a maestros entre ellos. Esto supuso uno de los mayores avances en el desarrollo del ajedrez. Al principio, Kasparov pensó que este ordenador solo era un “cajón de arena gigante”, pero rápidamente se dio cuenta de que era un excelente rival, aunque este parecía a veces frustrado por jugadas que no lograba comprender por parte de la máquina.

Las partidas se jugaron con un límite de tiempo de 40 movimientos en dos horas, seguido de 20 movimientos en una hora. En la primera partida, DeepBlue consiguió sorprender a todo el mundo tras coger desprevenido al campeón y conseguir la victoria.

La segunda partida continuó con un empate tras la decisión de jugar de manera más conservadora por parte del campeón. La tercera partida siguió con un empate. En la cuarta partida, sorprendió de nuevo Deep Blue logrando la ventaja. Pero la última partida fue victoria de Kasparov y tras esto decidió jugar de manera más conservadora, pensando a largo plazo y llevarse el empate.

Un año más tarde, en 1997, IBM lanzó una nueva versión, esta vez mucho más dispuesta a vencer al campeón. De nuevo se enfrentarían en un encuentro a seis partidas: Esta vez, “Deep Blue” venció en el enfrentamiento, convirtiéndose en la primera máquina capaz de ganar a un campeón de ajedrez.

El éxito de Deep Blue fue un hito en la historia de la inteligencia artificial y demostró que las máquinas podían superar a los humanos en juegos de estrategia complejos.

Muchos aficionados al ajedrez se reunieron en el centro de convenciones de Filadelfia para ver el juego en vivo, mientras que millones de personas en todo el mundo siguieron la partida por los medios de comunicación.

La segunda partida demostró la capacidad del ajedrez y la tecnología para atraer y entretener a un público masivo. También puso de manifiesto el potencial de la tecnología para involucrar a

la gente en eventos deportivos y de competición a través de las transmisiones en línea.



Figure 43: Garry Kasparov, former world chess champion, plays against the machine

Esta victoria de Deep Blue abrió nuevas puertas en el campo de la inteligencia artificial y la computación, remarcaba la potencia que podía llegar a alcanzar un ordenador para resolver problemas complejos. A raíz de este hito se han desarrollado nuevas técnicas y métodos para resolver problemas complejos.

El funcionamiento de este ordenador se basaba en una combinación de técnicas algorítmicas. Entre estas encontramos el uso de una base de datos con partidas previamente estudiadas por un equipo, de esta forma conseguían que la máquina encontrara patrones de manera casi automática.

Al igual que Turochamp utilizaba un algoritmo de búsqueda de árbol, en este caso utilizaban “Negamax Scout”. Este algoritmo es una variante del algoritmo de “MiniMax”, ya que si nos ponemos a pensar el juego del ajedrez no es más que un problema de “máximos y mínimos”,

podríamos imaginar el ajedrez como un juego en el que tratamos de maximizar las victorias por parte de las piezas blancas mientras las piezas negras tratan de minimizar ese resultado.

El superordenador también poseía de funciones heurísticas que le permitían evaluar una posición de manera precisa, un ejemplo de función heurística para calcular el valor de una posición es la que hemos visto anteriormente con Turochamp.

No solo, hacía todo eso, sino que también poseía información con tablas de finales y utilizaba técnicas de optimización como la paralelización de cálculos y la utilización de hardware especializado para cálculos de punto flotante.

6.3 Alphazero

Quizás AlphaZero es el algoritmo de ajedrez del que más hemos oído hablar en los últimos años, no solo por su capacidad en este juego, también se probó en juegos de mesa como el shogi y el Go, ambos son juegos muy complejos y con un amplio árbol de búsqueda.

Lo que hace único a AlphaZero es su capacidad de aprender juegos de mesa sin conocimientos previos sobre las estrategias humanas. Todo esto es posible gracias a una de las técnicas más influyentes en la actualidad, el “aprendizaje por refuerzo”, donde el motor juega partidas contra sí mismo y va ajustando su estrategia en función de su evaluación.

Una traza de lo que hace AlphaZero sería la siguiente:

1. Inicializamos una red neuronal para evaluar las posiciones.
2. Hasta condición de parada:
 - a. Juega juegos completos para tomar decisiones en cada movimiento del juego utilizando la red anterior. Guarda el estado de cada movimiento y actualiza la política de probabilidad de cada estado.
 - b. Despues de un número de partidas, Alphazero se evalúa para determinar su fuerza en el juego.

- c. Los pesos de la red neuronal se actualizan utilizando un algoritmo de descenso de gradiente estocástico.

En resumen, AlphaZero utiliza las redes neuronales como base para el aprendizaje reforzado y consigue evolucionar jugando muchas partidas y es capaz de jugar a juegos de mesa de manera eficaz, haciendo que ni los mejores jugadores de ajedrez se acerquen a su nivel.

AlphaZero consiguió vencer al mejor motor de ajedrez de 2017 en una competición a 100 partidas, venció a Stockfish en su versión 8. La victoria fue favorable para AlphaZero con un marcador aplastante de 28 victorias, 72 empates y ninguna derrota. A pesar de esto, la competencia de los motores de ajedrez se la lleva Stockfish desde la implementación de su versión número 13 en la que incorporaron también el aprendizaje reforzado. Lo realmente importante de esta hazaña fue la capacidad no solo de competir con el mejor motor de ajedrez del momento si no la victoria absoluta sobre el mismo.

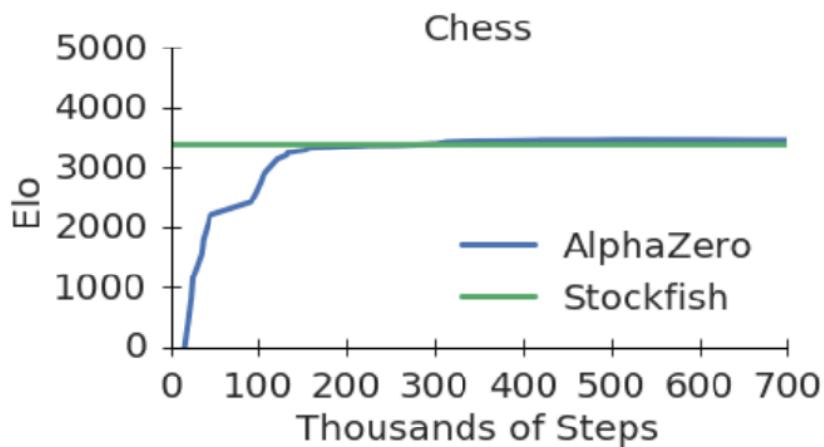


Figure 44: Training AlphaZero by self-play

Stockfish 8 utilizaba una búsqueda de árbol con una ponderación de Alpha y Beta que explicaré más adelante y un proceso de poda selectiva. También incorporaba una heurística compleja que utilizaba una base de datos y funciones complejas para evaluar las posiciones.

7. ESTUDIO DE LOS ALGORITMOS

7.1 Minimax

La primera implementación que me gustaría explorar es el algoritmo de Minimax, ya que cuando consideramos una computadora analizando una posición en un juego, ésta calcula todas las posibilidades en el tablero, aunque no sea el caso para todos los motores. Con este algoritmo, pretendemos examinar todo el árbol de búsqueda hasta una profundidad específica. Por ejemplo, si tenemos una posición de ajedrez en la que las piezas blancas se mueven primero, exploraríamos todas las posibles jugadas para las piezas blancas y luego analizaríamos todas las posibles respuestas de las piezas negras a esas jugadas, y así sucesivamente hasta alcanzar la profundidad deseada.

El algoritmo funciona de manera recursiva, de esta forma es posible evaluar todas las posibilidades. La base del “Minimax” es maximizar el valor para las piezas blancas y minimizar el valor para las piezas negras, ya que si no minimizáramos las piezas negras no tendríamos en cuenta posibles buenas jugadas de estas. Por lo tanto, asumimos que el rival nos va a contestar con el mejor movimiento siempre.

Para implementar este algoritmo debemos imaginarlo el árbol de búsqueda que haríamos en una posición de ajedrez, un ejemplo sería el siguiente:

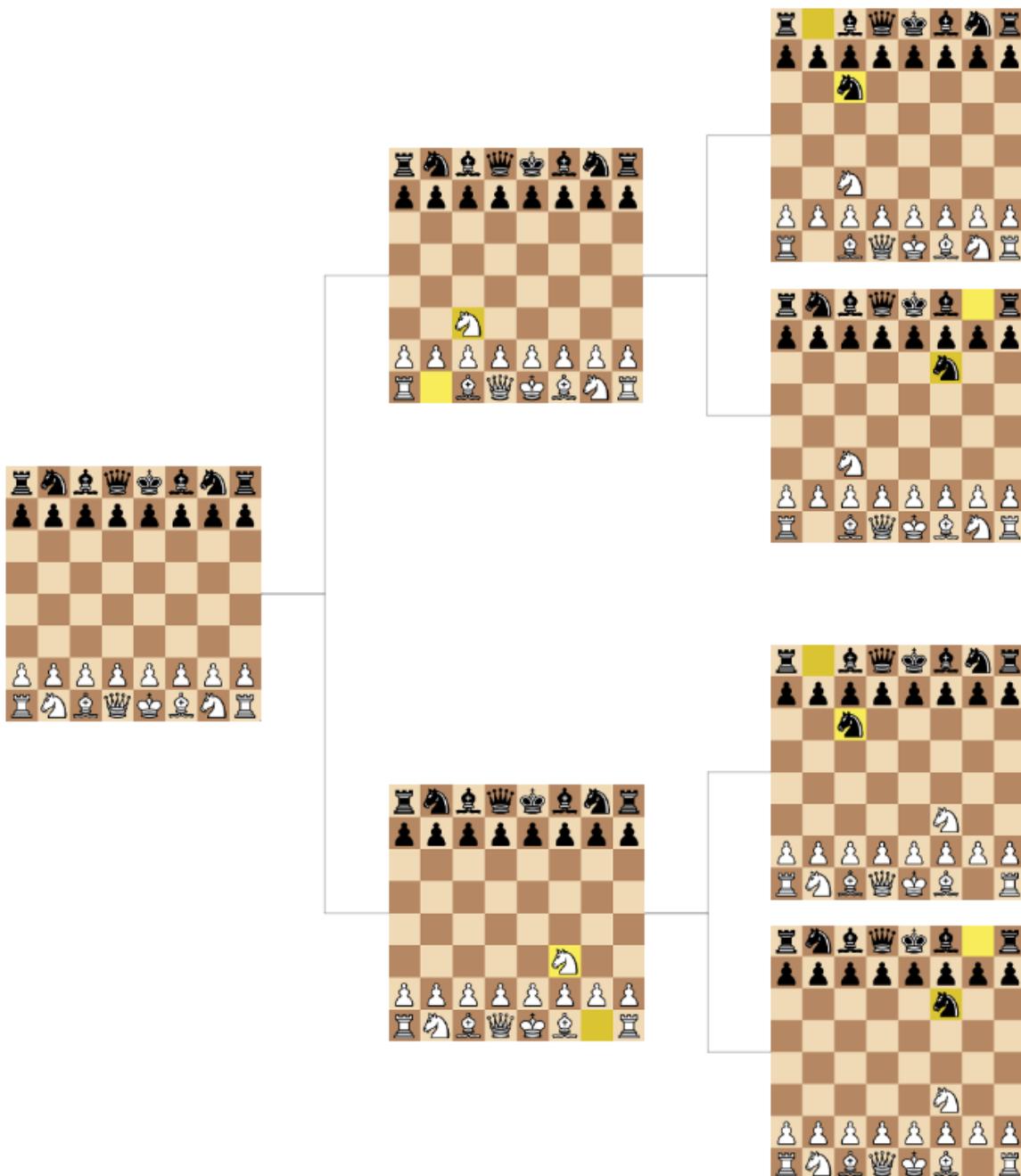


Figure 45: Menacho, E.

En este ejemplo, hacemos una traza de profundidad 2 de algunas de las posibles jugadas en la primera posición, si estuviéramos usando el algoritmo de máximos y mínimos utilizaríamos

una heurística para evaluar cada posible estado. Podríamos considerar los siguientes valores para esta traza:

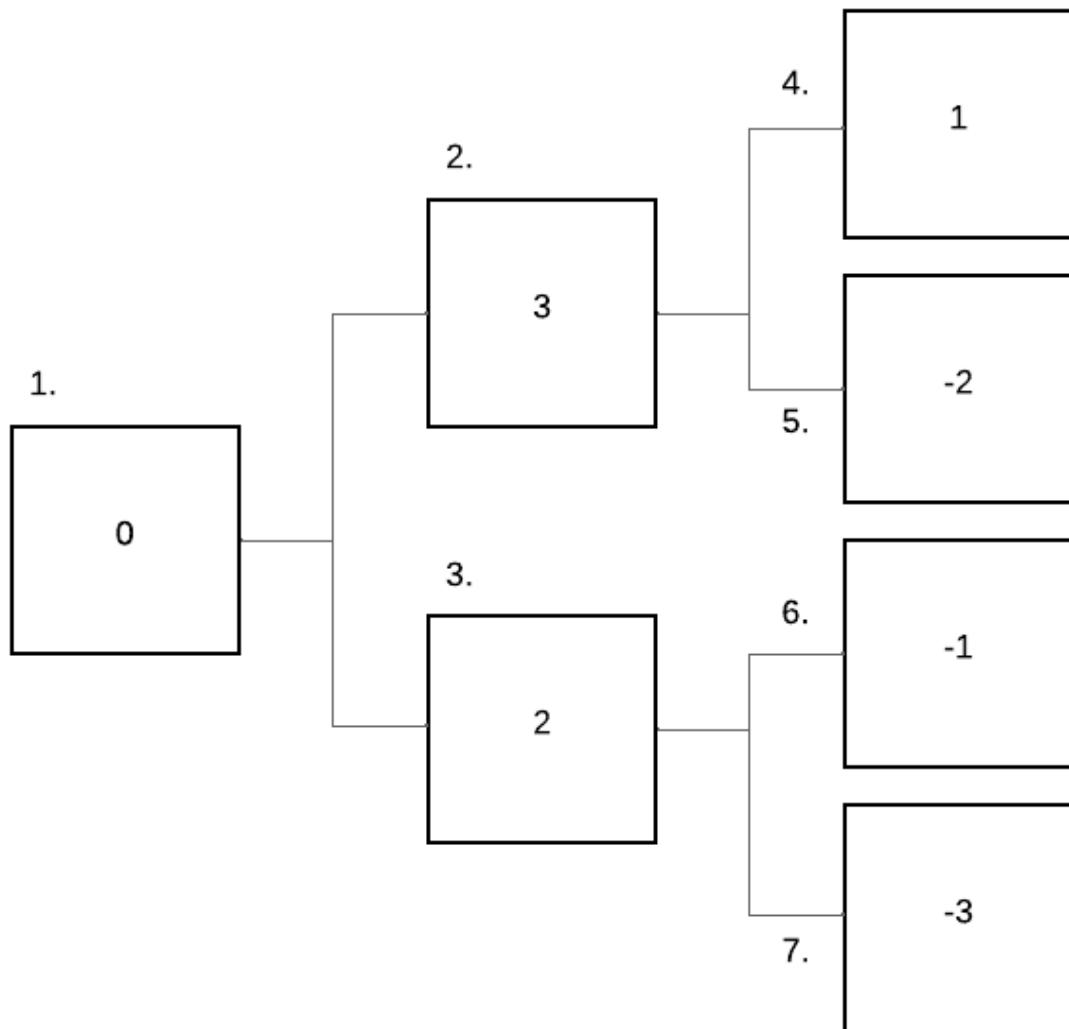


Figure 46: Menacho, E.

A continuación, debemos ir calculando desde el final del árbol, ya que evaluamos los movimientos de manera recursiva. Por ejemplo, si empezamos por la parte del final, evaluamos un movimiento por parte de las negras luego nos quedamos con el menor de los valores.

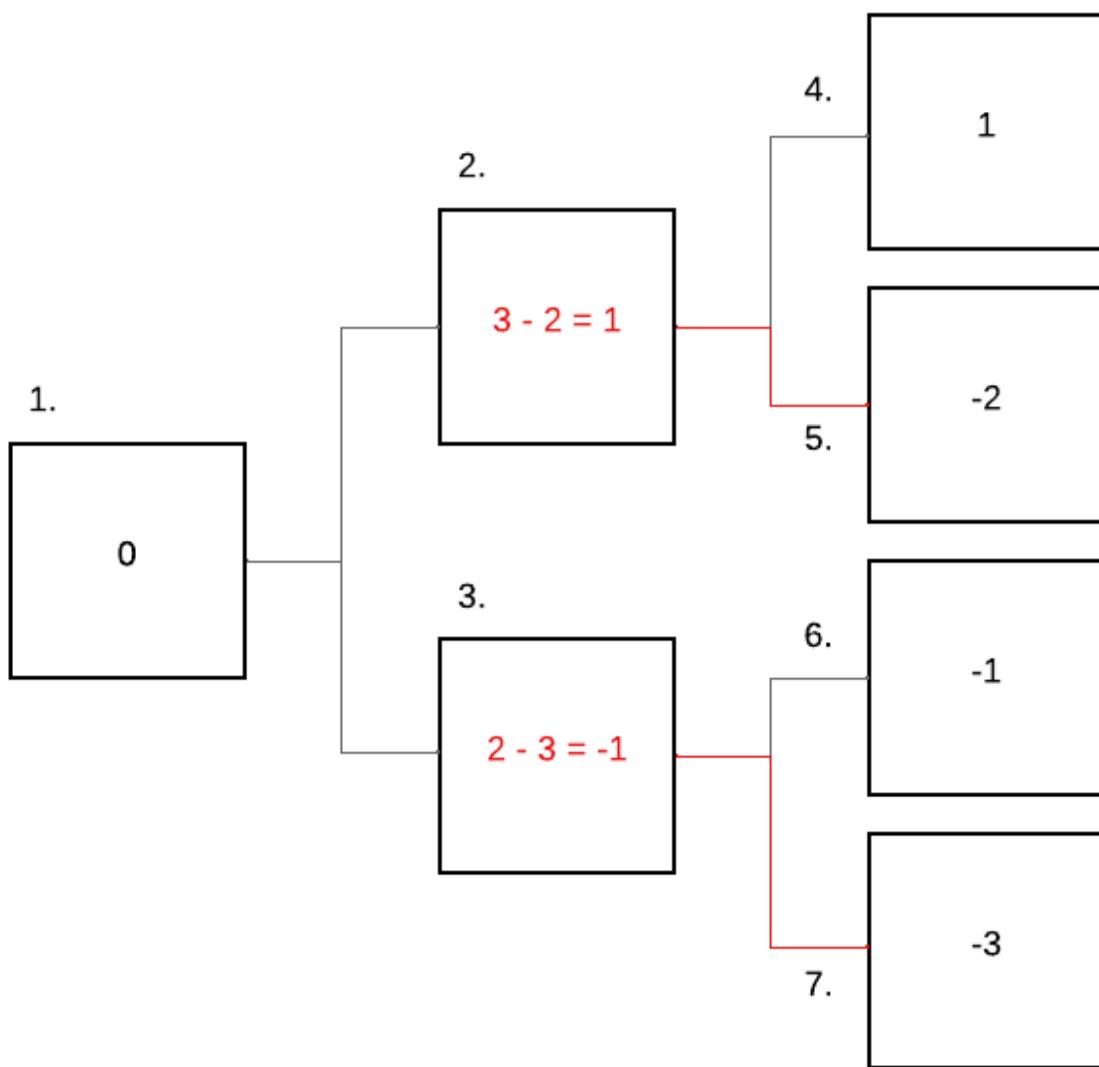


Figure 47: Menacho, E.

Por último, al mover las blancas tratamos de maximizar el movimiento de las mismas luego seleccionaremos el mayor de los valores posibles:

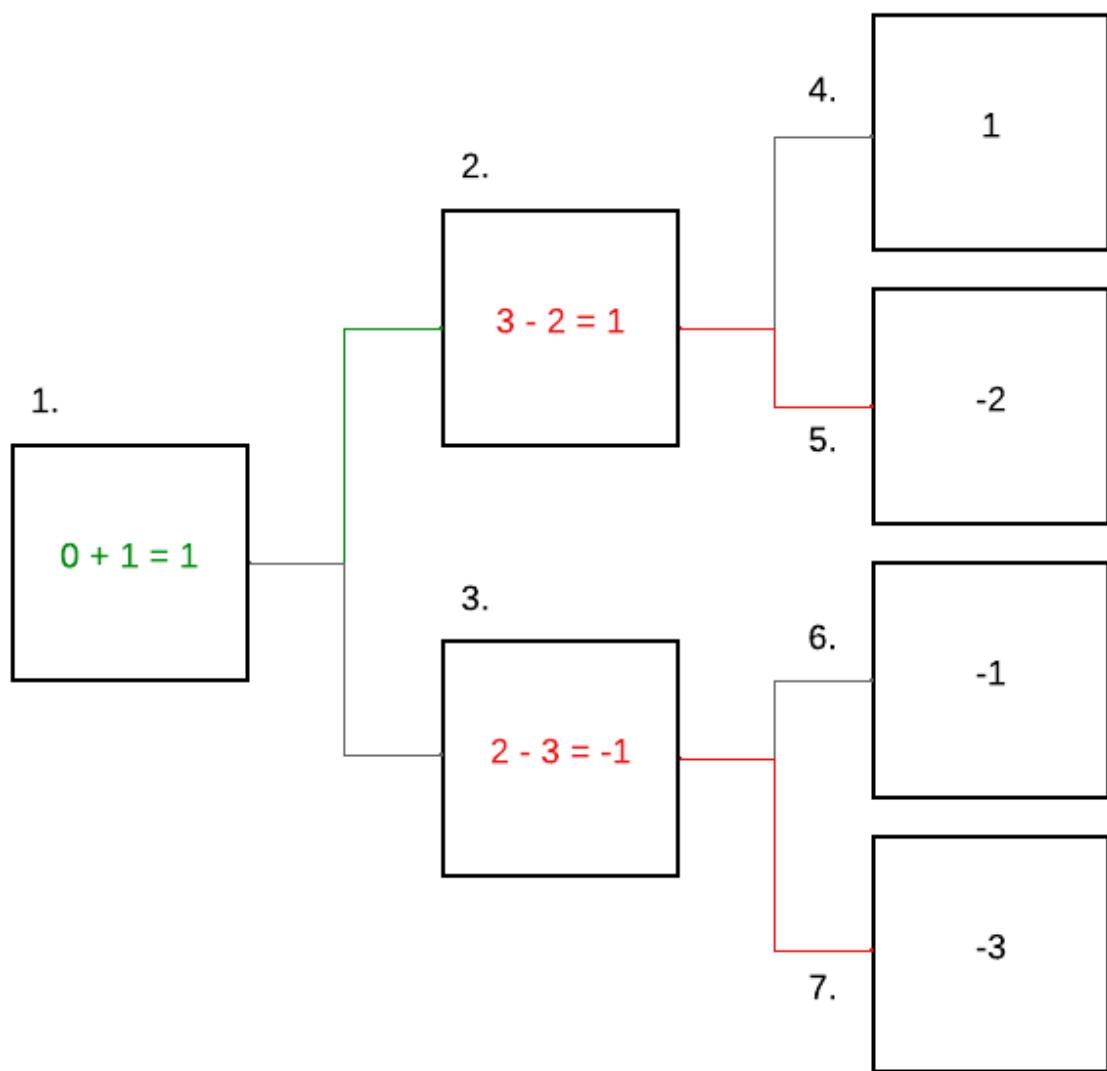


Figure 48: Menacho, E.

Luego la secuencia que considera como mejor el algoritmo sería: $1 \rightarrow 2 \rightarrow 5$.

Con un valor de 1.

A continuación, adjunto el pseudocódigo del algoritmo:

```
def minimax(nodo, profundidad):
```

Si la profundidad = 0 o el nodo es un nodo terminal:

devuelve el valor heurístico del nodo

Si juegan Blancas:

mejorValor = -infinito

para cada nodo hijo de nodo:

valor = minimax(hijo, profundidad - 1)

mejorValor = max(mejorValor, valor)

return mejorValor

Sino:

mejorValor = +infinito

para cada nodo hijo de nodo:

valor = minimax(hijo, profundidad - 1)

mejorValor = min(mejorValor, valor)

return mejorValor

Ahora que ya sabemos la teoría detrás del algoritmo solo nos queda implementarlo y comprobar su fuerza.

Para la primera implementación del algoritmo utilizaremos Python como lenguaje de programación y nos apoyaremos en una librería denominada “chess”. Para ello nos apoyaremos

en una herramienta llamada “Colaboratory”, ya que nos permite ejecutar un entorno de programación en un servidor y de esta forma cualquier persona podría ejecutar el cuaderno únicamente utilizando internet y una cuenta de Google.

El módulo de “chess” es super útil a la hora de hacer operaciones con un tablero de ajedrez, ya que posee una gran cantidad de métodos y funciones que nos facilitan la tarea bastante y nos permite evaluar los algoritmos antes de implementarlos en nuestro sitio web.

Para ello, primero creamos el cuaderno, accedemos a Google Drive e instalamos Colaboratory, para ello hacemos click derecho en una carpeta vacía de nuestro Google Drive y seleccionamos
→ más → Google Colaboratory

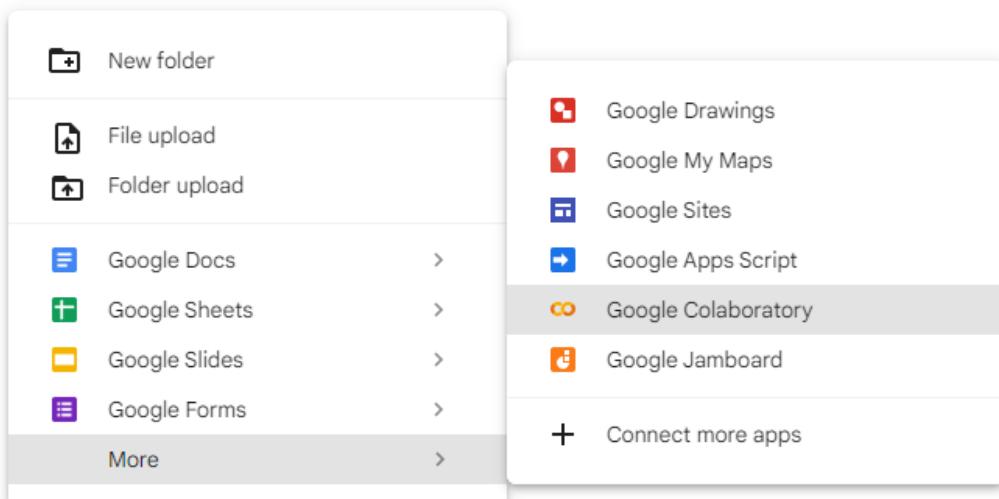
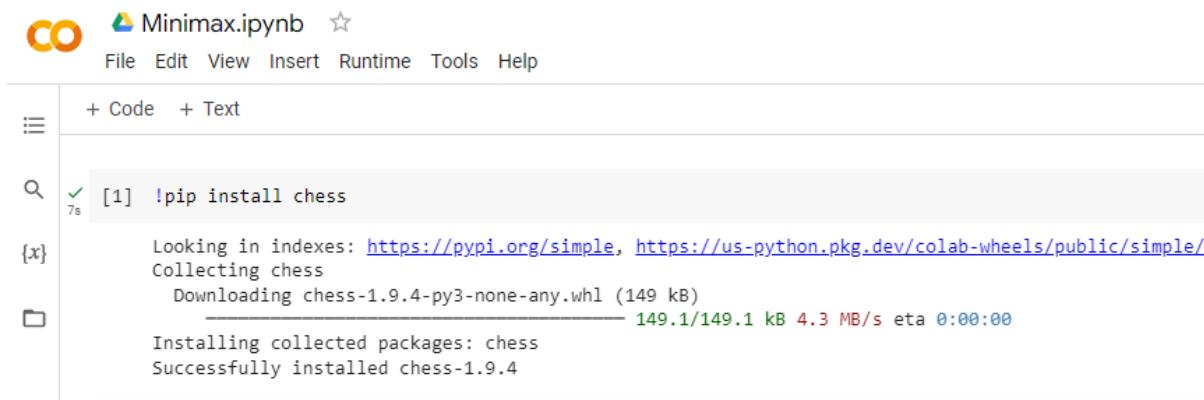


Figure 49: Menacho, E.

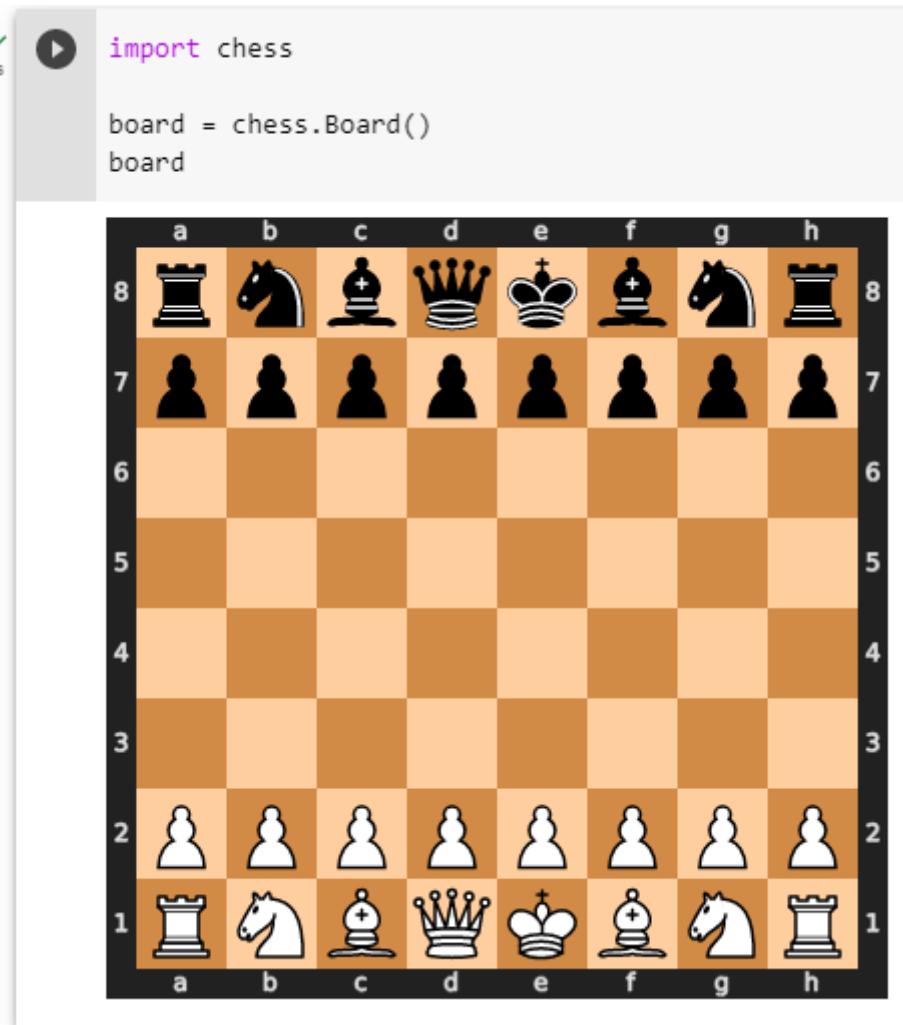
Una vez instalado elegimos el nombre de nuestro proyecto e instalamos la librería de chess para comprobar que todo estuviera en orden.



```
[1] !pip install chess
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting chess
  Downloading chess-1.9.4-py3-none-any.whl (149 kB)
149.1/149.1 kB 4.3 MB/s eta 0:00:00
Installing collected packages: chess
Successfully installed chess-1.9.4
```

Figure 50: Menacho, E.

Lo siguiente será crear nuestro primer tablero de ajedrez e imprimirlo por pantalla.



```
import chess
board = chess.Board()
board
```

The image shows a standard chessboard setup with black pieces on the top row (a8-h8) and white pieces on the bottom row (a1-h1). The pieces are represented by their standard icons: King, Queen, Rook, Bishop, Knight, and Pawn. The board is an 8x8 grid with squares alternating in color between light beige and dark brown.

Figure 51: Menacho, E.

La primera función que debemos implementar es una heurística que nos permita evaluar una posición, para ello consideraremos las piezas con la siguiente notación: Peón (P), Caballo (N), Alfil (B), Torre (R), Reina (Q) y Rey (K). Para ellos consideramos los siguientes valores: P = 1, N= 3, B= 3, R= 5, Q= 10. Por lo tanto, para evaluar la posición utilizando este módulo haríamos lo siguiente:

```
def get_pieces_value(board):
    white = board.occupied_co[chess.WHITE]
    black = board.occupied_co[chess.BLACK]
    return (
        chess.popcount(white & board.pawns) -
        chess.popcount(black & board.pawns) +
        3 * (chess.popcount(white & board.knights) -
              chess.popcount(black & board.knights)) +
        3.5 * (chess.popcount(white & board.bishops) -
                chess.popcount(black & board.bishops)) +
        5 * (chess.popcount(white & board.rooks) -
              chess.popcount(black & board.rooks)) +
        9 * (chess.popcount(white & board.queens) -
              chess.popcount(black & board.queens))
    )

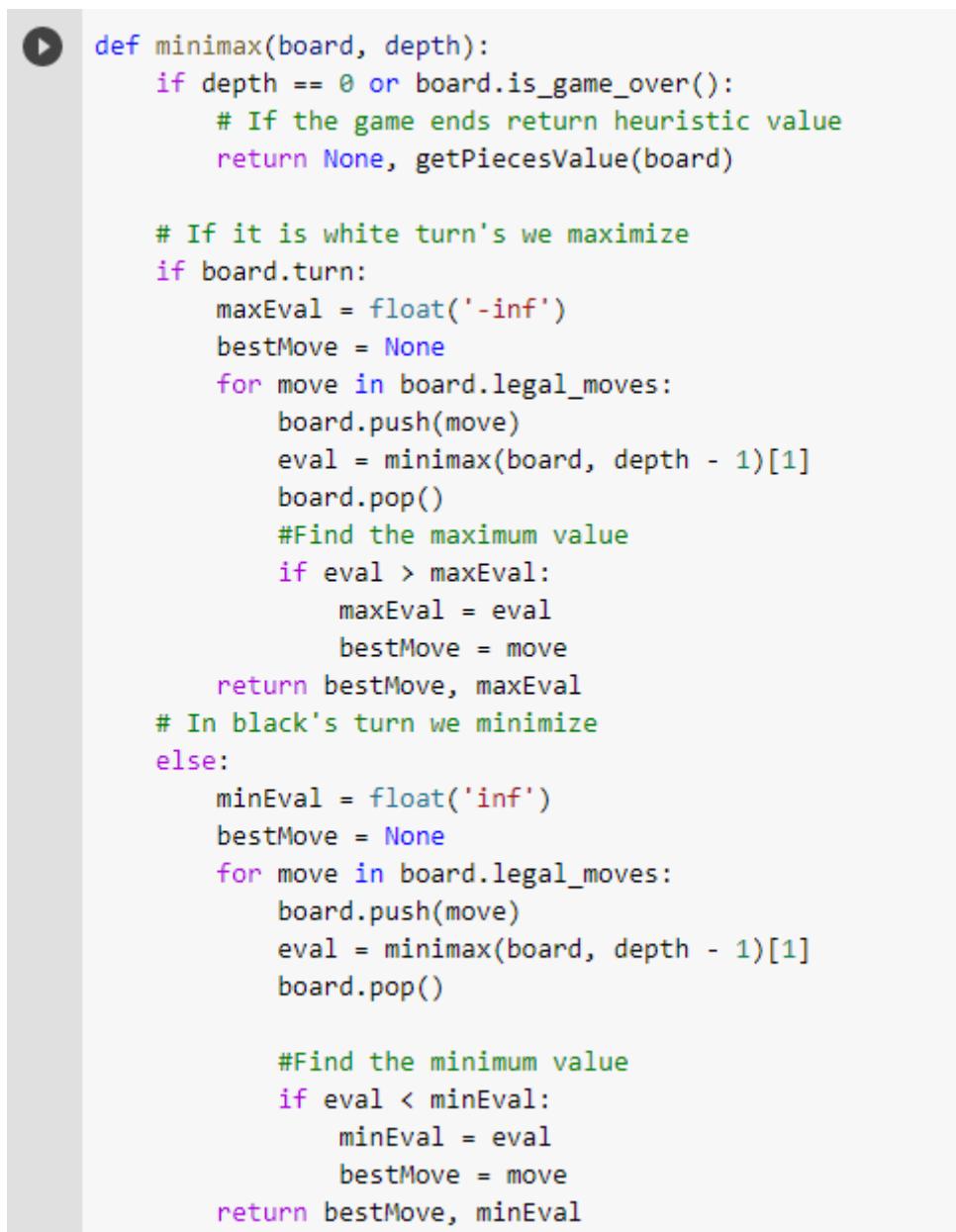
get_pieces_value(board)
```

0.0

Figure 52: Menacho, E.

Con esta función calculamos el número de piezas para las blancas y lo multiplicamos por su valor correspondiente. Después substraemos el valor de las piezas negras por su valor correspondiente de pieza.

Para el algoritmo de minimax creamos una función y usamos como heurística el valor del material de las piezas.



```
def minimax(board, depth):
    if depth == 0 or board.is_game_over():
        # If the game ends return heuristic value
        return None, get_pieces_value(board)

    # If it is white turn's we maximize
    if board.turn:
        maxEval = float('-inf')
        bestMove = None
        for move in board.legal_moves:
            board.push(move)
            eval = minimax(board, depth - 1)[1]
            board.pop()
            #Find the maximum value
            if eval > maxEval:
                maxEval = eval
                bestMove = move
        return bestMove, maxEval
    # In black's turn we minimize
    else:
        minEval = float('inf')
        bestMove = None
        for move in board.legal_moves:
            board.push(move)
            eval = minimax(board, depth - 1)[1]
            board.pop()

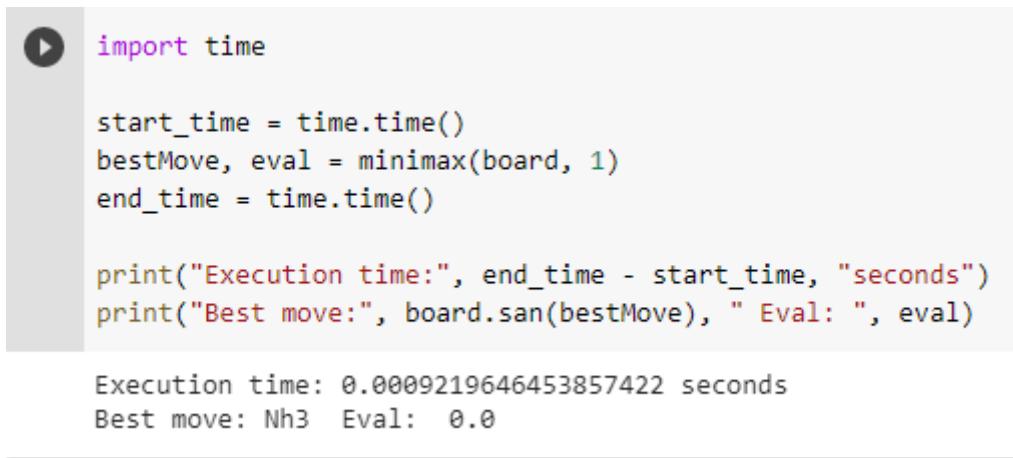
            #Find the minimum value
            if eval < minEval:
                minEval = eval
                bestMove = move
        return bestMove, minEval
```

Figure 53: Menacho, E.

En este caso devolvemos el mejor de los valores obtenidos tras el análisis completo de la posición. Maximizando en caso de que jueguen las blancas y minimizando en el caso contrario.

Ejecutamos el algoritmo en varias profundidades para comprobar que todo estuviera en orden y capturamos el tiempo de ejecución del mismo:

Profundidad 1:



```
import time

start_time = time.time()
bestMove, eval = minimax(board, 1)
end_time = time.time()

print("Execution time:", end_time - start_time, "seconds")
print("Best move:", board.san(bestMove), " Eval: ", eval)
```

Execution time: 0.0009219646453857422 seconds
Best move: Nh3 Eval: 0.0

Figure 54: Menacho, E.

Profundidad 2:

```
Execution time: 0.01525259017944336 seconds
Best move: Nh3 Eval: 0.0
```

Figure 55: Menacho, E.

Profundidad 3:

```
Execution time: 0.15116095542907715 seconds
Best move: Nh3 Eval: 0.0
```

Figure 56: Menacho, E.

Profundidad 4:

```
Execution time: 3.1331281661987305 seconds
Best move: Nh3  Eval:  0.0
```

Figure 57: Menacho, E.

Profundidad 5:

```
Execution time: 87.25947165489197 seconds
Best move: g3  Eval:  1.0
```

Figure 58: Menacho, E.

A mayor profundidad conseguimos mejores resultados y que la máquina juegue de forma más óptima. Observamos que es un algoritmo muy costoso a nivel computacional haciendo que crezca el tiempo de ejecución de manera exponencial con respecto al incremento de la profundidad en el árbol de búsqueda.

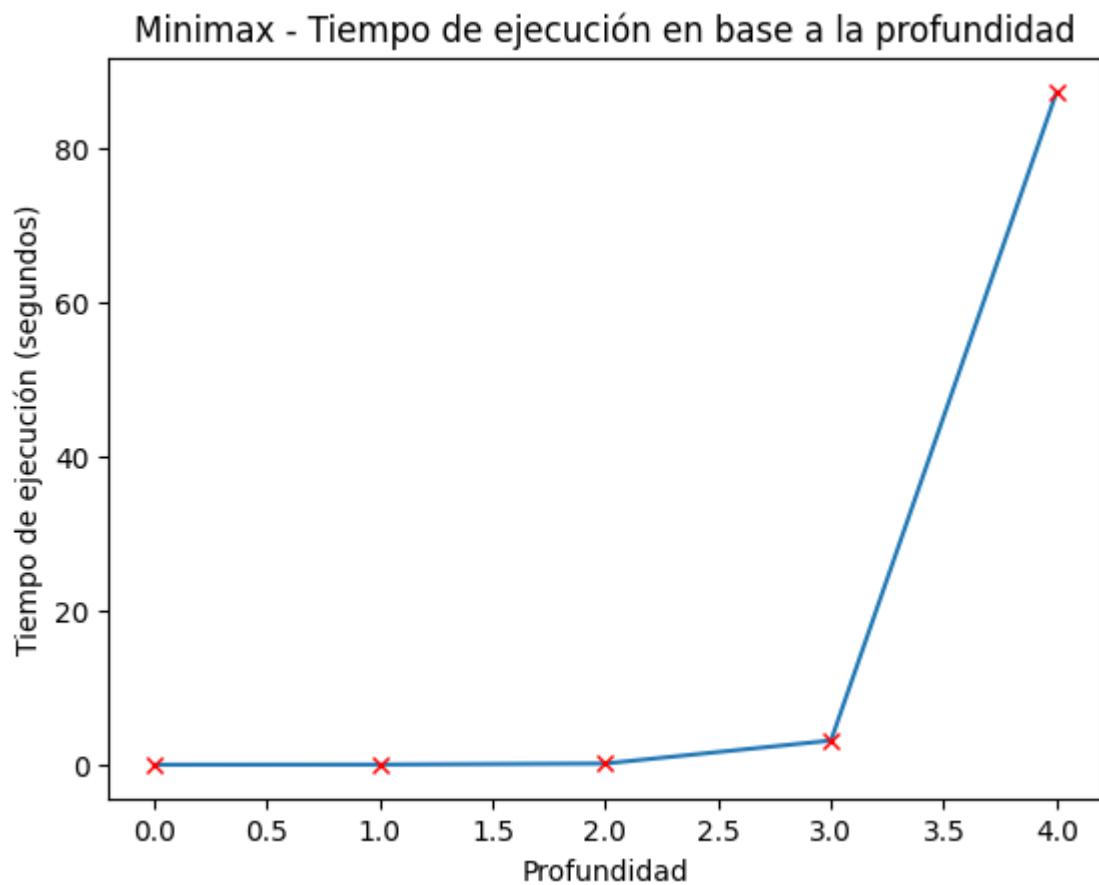


Figure 59: Menacho, E.

Para comprobar que el algoritmo está funcionando correctamente enfrentaremos a él contra sí mismo en este caso en profundidad 3, para ello ejecutaremos la siguiente celda de código proporcionado:

```
moves = []
cont = 1

# While game doesn't ends
while not board.is_game_over() or board.can_claim_draw():

    # White move

    # Get move
    move, eval = minimax(board, 3)
    move = board.san(move)
    # Make move
    board.push_san(move)
    moves.append(move)
    print("-----[" + str(cont) + "]-----")
    print("WHITE----->" + move)

    # Black move
    if board.is_game_over() or board.can_claim_draw():
        break;

    # Get move
    move2, eval = minimax(board, 3)
    # Make move
    move2 = board.san(move2)
    board.push_san(move2)
    moves.append(move2)
    print("BLACK----->" + move2 + "\n")
    print(board)
    print("\n")
    cont += 1

-----[ 1 ]-----
WHITE----->Nh3
BLACK----->Nh6

r n b q k b . r
p p p p p p p p
. . . . . . n
. . . . . . .
. . . . . . .
. . . . . . N
P P P P P P P P
R N B Q K B . R
```

Figure 60: Menacho, E.

Al ejecutar esta celda simularemos una partida del algoritmo contra sí mismo en profundidad

3. Para comprobar quién ha ganado la partida ejecutamos:

```
if board.result() == "1/2-1/2":
    print("-----DRAW-----")
# check if white wins
elif board.result() == "1-0":
    print("-----WHITE WINS-----")
# check if black wins
elif board.result() == "-----0-1-----":
    print("BLACK WINS")

print("\n")
board
```

-----DRAW-----

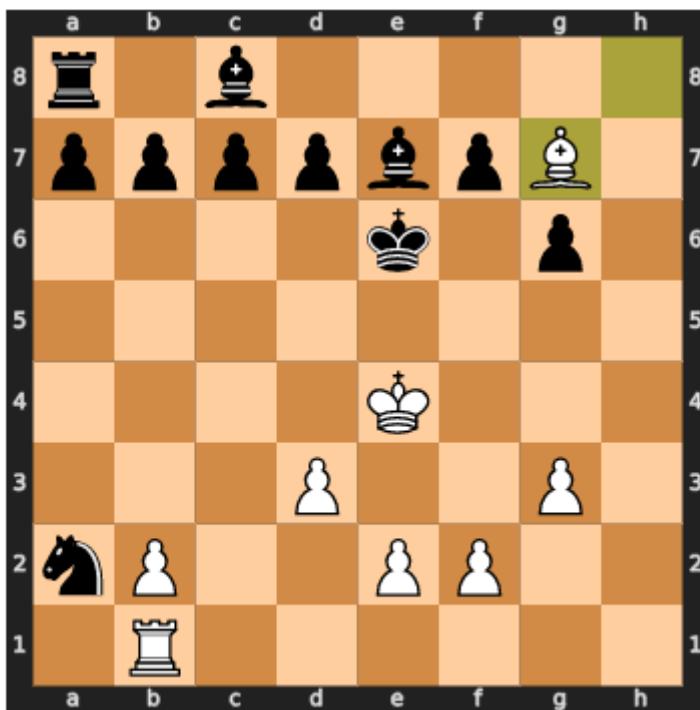


Figure 61: Menacho, E.

Esta partida acabó en empate, ya que no sabe de estructuras y tampoco sabe sobre los posibles jaques y jaques mate, esta partida se desenvolvió de la siguiente forma:

Minimax (3) vs Minimax (3)

1. Nh3 Nh6 2. Ng5 Ng4 3. Nc3 Nc6 4. Nd5 e6 5. Ne3 Qxg5 6. h4 Nxe3 7. hxg5 Nxd1 8. Kxd1 Be7 9. Rh5 Kf8 10. Ke1 Kg8 11. Kd1 Kf8 12. Ke1 Kg8 13. Kd1 Kf8 14. Ke1 Kg8 15. Kd1 Kf8 16. Ke1 Kg8 17. Rb1 Rb8 18. Kd1 Kf8 19. Ke1 Kg8 20. Kd1 Kf8 21. Ke1 Kg8 22. Kd1 Kf8 23. Ke1 Kg8 24. Kd1 Kf8 25. g3 Kg8 26. Bh3 Kf8 27. Bg4 Kg8 28. Bh3 Kf8 29. Bg4 Kg8 30. Bh3 Kf8 31. Bg4 Kg8 32. Bh3 Kf8 33. Bg4 Ke8 34. Bf3 Nd8 35. Bg4 Kf8 36. Bf3 Kg8 37. Bg4 Kf8 38. Bf3 Kg8 39. Bg4 Kf8 40. Bf3 Kg8 41. Bg4 Kf8 42. Bf3 Kg8 43. Be4 Ra8 44. Rxh7 Rxh7 45. Bxh7+ Kxh7 46. d3 Nc6 47. Bf4 Bd8 48. Kd2 Kg8 49. Ke3 e5 50. Bxe5 Bxg5+ 51. Bf4 Bd8 52. Ke4 Nb4 53. Be5 Kf8 54. Bf4 Kg8 55. Be5 Kf8 56. Bf4 Kg8 57. Be5 Kf8 58. Bf4 Kg8 59. Be5 Kf8 60. Bf4 Ke8 61. Be5 g6 62. Bf4 Rb8 63. Be5 Kf8 64. Bf4 Kg8 65. Be5 Kf8 66. Bf4 Kg8 67. Be5 Kf8 68. Bf4 Kg8 69. Be5 Kf8 70. Bf4 Kg8 71. Be5 Ra8 72. Bf4 Nxc2 73. Bh6 Nb4 74. Bf4 Rb8 75. Be5 Kf8 76. Bf4 Kg8 77. Be5 Kf8 78. Bf4 Kg8 79. Be5 Kf8 80. Bf4 Kg8 81. Be5 Kf8 82. Bf4 Ke8 83. Be5 Ke7 84. Bf4 Ke8 85. Be5 Ke7 86. Bf4 Ke8 87. Be5 Ke7 88. Bf4 Ke8 89. Be5 Ke7 90. Bf4 Ke6 91. Be5 Ra8 92. Bf4 Rb8 93. Be5 Ra8 94. Bf4 Rb8 95. Be5 Ra8 96. Bf4 Rb8 97. Be5 Ra8 98. Bf4 Nxa2 99. Bh6 Be7 100. Bg7 Rb8 101. Bh8 Ra8 102. Bg7 Rb8 103. Bh8 Ra8 104. Bg7 Rb8 105. Bh8 Ra8 106. Bg7 Rb8 107. Bh8 Ra8

1/2-1/2

De esta partida destaco dos peculiaridades, en primer lugar, la constante repetición de algunas posiciones y en segundo lugar un momento en el que con las piezas blancas no fue capaz de calcular a mayor profundidad y por eso se deja una pieza mayor. La posición es la siguiente:

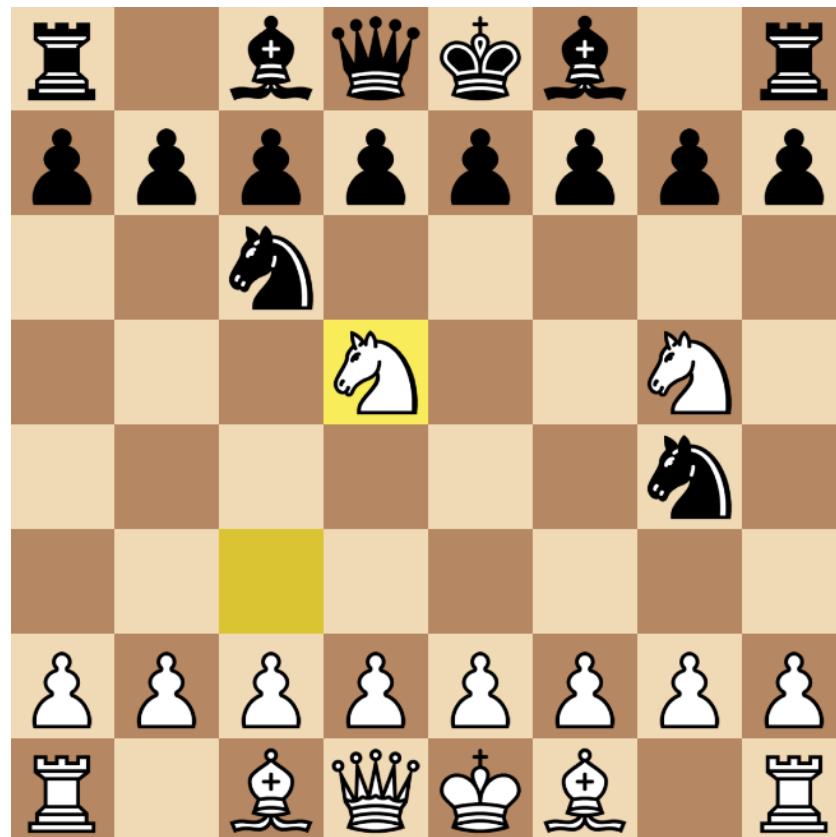


Figure 62: Chess Explorer

Observamos que este movimiento por parte de las blancas no es ni mucho menos el más acertado, ya que viene e6 con una doble amenaza sobre las piezas blancas que es en efecto el movimiento realizado por las piezas negras, tan solo por poder pensar un paso más hacia delante.

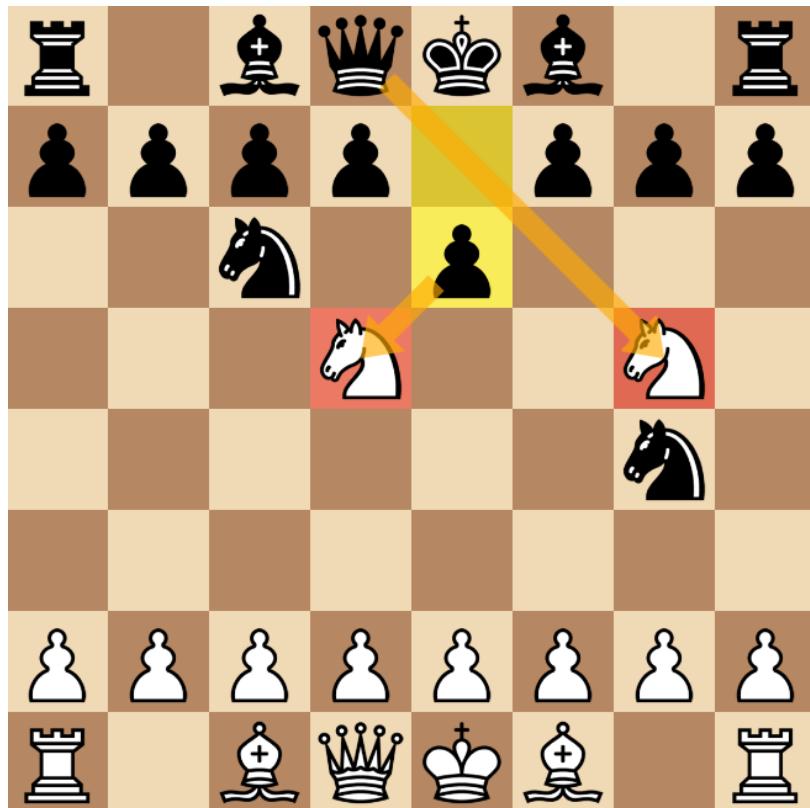


Figure 63: Chess Explorer

Esta doble amenaza fuerza a las blancas a perder una de las piezas mayores, lo que les permitió a las piezas negras tomar la ventaja durante la partida, aun así, al no poseer conocimientos heurísticos precisos los movimientos de las piezas no serán los más convenientes. Aún con la ventaja de las piezas negras la partida terminó en empate.

Bastante interesante el algoritmo de máximos y mínimos, ya que, por muchas técnicas de optimización para el árbol de búsqueda, lo que debemos hacer principalmente es explorar el árbol al completo para tomar la mejor decisión con certeza, pero en este caso el tiempo de

ejecución y su exponencial crecimiento lo hace inviable de forma que a día de hoy a pesar de ser la forma correcta sea imposible de aplicar para profundidades elevadas.

7.2 Turochamp

Con anterioridad vimos la heurística desarrollada por Turin y Champernowne, esta heurística estaba preparada para ejecutarse sobre un algoritmo de búsqueda de árbol como el Minimax, por lo tanto, implementar una heurística de este estilo hace que sea el perfecto compañero para nuestro algoritmo.

Para la implementación de este algoritmo he preparado otro cuaderno de Google Colaboratory que contiene la heurística. Lo primero que debemos hacer es instalar la librería de chess y crear nuestro tablero:

```
[1] !pip install chess
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting chess
  Downloading chess-1.9.4-py3-none-any.whl (149 kB)
    149.1/149.1 kB 4.8 MB/s eta 0:00:00
Installing collected packages: chess
Successfully installed chess-1.9.4

▶ import chess
import numpy as np

board = chess.Board()
board
```

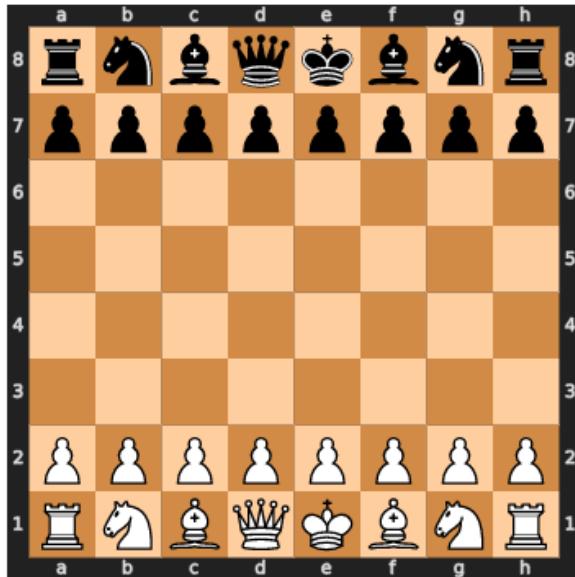
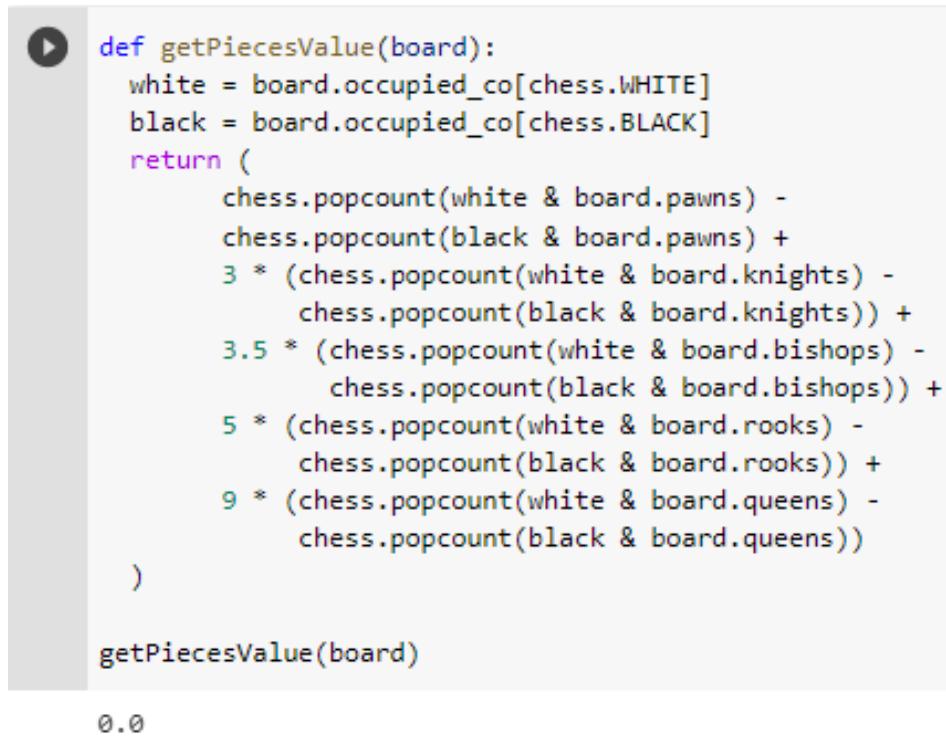


Figure 64: Menacho, E.

Una vez creado el tablero ya solo queda seguir los pasos anteriormente citados para calcular el valor de un estado del juego, lo siguiente que haremos será calcular el valor de las piezas en el tablero, para ello usaremos la función que obtenía el valor de las piezas en el algoritmo de Minimax, solo que en este caso usaremos 3.5 como valor de los alfiles, ya que así lo sugirió Turing.



```
def get_pieces_value(board):
    white = board.occupied_co[chess.WHITE]
    black = board.occupied_co[chess.BLACK]
    return (
        chess.popcount(white & board.pawns) -
        chess.popcount(black & board.pawns) +
        3 * (chess.popcount(white & board.knights) -
              chess.popcount(black & board.knights)) +
        3.5 * (chess.popcount(white & board.bishops) -
                chess.popcount(black & board.bishops)) +
        5 * (chess.popcount(white & board.rooks) -
              chess.popcount(black & board.rooks)) +
        9 * (chess.popcount(white & board.queens) -
              chess.popcount(black & board.queens))
    )

get_pieces_value(board)
```

0.0

Figure 65: Menacho, E.

El siguiente valor que debemos calcular es la movilidad que lo calcularemos obteniendo el número de movimientos posibles para todas aquellas piezas que no sean peones o reyes y calculando la raíz de ese resultado:

```
▶ def getMobility(board):
    res = 0

    # Iterate over all squares on the board
    for square in chess.SQUARES:
        piece = board.piece_at(square)

        # If the piece is a king or a pawn stop
        if piece is None or piece.color != board.turn or piece.piece_type == 1 or piece.piece_type==6:
            continue

        moves = board.legal_moves
        aux = 0
        for move in moves:
            m = str(move)[0:2]
            s = chess.square_name(square)

            if s == m:
                aux = aux + 1

        res = res + aux

    return np.sqrt(res)

mobility = getMobility(board)
print(mobility)
```

2.0

Figure 66: Menacho, E.

A continuación, calculamos la seguridad de las torres, los alfiles y los caballos. Para ello chequeamos las piezas que están defendidas y le sumamos 1.5 si está defendida más de una vez y 1 si sólo lo estuviera una vez.

```
def pieceSecurity(board):
    defending_color = board.turn
    res = 0

    # loop through all the squares on the board
    for square in chess.SQUARES:
        # check if the square has a piece on it and if it is a N, B or R
        piece = board.piece_at(square)
        if piece and (piece.piece_type == 4 or piece.piece_type == 2 or piece.piece_type==3):
            # check if the piece on the square is of the defending color
            if board.color_at(square) == defending_color:
                # get the attackers of the piece on the square
                attackers = board.attackers(defending_color, square)
                # check if there's more than one defender in the square with a piece
                aux = len(list(attackers))

                if aux > 1:
                    res = res + 1.5
                else:
                    res = res + aux
    return res

    # print the defended pieces
print("Result:", pieceSecurity(board))

Result: 4
```

Figure 67: Menacho, E.

Calculamos la movilidad del rey, para ello hacemos lo mismo que en la movilidad, pero en este caso solo analizamos los movimientos del rey.

```
def kingMobility(board):
    res = 0

    # Iterate over all squares on the board
    for square in chess.SQUARES:
        piece = board.piece_at(square)

        if piece is None or piece.color != board.turn or piece.piece_type != 6:
            continue

        moves = board.legal_moves
        aux = 0
        for move in moves:
            m = str(move)[0:2]
            s = chess.square_name(square)

            if s == m:
                aux = aux + 1

        res = res + aux

    return np.sqrt(res)

kingMob = kingMobility(board)
print(kingMob)
```

0.0

Figure 68: Menacho, E.

Ahora calculamos la seguridad del rey, para ello colocamos una reina en su lugar y calculamos su movilidad, después substraemos ese valor:

```
def kingSecurity(board):
    #Create a copy of the actual board
    board2 = board.copy()
    kingPos = board2.king(board.turn)

    #Replace a queen for the king
    board2.set_piece_at(kingPos, chess.Piece(chess.QUEEN, chess.WHITE))

    moves = board2.legal_moves
    aux = 0
    for move in moves:
        m = str(move)[0:2]
        s = chess.square_name(kingPos)

        if s == m:
            aux = aux + 1

    return -np.sqrt(aux)

print(kingSecurity(board))
-0.0
```

Figure 69: Menacho, E.

A continuación, calculamos el valor de los enroques, añadiendo uno por cada enroque posible y otro más por cada enroque que fuera posible en el siguiente movimiento.

```
def castleValue(board):
    # count the number of possible castles
    num_castles = 0
    if board.turn:
        if board.has_kingside_castling_rights(chess.WHITE):
            num_castles += 1
        if board.has_queenside_castling_rights(chess.WHITE):
            num_castles += 1
    else:
        if board.has_kingside_castling_rights(chess.BLACK):
            num_castles += 1
        if board.has_queenside_castling_rights(chess.BLACK):
            num_castles += 1

    moves = board.legal_moves
    for move in moves:
        if str(board.san(move)) == "O-O":
            num_castles += 1
        if str(board.san(move)) == "O-O-O":
            num_castles +=1

    return num_castles

castleVal = castleValue(board)

print(castleVal)
```

Figure 70: Menacho, E.

Calculamos el valor de los peones, para ello añadimos 0.3 si el peón estuviera defendido y añadimos otros 0.2 por cada fila que avance el peón.

```
def pawnValues(board):
    defending_color = board.turn
    res = 0

    # loop through all the squares on the board
    for square in chess.SQUARES:
        # check if the square has a piece on it and if it is a R,B or K
        piece = board.piece_at(square)
        if piece and piece.piece_type == 1 and piece.color==defending_color:
            # check if the piece on the square is of the defending color
            fila = chess.square_rank(square)

            if defending_color:
                if fila >=2:
                    res = res + (fila*0.2) - 0.2
            else:
                aux = 7-fila
                if aux >=2:
                    res = res + (fila*0.2) - 0.2

            if board.color_at(square) == defending_color:
                # get the attackers of the piece on the square
                attackers = board.attackers(defending_color, square)
                # check if there's more than one defender in the square with a piece
                aux = len(list(attackers))

                if aux >= 1:
                    res = res + 0.3

    return res

pawnVal = pawnValues(board)
print(pawnVal)
```

2.4

Figure 71: Menacho, E.

Por último, calculamos las amenazas de jaque y de jaque mate que tengamos en el siguiente movimiento, por cada una añadimos 1 al resultado final.

```

def mateCheckThreats(board):
    # count the number of checks and mates in the next move
    num_checks = 0
    num_mates = 0
    for move in board.legal_moves:
        board.push(move)
        if board.is_check():
            num_checks += 1
        if board.is_checkmate():
            num_mates += 1
        board.pop()

    return num_checks + num_mates

mateChecks = mateCheckThreats(board)
print(mateChecks)

```

0

Figure 72: Menacho, E.

Una vez tenemos todo lo necesario para evaluar una posición, creamos una función capaz de evaluar la posición entera por un tablero dado.

```

def turochamp(board):
    piecesValue = getPiecesValue(board)
    mobility = getMobility(board)
    security = pieceSecurity(board)
    kingMob = kingMobility(board)
    kingSec = kingSecurity(board)
    castle = castleValue(board)
    pawnVal = pawnValues(board)
    matesChecks = mateCheckThreats(board)

    if board.turn:
        return piecesValue + mobility + security + kingMob + kingSec + castle + pawnVal + matesChecks
    else:
        return (-1)*(piecesValue + mobility + security + kingMob + kingSec + castle + pawnVal + matesChecks)

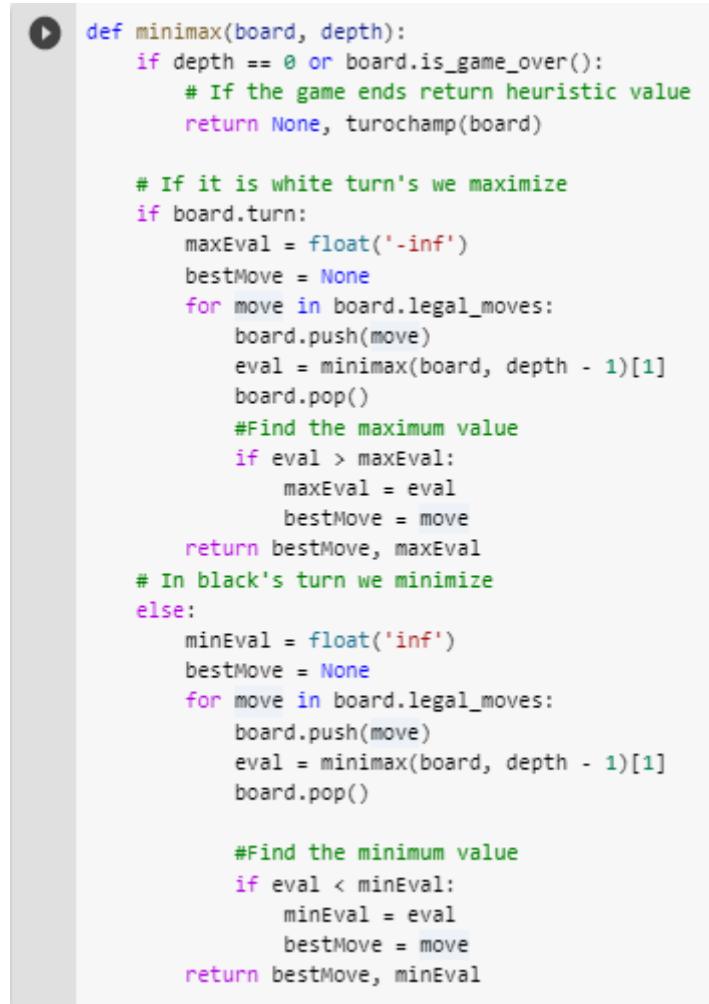
print(turochamp(board))

```

10.4

Figure 73: Menacho, E.

Para probar la heurística usamos el algoritmo de Minimax que usamos con anterioridad, pero en este caso añadiremos la función de evaluación de Turochamp y comprobaremos si mejoran nuestros resultados.



```
def minimax(board, depth):
    if depth == 0 or board.is_game_over():
        # If the game ends return heuristic value
        return None, turochamp(board)

    # If it is white turn's we maximize
    if board.turn:
        maxEval = float('-inf')
        bestMove = None
        for move in board.legal_moves:
            board.push(move)
            eval = minimax(board, depth - 1)[1]
            board.pop()
            #Find the maximum value
            if eval > maxEval:
                maxEval = eval
                bestMove = move
        return bestMove, maxEval
    # In black's turn we minimize
    else:
        minEval = float('inf')
        bestMove = None
        for move in board.legal_moves:
            board.push(move)
            eval = minimax(board, depth - 1)[1]
            board.pop()

            #Find the minimum value
            if eval < minEval:
                minEval = eval
                bestMove = move
        return bestMove, minEval
```

Figure 74: Menacho, E.

Al igual que antes enfrentaremos al motor de ajedrez contra sí mismo y comprobamos si el estilo de juego de nuestro algoritmo varía un poco.

```
moves = []
cont = 1

# While game doesn't ends
while not board.is_game_over() or board.can_claim_draw():

    # White move

    # Get move
    move, eval = minimax(board, 2)
    move = board.san(move)
    # Make move
    board.push_san(move)
    moves.append(move)
    print("-----[" + str(cont) + "]-----")
    print("WHITE----->" + move)

    # Black move
    if board.is_game_over() or board.can_claim_draw():
        break;

    # Get move
    move2, eval = minimax(board, 2)
    # Make move
    move2 = board.san(move2)
    board.push_san(move2)
    moves.append(move2)
    print("BLACK----->" + move2 + "\n")
    print(board)
    print("\n")
    cont += 1

-----[ 1 ]-----
WHITE----->Nf3
BLACK----->g5

r n b q k b n r
p p p p p p . p
. . . . .
. . . . p .
. . . . .
. . . . N . .
P P P P P P P P
R N B Q K B . R
```

Figure 75: Menacho, E.

La partida ocurrió de la siguiente forma:

Minimax + Turochamp (2) vs Minimax + Turochamp (2)

1. Nf3 g5 2. Nc3 g4 3. Ne5 b5 4. Nxb5 Na6 5. e3 Nb8 6. Qxg4 h6 7. Nxf7 Rh7 8. Nxd8 h5 9. Qxg8 h4 10. Qxh7 h3 11. Nf7 a6 12. Qg6 Ra7 13. Nxa7 a5 14. Nxc8 a4 15. gxh3 a3 16. bxa3 Na6 17. Bxa6 c5 18. Bb2 c4 19. Bc3 d5 20. h4 Kd7 21. Nb6+ Ke8 22. Nxd5 Kd7 23. Bb5+ Kc8 24. Qb6 Bh6 25. Nxh6 e5 26. Nf5 e4 27. Bb4 c3 28. d3 exd3 29. cxd3 c2 30. Ba6+ Kd7 31. Bb5+ Kc8 32. Ba6+ Kd7 33. Bb5+ Kc8 34. Qc7++

1/0

A pesar de estar en profundidad 2, el algoritmo nos deja en esta partida jugadas brillantes incluso encontrando mejores jugadas en posiciones complejas que requieren de sacrificios, aquí vemos un ejemplo claro en el que nuestra metodología encuentra la mejor jugada sacrificando el caballo por un peón para conseguir ventaja.

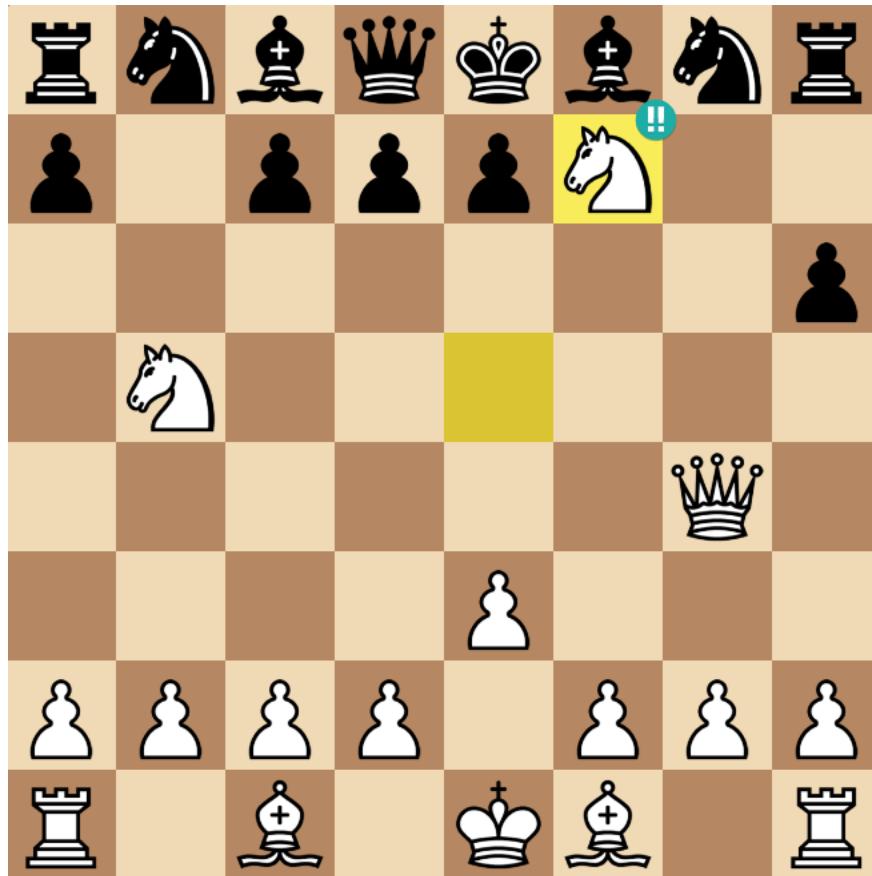


Figure 76: Chess Explorer

Esta jugada es brillante ya que si el rey come en f7 la reina dará jaque en h5 forzando como única jugada Kf6 ya que cualquier otra jugada será un jaque mate inevitable.

+6.96	8... Kf6 9. Bd3 e6 10. Qh4+ Kf7 11. Qxd8 Na6 12. Nxc7 Nxc7 13. Qxc7...
+M4	8... Ke6 9. Bc4+ d5 10. Bd3 Nf6 11. Bf5+ Ke5 12. d4#
+M4	8... Kg7 9. Bd3 Qe8 10. Qxe8 Rh7 11. Qg6+ Kh8 12. Qxh7#

Figure 77: Chess Explorer



Figure 78: Chess Explorer

La partida terminó con un jaque mate por parte de las piezas blancas, es notoria la ventaja de salida, sobre todo en profundidades bajas, esto hace que las blancas tiendan a ganar siempre cuando esto no debería de ser así.

-----WHITE WINS-----

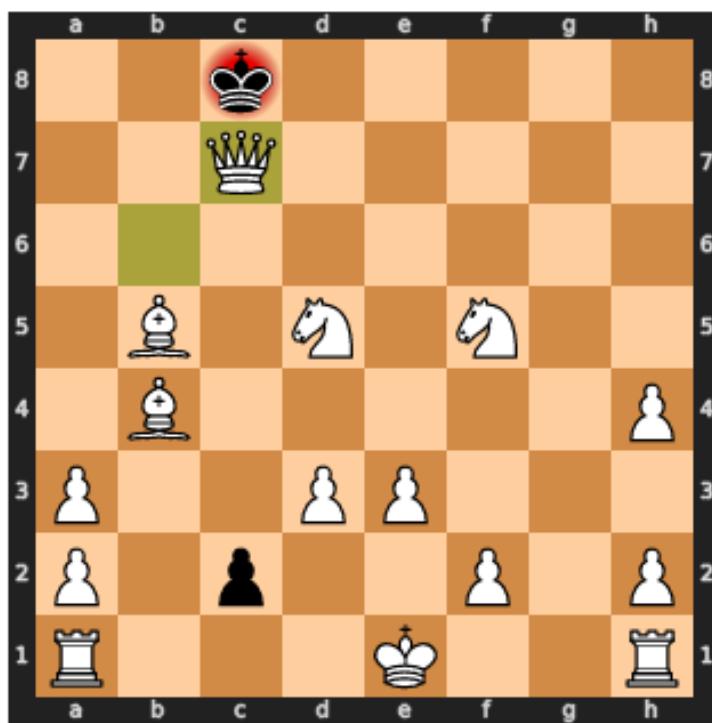


Figure 79: Menacho, E.

A pesar de no reconocer los jaques mate y los jaques, las piezas blancas fueron capaces de propinarle una aplastante victoria a las negras, no perdonando prácticamente ningún error. Observamos el impacto tan fuerte que tienen las heurísticas sobre este tipo de problemas complejos, pudiendo mejorar su eficacia incluso en profundidades bajas.

Al enfrentar el Minimax sin heurística contra el Minimax con Turochamp pasó lo siguiente:

Minimax (2) vs Minimax + Turochamp (2)

1. Nf3 Nh6 2. Nc3 Rg8 3. e3 Rh8 4. Nb5 Rg8 5. a4 Rh8 6. Be2 Rg8 7. h4 Rh8 8. c4
Rg8 9. h5 Rh8 10. b3 Rg8 11. Ba3 Rh8 12. c5 Rg8 13. b4 Rh8 14. d4 Rg8 15. Qc2
Rh8 16. Bb2 Rg8 17. Qxh7 Rh8 18. Qxh8 Ng8 19. Qxg8 Nc6 20. Bc4 Nxd4 21.

Qxf7++

1/0

En la partida Minimax se mantuvo a la defensiva sin hacer avance alguno, de esta forma el algoritmo con heurística comenzó a ganar avance sobre el tablero, haciendo que alcance la ventaja y finalmente de jaque mate tras la posterior adición del reconocimiento de las posiciones victoriosas. La posición en la que se efectuó jaque mate fue la siguiente:

WHITE WINS-----

Figure 80: Menacho, E.

Un claro avance de las piezas blancas sobre el tablero, luego podemos llegar a la conclusión de que el uso de una heurística mejora considerablemente nuestras oportunidades para ganar, incluso si esta tuviera lugar en siglos anteriores.

7.3 Minimax con ponderación Alpha y Beta

La técnica de poda alfa-beta se utiliza para hacer que el algoritmo minimax sea más eficiente. En lugar de considerar cada posible movimiento en cada nodo del árbol, la poda alfa-beta elimina aquellos movimientos que no parecen ser prometedores. Los valores de alfa y beta son límites que representan el mejor resultado que un jugador puede obtener hasta ese momento en el camino hacia una hoja del árbol. Si un nodo encuentra un movimiento que conduce a una ganancia menor que alfa (para un jugador maximizador) o mayor que beta (para un jugador minimizador), se elimina esa rama del árbol y no se consideran más movimientos en esa dirección.

Para el jugador que maximiza, se establece alfa como el valor más alto encontrado hasta ese momento en el nivel actual o niveles superiores. Si se encuentra un nodo con una puntuación mayor que alfa, entonces se actualiza alfa con esa puntuación. Si se encuentra un nodo con una puntuación menor o igual que alfa, se considera que ese nodo no aporta ningún valor adicional y se descarta. Esto significa que la exploración de la rama correspondiente se detiene porque no puede proporcionar una puntuación mayor que alfa.

Para un jugador que minimiza, beta se establece en el valor más bajo encontrado hasta ese momento en el nivel actual o niveles superiores. Si se encuentra un nodo con una puntuación menor que beta, entonces se actualiza beta con esa puntuación. Si se encuentra un nodo con una puntuación mayor o igual que beta, se considera que ese nodo no aporta ningún valor adicional y se descarta. Esto significa que la exploración de la rama correspondiente se detiene porque no puede proporcionar una puntuación menor que beta.

Para inicializar la búsqueda, lo primero que debemos hacer es asignar -Infinito para Alpha e Infinito para Beta. Una vez asignados, consideraremos el siguiente árbol de búsqueda:

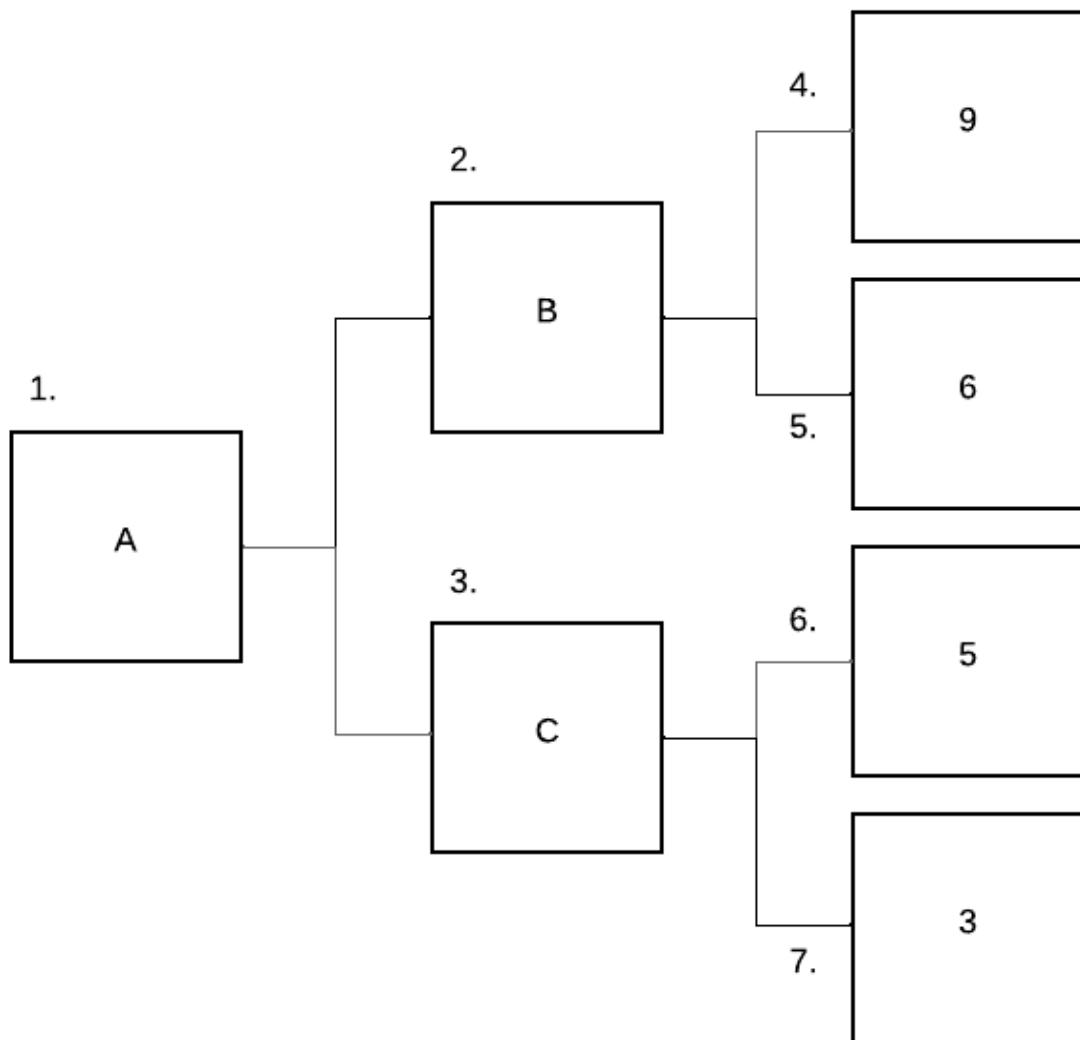


Figure 81: Menacho, E.

En C, debemos encontrar $\max(\text{Alpha}, 3)$, luego tendremos en C un valor de 3. Para decidir si merece la pena seguir explorando el árbol hacia arriba debe cumplir la condición de que Beta $\leq \text{Alpha}$, en este caso $-\text{Infinito} < 3$ luego continuaremos nuestra búsqueda hacia arriba. A continuación, buscamos el mayor de los valores entre el nodo 6 y 7, luego, $\max(5, 3) = 5$, por lo tanto, el valor actual de C será de 5 para Alpha.

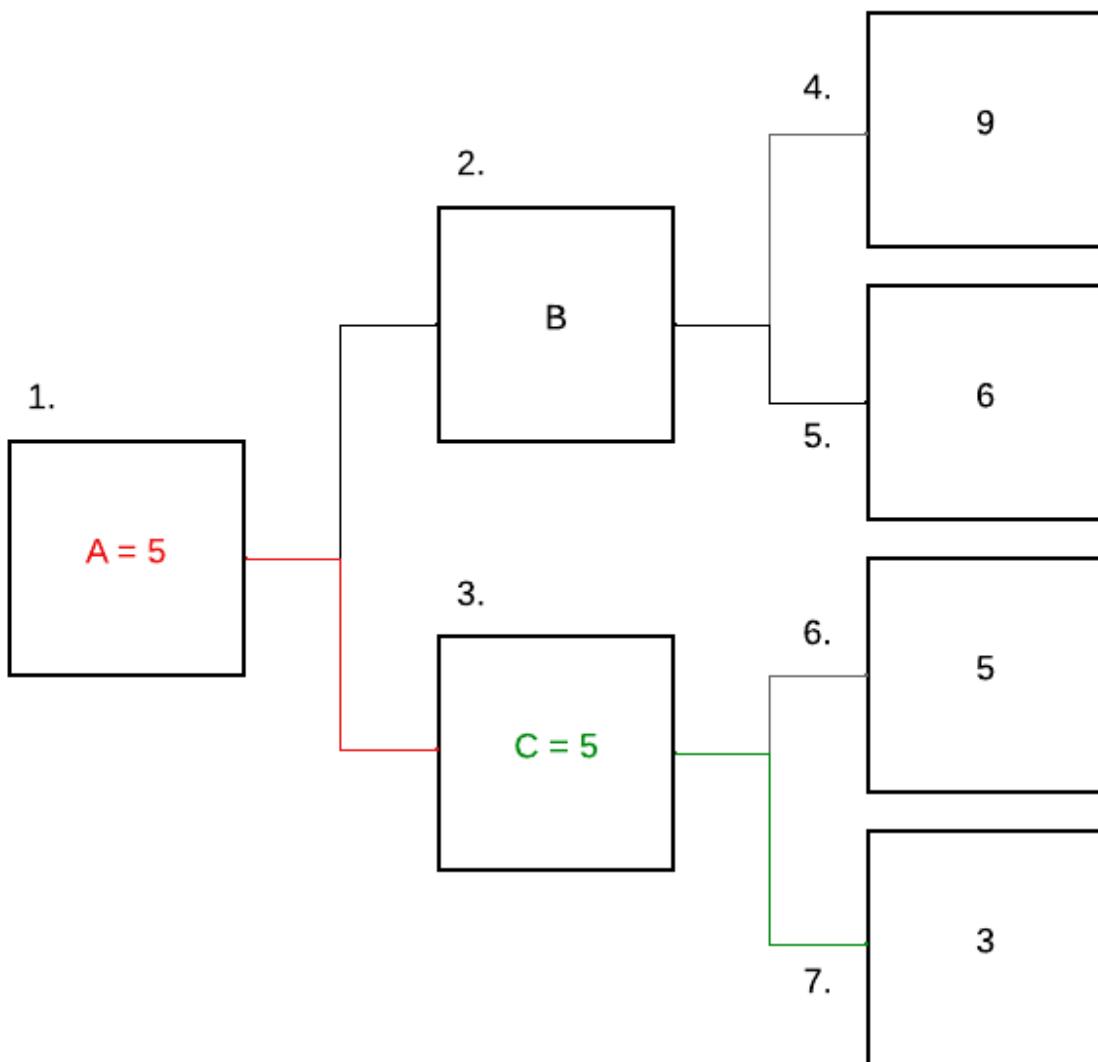


Figure 82: Menacho, E.

Ahora que sabemos el valor del nodo C, podemos calcular A como Beta = min (Infinito, 5) = 5. Ahora, el minimizador tiene garantizado un valor de al menos 5 o menor. A continuación, A llama a B para ver que nodo es más prometedor, si B o C. En B, los valores de Alpha y Beta son de -Infinito para Alpha y 5 para Beta. Así que en el nodo número 5 debemos calcular, Alpha = max (-Infinito, 6) = 6. Como Beta <= Alpha → 5 < 6. Debemos seguir nuestra búsqueda.

Cómo el resultado del nodo B de al menos 6, el minimizador ya sabe que debe truncar la rama y dejar de explorarla ya que el resultado obtenido para C es menor que para B, ya que $5 < 6$.

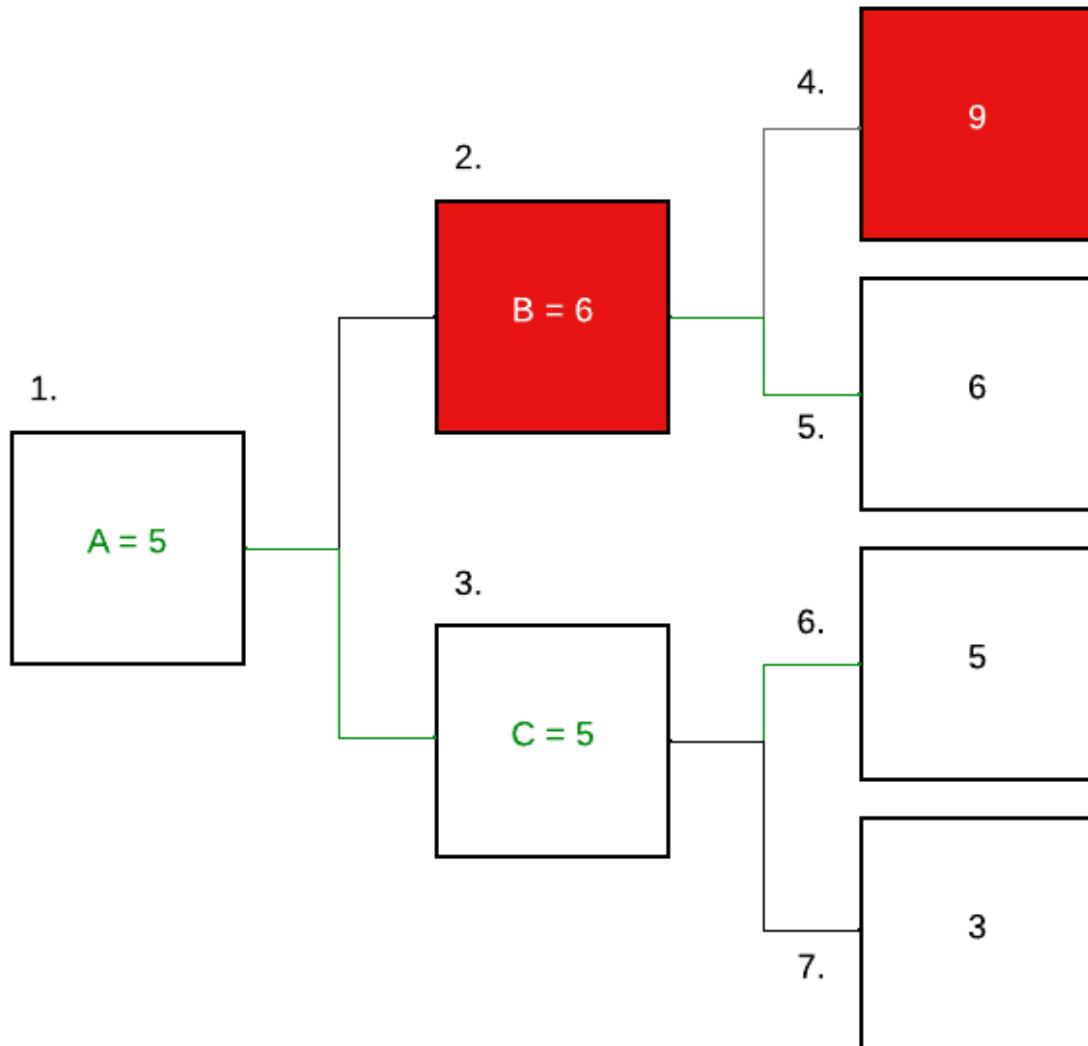


Figure 83: Menacho, E.

Luego podemos descartar la rama al completo, ya que nos aseguramos de un mejor resultado en la rama saliente del nodo C. A continuación, muestro el pseudocódigo para desarrollar el algoritmo.

funcion minimaxAlphaBeta(nodo, profundidad, alpha, beta, esMaximizador):

si profundidad es 0 o el nodo es un nodo terminal:

devuelve el valor heurístico del nodo

si esMaximizador es Verdadero:

mejorValor = -infinito

para cada nodo hijo de nodo:

valor = minimaxAlphaBeta (hijo, profundidad - 1, alpha, beta, Falso)

mejorValor = max(mejorValor, valor)

alpha = max(alpha, mejorValor)

si beta <= alpha:

break

devuelve mejorValor

sino:

mejorValor = +infinito

para cada nodo hijo de nodo:

valor = minimaxAlphaBeta(hijo, profundidad - 1, alpha, beta, Verdadero)

mejorValor = min(mejorValor, valor)

beta = min(beta, mejorValor)

si beta <= alpha:

break

devuelve mejorValor

Lo siguiente que haremos, será crear un cuaderno de Google Colaboratory para el algoritmo, por lo tanto, al igual que en los algoritmos anteriores lo primero de todo será importar la librería de chess en nuestro entorno.

```
!pip install chess
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting chess
  Downloading chess-1.9.4-py3-none-any.whl (149 kB)
                                             149.1/149.1 kB 2.8 MB/s eta 0:00:00
Installing collected packages: chess
Successfully installed chess-1.9.4
```

```
import chess
board = chess.Board()
board
```

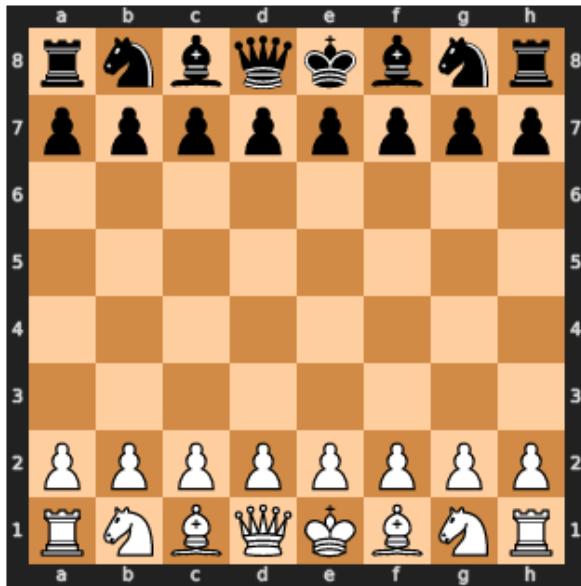


Figure 84: Menacho, E.

Lo siguiente que debemos codificar será nuestra heurística, en este caso, usaremos el valor de las piezas y los jaques mate como función para evaluar nuestras posiciones.



```
def getPiecesValue(board):
    # Si es jaque mate
    if board.is_checkmate():
        if board.turn:
            return -9999
        else:
            return 9999
    white = board.occupied_co[chess.WHITE]
    black = board.occupied_co[chess.BLACK]
    return (
        1 * (chess.popcount(white & board.pawns) -
              chess.popcount(black & board.pawns)) +
        3 * (chess.popcount(white & board.knights) -
              chess.popcount(black & board.knights)) +
        3.5 * (chess.popcount(white & board.bishops) -
                chess.popcount(black & board.bishops)) +
        5 * (chess.popcount(white & board.rooks) -
              chess.popcount(black & board.rooks)) +
        9 * (chess.popcount(white & board.queens) -
              chess.popcount(black & board.queens))
    )
```

Figure 85: Menacho, E.

Ya solo nos queda codificar el algoritmo. Para ello nos basamos en el pseudocódigo proporcionado con anterioridad.

```
def alphabeta(board, depth, alpha, beta, maximizingPlayer):
    if depth == 0 or board.is_game_over():
        return None, getPiecesValue(board)

    if maximizingPlayer:
        bestMove = None
        maxEval = float('-inf')
        for move in board.legal_moves:
            board.push(move)
            _, eval = alphabeta(board, depth - 1, alpha, beta, False)
            board.pop()
            if eval > maxEval:
                maxEval = eval
                bestMove = move
            alpha = max(alpha, eval)
            if beta <= alpha:
                break
        return bestMove, maxEval
    else:
        bestMove = None
        minEval = float('inf')
        for move in board.legal_moves:
            board.push(move)
            _, eval = alphabeta(board, depth - 1, alpha, beta, True)
            board.pop()
            if eval < minEval:
                minEval = eval
                bestMove = move
            beta = min(beta, eval)
            if beta <= alpha:
                break
        return bestMove, minEval
```

Figure 86: Menacho, E.

Comprobamos los resultados de nuestro algoritmo.

```
def minimaxAB(board, depth):
    if board.turn:
        return alphabeta(board, depth, float('-inf'), float('inf'), True)
    else:
        return alphabeta(board, depth, float('-inf'), float('inf'), False)

import time

start_time = time.time()
bestMove, eval = minimaxAB(board, 5)
end_time = time.time()

print("Execution time:", end_time - start_time, "seconds")
print("Best move:", board.san(bestMove), " Eval: ", eval)

Execution time: 0.43221259117126465 seconds
Best move: g3  Eval:  1.0
```

Figure 87: Menacho, E.

Observamos unos tiempos de ejecución muy inferiores a los anteriores para el Minimax sin poda para las ramas. Mucho más eficiente que el algoritmo anterior, aun para llegar al mismo resultado, de todas formas, no conseguimos evitar el crecimiento exponencial del tiempo de ejecución a medida que aumentamos la profundidad.

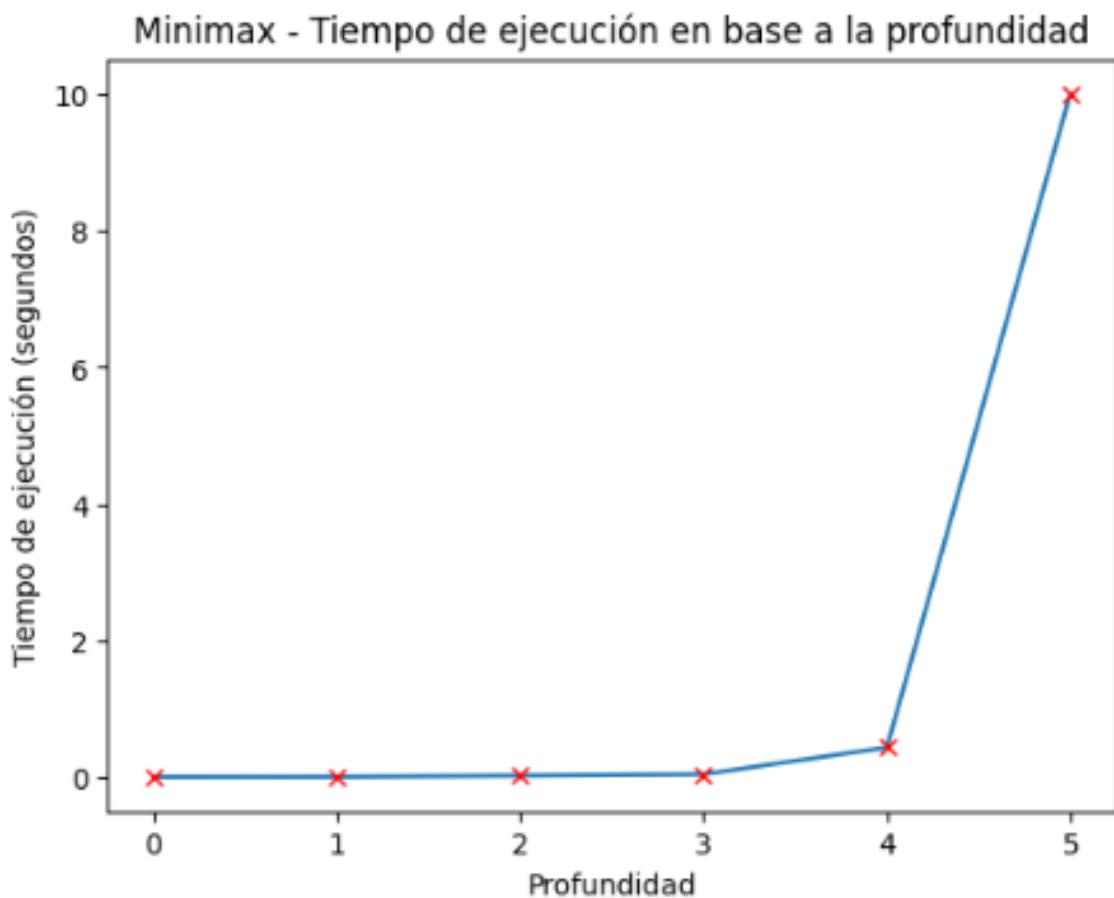


Figure 88: Menacho, E.

La mejora en la velocidad de cómputo es completamente notoria, haciendo que pueda ejecutarse a mayor profundidad. Para ello, pondremos a competir al algoritmo ponderado contra el algoritmo sin ponderar a la misma profundidad y calcularemos los tiempos de ejecución correspondientes.

Primero importamos la función anterior para el Minimax y creamos un código para ejecutar el algoritmo y enfrentar al Minimax sin ponderar con el ponderado.



```
# While game doesn't ends
while not board.is_game_over():

    # White move

    # Get move
    start_time = time.time()
    move, eval = minimaxAB(board, 3)
    end_time = time.time()

    time1.append(end_time - start_time)

    move = board.san(move)
    # Make move
    board.push_san(move)
    moves.append(move)
    print("-----[" + cont + "]-----")
    print("WHITE----->" + move)

    # Black move
    if board.is_game_over():
        break;

    # Get move
    start_time = time.time()
    move2, eval = minimax(board, 3)
    end_time = time.time()

    time2.append(end_time - start_time)
    # Make move
    move2 = board.san(move2)
    board.push_san(move2)
    moves.append(move2)
    print("BLACK----->" + move2 + "\n")
    print(board)
    print("\n")
    cont += 1
```

Figure 89: Menacho, E.

Se jugó exactamente la misma partida que la anterior, esto tiene sentido, ya que evaluó las posiciones de la misma forma, ahora comparemos el tiempo de ejecución de ambos durante el transcurso de la partida.

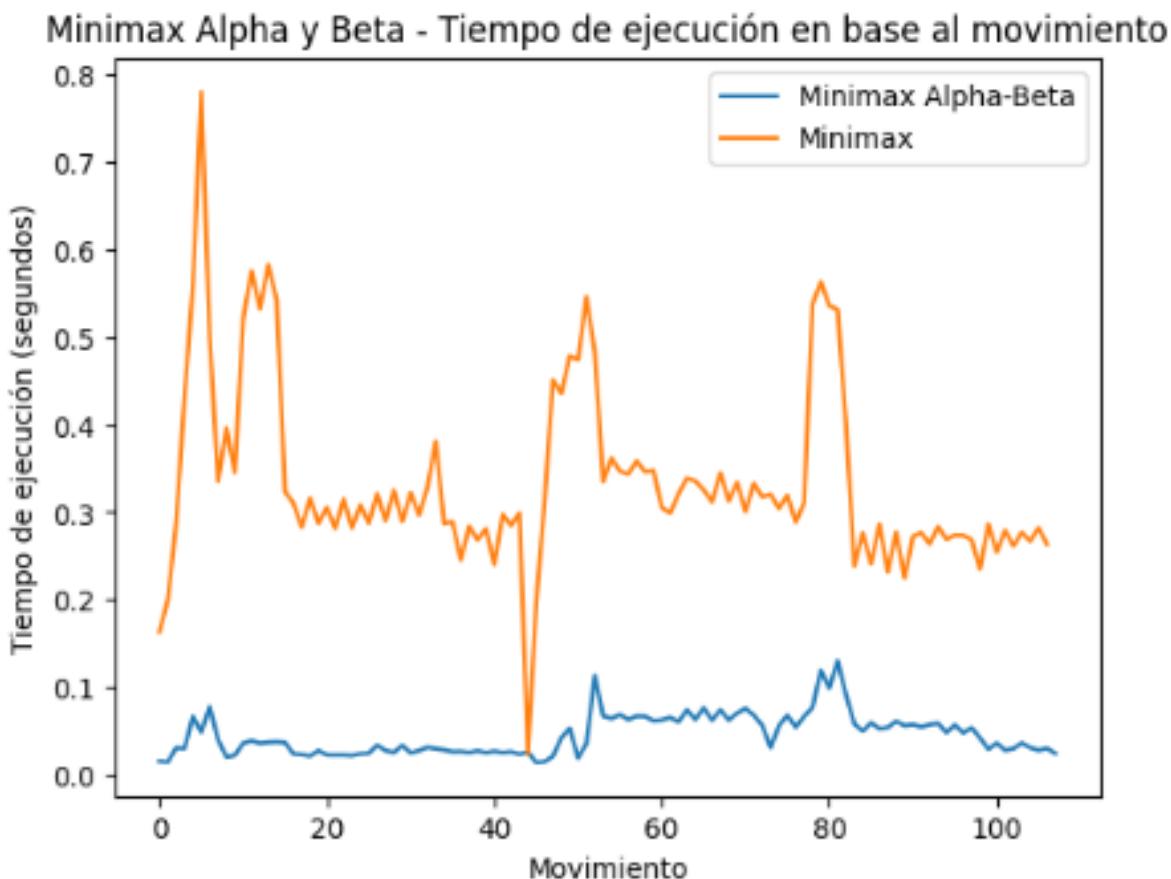


Figure 90: Menacho, E.

Una diferencia brutal en el tiempo de ejecución, observamos una mejora notable en los tiempos de ejecución y quizás esta sea una de las metodologías más acertadas a la hora de calcular el mejor movimiento en una posición.

Para comprobar que nuestro algoritmo funciona correctamente enfrentaremos a la versión del Minimax ponderado (Blancas) en profundidad 5 contra la versión no ponderada en profundidad 3 (Negras).

Minimax (2) vs Minimax + Turochamp (2)

1. g3 Nh6 2. Nh3 Ng4 3. Ng1 Rg8 4. Nh3 Rh8 5. Ng1 Rg8 6. Nh3 Rh8 7. Ng1 Rg8 8. Nh3 Rh8 9. Ng1 Rg8 10. Nh3 Rh8 11. Bg2 Na6 12. Ng5 Rb8 13. Ne4 Rg8 14. Ng5 Rh8 15. Ne4 Rg8 16. Ng5 Rh8 17. Ne4 Rg8 18. Ng5 Rh8 19. Ne4 Rg8 20. Nec3 Ra8 21. Nd5 Rh8 22. Nf4 Rg8 23. Nh5 Rh8 24. Nf4 Rg8 25. Nh5 Rh8 26. Nf4 Rg8 27. Nh5 Rh8 28. Nf4 Rg8 29. Nh5 Rb8 30. Bd5 Rh8 31. Nf4 Rg8 32. Be4 Rh8 33. Nh5 Ra8 34. Nf4 Rb8 35. Nh5 Ra8 36. Nf4 Rb8 37. Nh5 Ra8 38. Nf4 Rb8 39. Nh5 Ra8 40. Bf5 Nh6 41. Bh3 Nb4 42. Nf4 Rg8 43. Nh5 Rh8 44. Nf4 Rg8 45. Nh5 Rh8 46. Nf4 Rg8 47. Nh5 Rh8 48. Nf4 Rg8 49. Nd3 Nc6 50. Bg2 Nd4 51. e3 Ndf5 52. Qh5 Rh8 53. Qg5 Rg8 54. Be4 Rh8 55. Bxf5 Nxsf5 56. Qxf5 Rb8 57. Ne5 f6 58. Qh5+ g6 59. Nxg6 Rg8 60. Nh8+ Rg6 61. Qxh7 Rg7 62. Qh5+ Rf7 63. Qxf7++

1/0

Ganaron las piezas blancas por un jaque mate tras la desventaja de las piezas negras en un error de cálculo, uno de los momentos más interesantes de la partida fue cuando las piezas blancas consiguieron tomar la iniciativa tras un intercambio en el centro del tablero.



Figure 91: Chess Explorer

Una doble amenaza en la que las piezas blancas comen el caballo con el alfil para conseguir ventaja. Otro momento a destacar es cuando encontró una táctica común en ajedrez para amenazar a la torre que está en la esquina. Este es un problema muy común y lo resolvió con bastante facilidad.

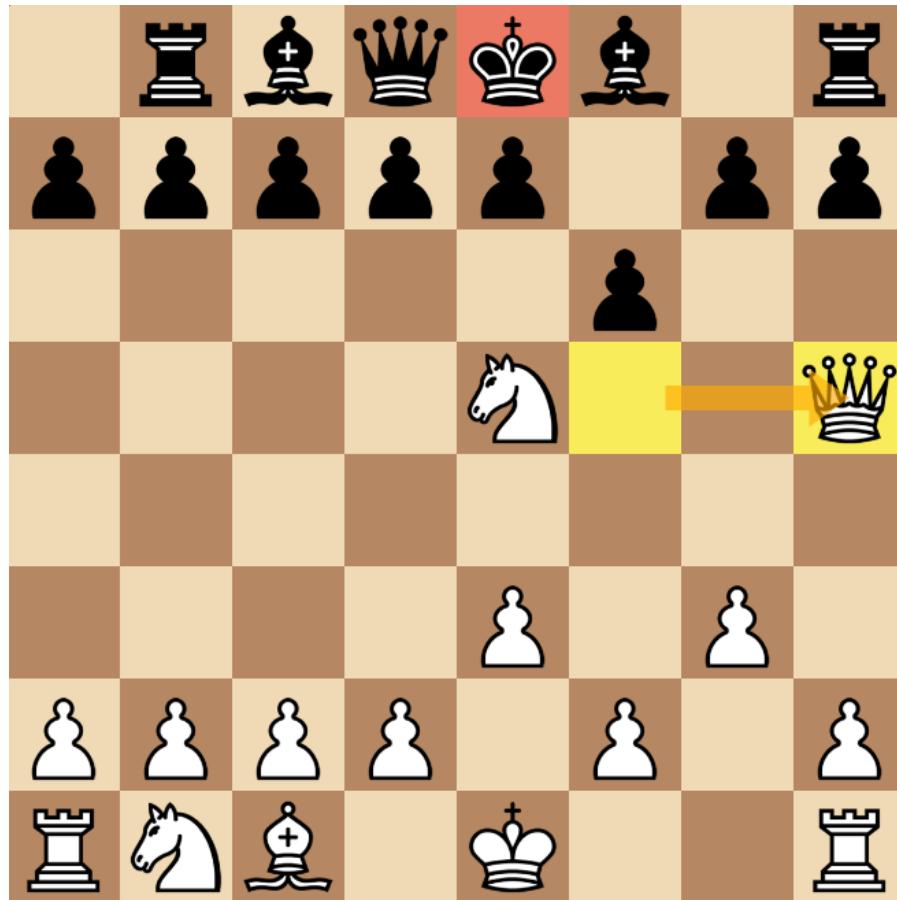


Figure 92: Chess Explorer

Todo comienza con un jaque, haciendo que las piezas negras se vean obligadas a avanzar su peón de la columna g para defenderse del jaque, pero tras esta jugada viene el sacrificio del caballo en g7 para obtener una torre de ventaja, ya que si comen con el peón podemos comer la torre en h8.



Figure 93: Chess Explorer

Y tras este movimiento las piezas negras intentaron defender la torre como pudiera, obviando las intenciones de jaque mate por parte de las piezas negras. La posición de la que proviene la secuencia de mate fue la siguiente:



Figure 94: Chess Explorer

De la que provinieron las siguientes jugadas: 60. Nh8+ Rg6 61. Qxh7 Rg7 62. Qh5+ Rf7 63. Qxf7++. Finalizando con este jaque mate:



Figure 95: Chess Explorer

Por último, haremos la comparativa entre los tiempos de ejecución de ambos algoritmos.

Minimax Alpha y Beta - Tiempo de ejecución en base al movimiento

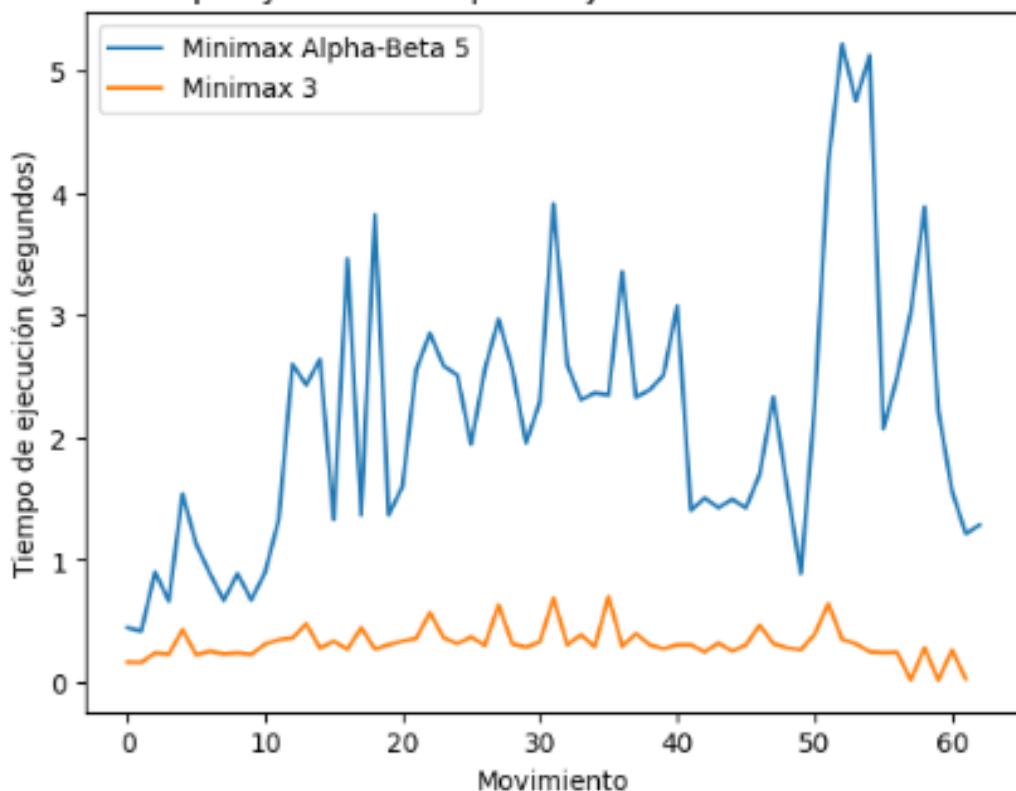


Figure 96: Menacho, E.

Son mayores los tiempos de ejecución para el algoritmo ponderado, pero aun así no están tan distantes en los resultados.

En general, la ponderación Alpha y Beta es una extensión del algoritmo Minimax, diseñada para reducir el número de nodos que el algoritmo necesita explorar para encontrar la mejor jugada en un juego determinado.

En comparación con el algoritmo Minimax tradicional, la ponderación Alpha y Beta puede ser significativamente más rápida y eficiente en términos de uso de recursos. Esto se debe a que el algoritmo elimina la necesidad de explorar ramas que no serán útiles en la solución final. En otras palabras, nos permite “podar” partes del árbol de búsqueda que son irrelevantes para la solución final.

En general, esta técnica es una de las más efectivas para acelerar los algoritmos de búsqueda de juegos. Aunque la ponderación Alpha y Beta no siempre garantiza la solución óptima, en la práctica, la diferencia de calidad entre la solución producida por la ponderación Alpha y Beta y la solución óptima completa puede ser mínima, especialmente cuando se usa con una heurística efectiva. Por lo tanto, este algoritmo se considera una técnica valiosa para acelerar los algoritmos de búsqueda de juegos sin comprometer en gran medida la calidad de la solución final.

8. OTRA PERSPECTIVA

8.1 Redes convolucionales para la predicción de jaques sin conocimiento

Para continuar con la investigación en este juego, haremos un cambio en el campo de investigación ya que la aparición de la inteligencia artificial ha sido un fuerte impacto en la forma de ver nuestra vida cotidiana. Existen algoritmos que son capaces de adaptarse a cualquier problema complejo, por esto me surgió la duda de si sería posible entrenar un algoritmo para aprender las reglas del juego. Esto sería interesante ya que, si fuera posible, quien sabe si fuera posible crear una inteligencia autónoma capaz de aprender a jugar cualquier juego.

Por esto, el siguiente fragmento de la investigación consistirá en desarrollar un sistema complejo para predecir si una posición es jaque o no, de esta forma nos aseguramos de que nuestro modelo sea capaz de aprender las mismas.

Para este procedimiento usaremos las Redes Neuronales Convolucionales (CNN). Estas son un tipo de red neuronal que está basado en una operación llamada convolución. Esta consiste en una operación matemática entre dos funciones (f y g), la cual produce una tercera función ($f * g$) que expresa como la forma de uno es modificada por el otro. Formalmente se expresa con un $*$. Su fórmula es la siguiente:

$$(f * g)(t) \doteq \int_{-\infty}^{\infty} f(\eta)g(t - \eta)d\eta$$

Figure 97: Convolution formula

Como podemos observar se calcula el área que se produce al cruzar ambas funciones en un punto determinado, de esta forma conseguimos una expresión que nos define a la anterior. Esta

operación es muy utilizada en las matemáticas, un ejemplo de uso de la misma es a la hora de filtrar frecuencias. Podríamos transformar una frecuencia alta en una baja.

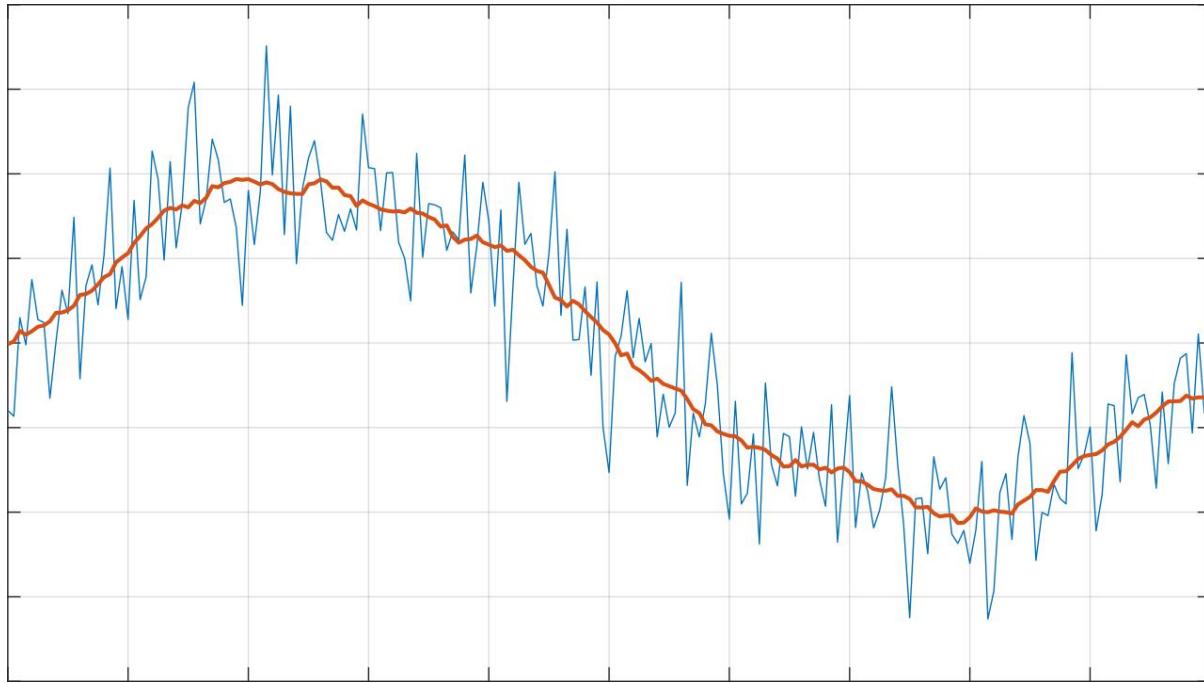


Figure 98: Convolution filter

El resultado de una convolución de una función f (en este la frecuencia de color azul) sobre un filtro g nos da como resultado la frecuencia de color rojo. Esto es un ejemplo de aplicación de convolución para transformar una onda de alta frecuencia a baja frecuencia.

Ahora que sabemos cómo funciona la convolución, es hora de entender como es la metodología a seguir cuando queremos aplicar una red convolucional. Es importante entender primero algunos conceptos clave que ocurren con este tipo de redes.

En este caso, ya que queremos predecir si una posición es jaque debemos entender que se trata de un problema de clasificación y no de predicción. Entonces podemos entender que queremos asignarle un valor o etiqueta a una posición. En resumen, queremos que nuestra red cuando reciba una posición de ajedrez esta la clasifique con dos etiquetas “Jaque” o “No Jaque”. Si

quisiéramos que nos diera un valor numérico se daría el caso de la predicción, que es otra posibilidad para las redes convolucionales.

A continuación, explicaré paso por paso cómo funcionan:

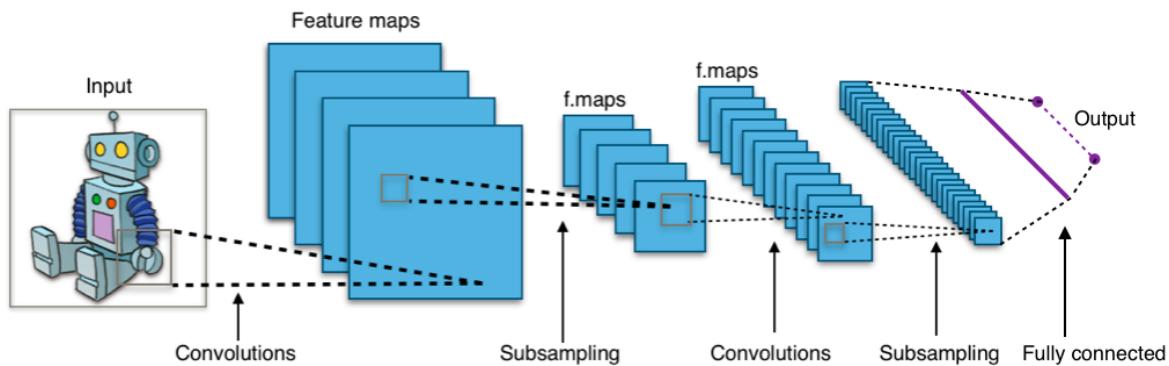


Figure 99: CNN Structure

Entrada: La entrada de una red convolucional puede ser una imagen o una señal. Normalmente las imágenes se representan como una matriz tridimensional en la que encontramos la altura, el ancho y canal de color (RGB) de la imagen.

Convolución: La primera de las capas por la que pasará nuestra imagen será un filtro de convolución. En esta capa, se utilizan filtros o pequeñas plantillas que se desplazan por la imagen para detectar patrones específicos. Los filtros multiplican los valores de píxeles en cada sección de la imagen y producen un nuevo mapa de características que resalta las partes importantes de la imagen. De esta manera, la red neuronal convolucional puede identificar objetos o características importantes en la imagen, extraemos los mapas de la imagen recibida tras aplicar el kernel. En otras palabras, estaríamos extrayendo los atributos para hacer las predicciones a posteriori.

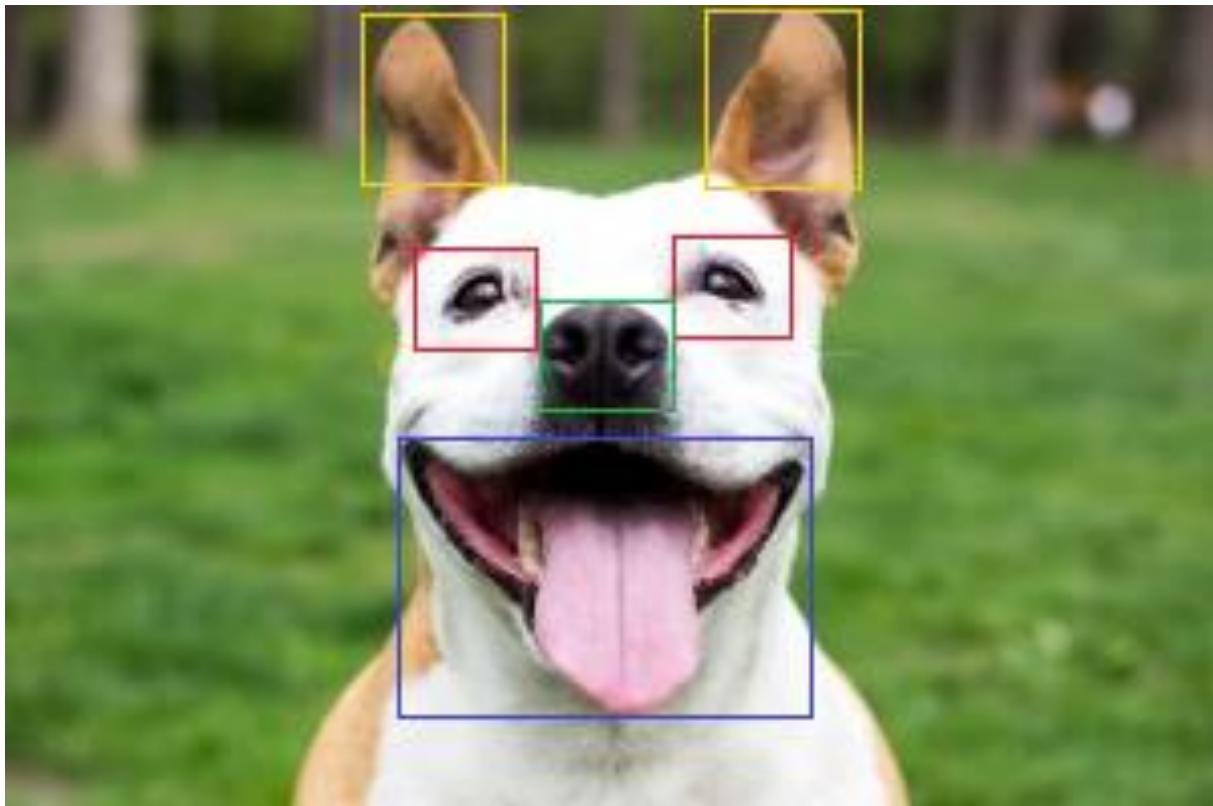


Figure 100: Menacho, E.

Función de activación: La función de activación se aplica después de la convolución para permitir que la red neuronal convolucional capture patrones más complejos en la imagen. La función de activación no lineal (ReLU es un ejemplo de función de activación) le da a la red la capacidad de aprender características más complejas en la imagen, lo que mejora la capacidad de la red para representar la información visual. En resumen, la función de activación ayuda a la red a "aprender" y "entender" mejor las características visuales de la imagen.

Pooling: En la capa de pooling, se reduce la dimensión del mapa de características. Esto se logra mediante la aplicación de una operación de agrupamiento (como el "max pooling" o el "average pooling"), que básicamente agrupa varios píxeles cercanos para reducir la cantidad de información en la imagen. La operación de agrupamiento reduce la cantidad de parámetros que la red neuronal convolucional debe procesar, lo que a su vez ayuda a mejorar la capacidad

del modelo para generalizar y predecir correctamente. En resumen, el pooling ayuda a reducir la complejidad del modelo y mejora su capacidad para manejar datos nuevos y no vistos anteriormente.

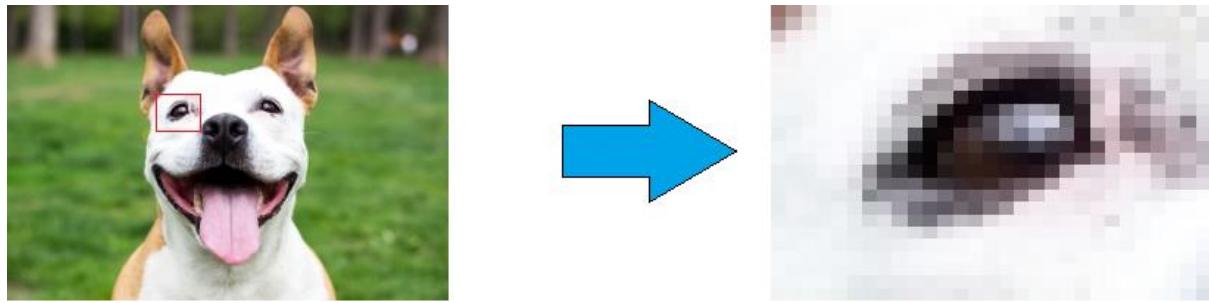


Figure 101: Menacho, E.

Capas conectadas: La red neuronal convolucional utiliza una o varias capas completamente conectadas para clasificar la imagen. Estas capas son similares a las capas de una red neuronal tradicional y están diseñadas para procesar las características extraídas por las capas anteriores. Las capas completamente conectadas utilizan la información aprendida en las capas anteriores para producir una salida final, que es la predicción de la red neuronal convolucional. En resumen, las capas completamente conectadas funcionan como un clasificador final que utiliza las características extraídas de las capas anteriores para producir una predicción final sobre la imagen.

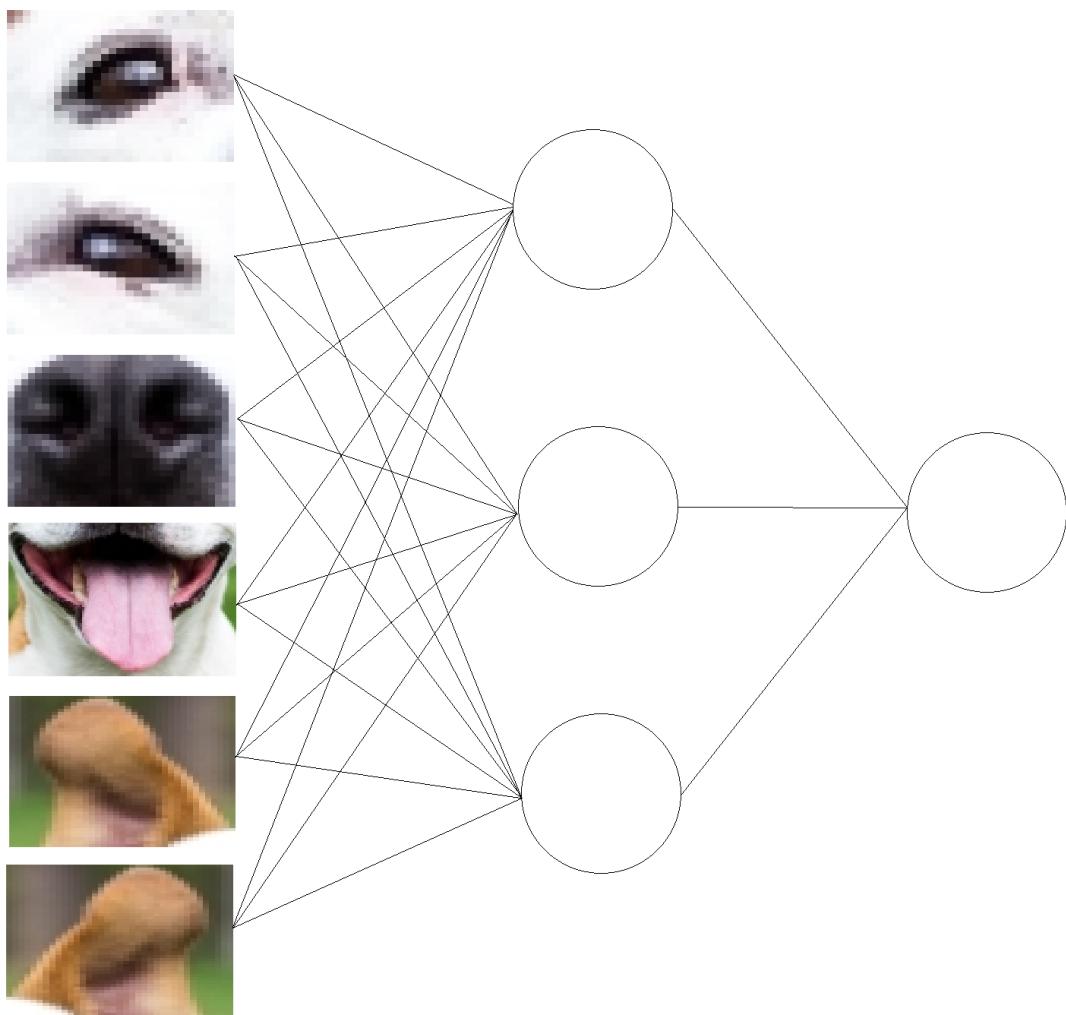


Figure 102: Menacho, E.

Salida:

Por último, una vez conectadas las capas podemos lanzar una predicción sobre la imagen y darle una etiqueta a la entrada recibida. Como podría ser “perro” o “gato”.

Lo primero que debemos hacer para poder entrenar nuestra red convolucional es obtener los datos suficientes para poder tratarlos a posteriori, para ello utilizo un IDE llamado Spyder que me permite tratar las estructuras de datos con más facilidad. Para ello extraeremos de una posición de ajedrez varios mapas que contengan las posiciones de las piezas, para así convertir

la posición en una matriz tridimensional, para que se pueda estimar los valores con mejor precisión que con una bidimensional.

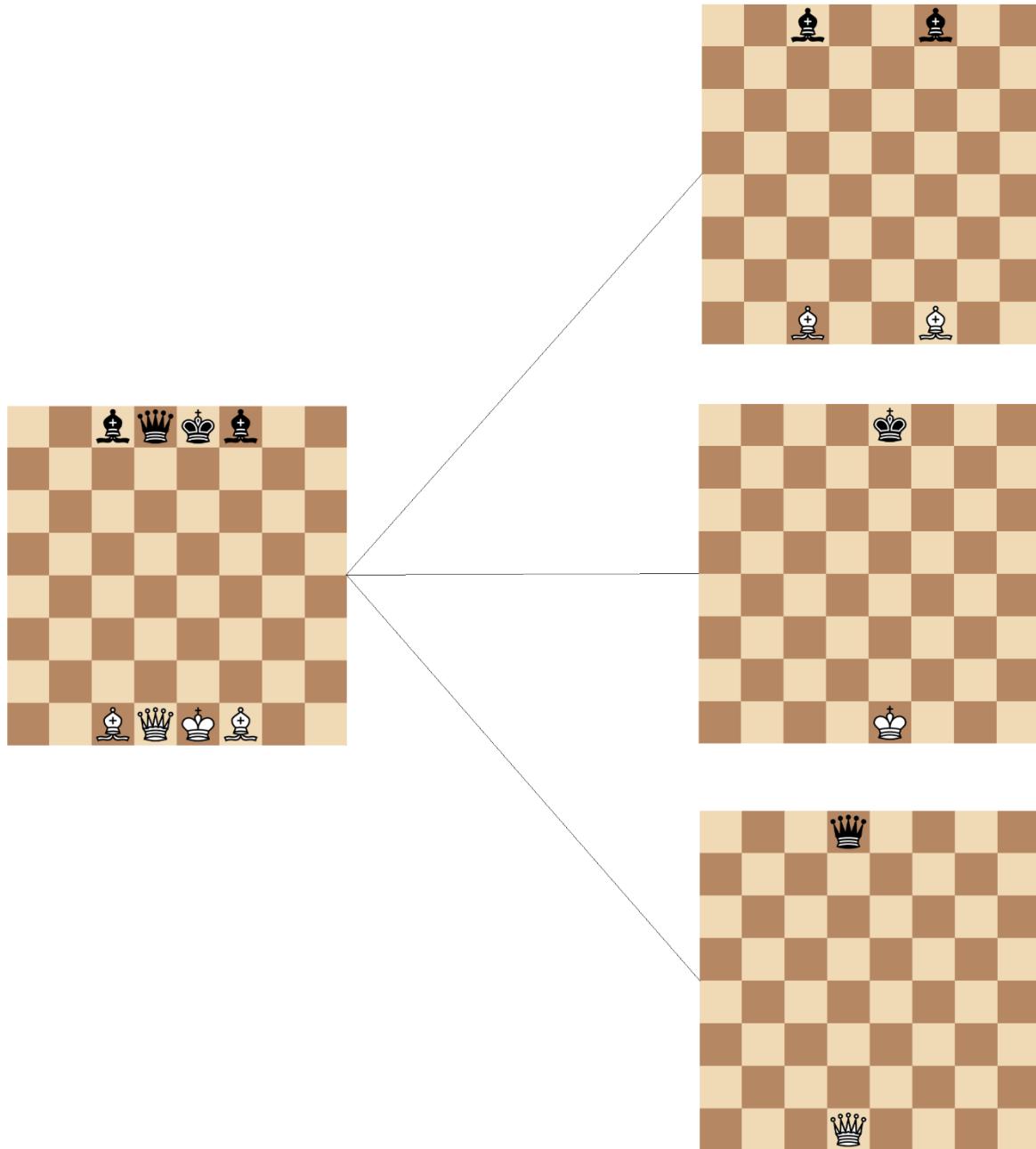


Figure 103: Menacho, E.

Si tuviéramos una posición así de ajedrez querríamos crear un mapa como el anterior, creando matrices con la posición en la que se encuentran las piezas. En este caso lo hacemos solo para la reina, el rey y los alfiles, pero deberíamos hacerlo para cada pieza que se encuentre en el tablero.

A continuación, muestro los códigos utilizados en Python para la extracción de los datos.

Para ello creamos un archivo cuyo nombre sea jaquesFile.py y importamos las siguientes librerías:

```
import chess
import chess.engine
import pandas as pd
import numpy as np
import os
```

Figure 104: Menacho, E.

A continuación, descargamos el motor “Stockfish” para enfrentarlo con un motor aleatorio, de esta forma lograremos sacar los jaques de manera más rápida.

📁 stockfish_15.1_win_x64_popcnt	07/05/2023 19:50	Carpeta de archivos
📄 filemakerFullJaques.py	13/03/2023 13:20	Python File
📄 jaques.json	07/05/2023 19:50	Archivo de origen ...
📄 jaquesFile.py	07/05/2023 19:49	Python File
📄 posiciones.json	07/05/2023 19:50	Archivo de origen ...

Figure 105: Menacho, E.

Lo siguiente será crear una función para crear la matriz:

```
def make_matrix(board): #type(board) == chess.Board()
    pgn = board.epd()
    foo = [] #Final board
    pieces = pgn.split(" ", 1)[0]
    rows = pieces.split("/")
    for row in rows:
        foo2 = [] #This is the row I make
        for thing in row:
            if thing.isdigit():
                for i in range(0, int(thing)):
                    foo2.append(0)
            else:
                if thing == 'P':
                    foo2.append(1)
                elif thing == 'N':
                    foo2.append(2)
                elif thing == 'B':
                    foo2.append(3)
                elif thing == 'R':
                    foo2.append(4)
                elif thing == 'Q':
                    foo2.append(5)
                elif thing == 'K':
                    foo2.append(6)

                if thing == 'p':
                    foo2.append(-1)
                elif thing == 'n':
                    foo2.append(-2)
                elif thing == 'b':
                    foo2.append(-3)
                elif thing == 'r':
                    foo2.append(-4)
                elif thing == 'q':
                    foo2.append(-5)
                elif thing == 'k':
                    foo2.append(-6)

    foo.append(foo2)
return foo
```

Figure 106: Menacho, E.

Otra función para hacer movimientos aleatorios y otra para realizar movimientos por parte de la máquina.

```
def engineMove(board):
    engine = chess.engine.SimpleEngine.popen_uci(
        r"./stockfish_15.1_win_x64_popcnt/stockfish-windows-2022-x86-64-modern.exe")
    result = engine.play(board, chess.engine.Limit(time=0.02))
    best_move = result.move
    engine.quit()
    return best_move

def randomMove(board):
    possible_moves = []
    moves = board.legal_moves

    for move in moves:
        possible_moves.append(move)

    return possible_moves[np.random.randint(len(possible_moves))]
```

Figure 107: Menacho, E.

Nos falta el bucle principal para jugar las partidas y poder crear nuestro fichero con la longitud deseada. También debemos comprobar si alguna posición es jaque y si así lo fuera, darle un valor de 1 al resultado final.

```

res = []
res2 = []
j = 0
i = 0

while i < 10 and j < 20:

    board = chess.Board()
    while board.is_game_over():
        board.push(engineMove(board))
        if board.is_game_over():
            break

        if not board.turn and board.is_check():
            print("POSITION (" + str(j) +
                  ")-----")
            print(board)
            res.append(make_matrix(board))
            j = j + 1
    else:
        res2.append(make_matrix(board))

    board.push(randomMove(board))

# save the DataFrame as a JSON file
df = pd.DataFrame({'posiciones': res})
df.to_json('jaques.json')

# save the DataFrame as a JSON file
df2 = pd.DataFrame({'posiciones': res2})
df2.to_json('posiciones.json')

```

Figure 108: Menacho, E.

Ahora que tenemos las matrices con las posiciones de jaque y no jaque nos queda crear los ficheros de testeo de la red y los que son para entrenar a la misma. Para esto creamos otro archivo llamado fileMakerFullJaques.py:

```
import pandas as pd

#Posicione de jaque

df3 = pd.read_json('jaques.json').reset_index(drop=True)
df3['y'] = 1

#Posiciones sin mate en las que mueven las negras para test y entrenamiento
testJaques = df3.tail(3000).reset_index(drop=True)
trainJaques = df3.iloc[:-3000].reset_index(drop=True)

df4 = pd.read_json('posiciones.json').drop_duplicates().reset_index(drop=True)
df4 = df4[0:23000]
df4['y'] = 0

#Posiciones sin mate en las que mueven las negras para test y entrenamiento
testNoMates = df4.tail(3000).reset_index(drop=True)
trainNoMates = df4.iloc[:-3000].reset_index(drop=True)

#Resultado: Archivo de train y de test
trainJaques = pd.concat([trainJaques, trainNoMates], ignore_index=True)
trainJaques = trainJaques.reset_index(drop=True)

testJaques = pd.concat([testJaques, testNoMates], ignore_index=True)
testJaques = testJaques.reset_index(drop=True)

trainJaques.to_json('trainJaques.json')
testJaques.to_json('testJaques.json')
```

Figure 109: Menacho, E.

Y, por último, creamos otro archivo para crear los mapas con la posición de las piezas.

```
import pandas as pd

def printMat(matrix):
    for m in matrix:
        print(m)

def peones_blancos(matrix):
    # create a new matrix with the same dimensions as the input matrix
    result = [[0 for _ in range(len(matrix[0]))] for _ in range(len(matrix))]

    # loop through each element of the matrix and set the corresponding element
    # in the result matrix to 1 if it's not zero
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if matrix[i][j] == 1:
                result[i][j] = 1

    # return the result matrix
    return result

def peones_negros(matrix):
    # create a new matrix with the same dimensions as the input matrix
    result = [[0 for _ in range(len(matrix[0]))] for _ in range(len(matrix))]

    # loop through each element of the matrix and set the corresponding
    # element in the result matrix to 1 if it's not zero
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if matrix[i][j] == -1:
                result[i][j] = -1

    # return the result matrix
    return result

def caballos_blancos(matrix):
    # create a new matrix with the same dimensions as the input matrix
    result = [[0 for _ in range(len(matrix[0]))] for _ in range(len(matrix))]

    # loop through each element of the matrix and set the corresponding
    # element in the result matrix to 1 if it's not zero
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if matrix[i][j] == 2:
                result[i][j] = 2

    # return the result matrix
    return result
```

Figure 110: Menacho, E.

```

def caballos_negros(matrix):
    # create a new matrix with the same dimensions as the input matrix
    result = [[0 for _ in range(len(matrix[0]))] for _ in range(len(matrix))]

    # loop through each element of the matrix and set the corresponding
    # element in the result matrix to 1 if it's not zero
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if matrix[i][j] == -2:
                result[i][j] = -2

    # return the result matrix
    return result


def alfiles_blancos(matrix):
    # create a new matrix with the same dimensions as the input matrix
    result = [[0 for _ in range(len(matrix[0]))] for _ in range(len(matrix))]

    # loop through each element of the matrix and set the corresponding
    # element in the result matrix to 1 if it's not zero
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if matrix[i][j] == 3:
                result[i][j] = 3

    # return the result matrix
    return result


def alfiles_negros(matrix):
    # create a new matrix with the same dimensions as the input matrix
    result = [[0 for _ in range(len(matrix[0]))] for _ in range(len(matrix))]

    # loop through each element of the matrix and set the corresponding
    # element in the result matrix to 1 if it's not zero
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if matrix[i][j] == -3:
                result[i][j] = -3

    # return the result matrix
    return result


def torres_blancos(matrix):
    # create a new matrix with the same dimensions as the input matrix
    result = [[0 for _ in range(len(matrix[0]))] for _ in range(len(matrix))]

    # loop through each element of the matrix and set the corresponding
    # element in the result matrix to 1 if it's not zero
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if matrix[i][j] == 4:
                result[i][j] = 4

```

Figure 111: Menacho, E.

```

def torres_negros(matrix):
    # create a new matrix with the same dimensions as the input matrix
    result = [[0 for _ in range(len(matrix[0]))] for _ in range(len(matrix))]

    # loop through each element of the matrix and set the corresponding
    # element in the result matrix to 1 if it's not zero
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if matrix[i][j] == -4:
                result[i][j] = -4

    # return the result matrix
    return result


def reina_blanacos(matrix):
    # create a new matrix with the same dimensions as the input matrix
    result = [[0 for _ in range(len(matrix[0]))] for _ in range(len(matrix))]

    # loop through each element of the matrix and set the corresponding
    # element in the result matrix to 1 if it's not zero
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if matrix[i][j] == 5:
                result[i][j] = 5

    # return the result matrix
    return result


def reina_negros(matrix):
    # create a new matrix with the same dimensions as the input matrix
    result = [[0 for _ in range(len(matrix[0]))] for _ in range(len(matrix))]

    # loop through each element of the matrix and set the corresponding
    # element in the result matrix to 1 if it's not zero
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if matrix[i][j] == -5:
                result[i][j] = -5

    # return the result matrix
    return result


def rey_blanacos(matrix):
    # create a new matrix with the same dimensions as the input matrix
    result = [[0 for _ in range(len(matrix[0]))] for _ in range(len(matrix))]

    # loop through each element of the matrix and set the corresponding
    # element in the result matrix to 1 if it's not zero
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if matrix[i][j] == 6:
                result[i][j] = 6

```

Figure 112: Menacho, E.

```
def rey_negros(matrix):
    # create a new matrix with the same dimensions as the input matrix
    result = [[0 for _ in range(len(matrix[0]))] for _ in range(len(matrix))]

    # loop through each element of the matrix and set the corresponding
    # element in the result matrix to 1 if it's not zero
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if matrix[i][j] == -6:
                result[i][j] = -6

    # return the result matrix
    return result
```

Figure 113: Menacho, E.

Creamos estas funciones para poder crear los mapas a partir de los ficheros previamente creados. Finalmente ejecutamos las funciones sobre los ficheros para crear nuestro archivo.

```

train = pd.read_json('trainJaques.json')

train['peones_blancos'] = train["posiciones"].apply(peones_blanco)
train['peones_negros'] = train["posiciones"].apply(peones_negro)

train['caballos_blanco'] = train["posiciones"].apply(caballos_blanco)
train['caballos_negro'] = train["posiciones"].apply(caballos_negro)

train['alfiles_blanco'] = train["posiciones"].apply(alfiles_blanco)
train['alfiles_negro'] = train["posiciones"].apply(alfiles_negro)

train['torres_blanco'] = train["posiciones"].apply(torres_blanco)
train['torres_negro'] = train["posiciones"].apply(torres_negro)

train['reina_blanco'] = train["posiciones"].apply(reina_blanco)
train['reina_negro'] = train["posiciones"].apply(reina_negro)

train['rey_blanco'] = train["posiciones"].apply(rey_blanco)
train['rey_negro'] = train["posiciones"].apply(rey_negro)

test = pd.read_json('testJaques.json')

test['peones_blanco'] = test["posiciones"].apply(peones_blanco)
test['peones_negro'] = test["posiciones"].apply(peones_negro)

test['caballos_blanco'] = test["posiciones"].apply(caballos_blanco)
test['caballos_negro'] = test["posiciones"].apply(caballos_negro)

test['alfiles_blanco'] = test["posiciones"].apply(alfiles_blanco)
test['alfiles_negro'] = test["posiciones"].apply(alfiles_negro)

test['torres_blanco'] = test["posiciones"].apply(torres_blanco)
test['torres_negro'] = test["posiciones"].apply(torres_negro)

test['reina_blanco'] = test["posiciones"].apply(reina_blanco)
test['reina_negro'] = test["posiciones"].apply(reina_negro)

test['rey_blanco'] = test["posiciones"].apply(rey_blanco)
test['rey_negro'] = test["posiciones"].apply(rey_negro)

train.to_json('trainJaquesMap.json')
test.to_json('testJaquesMap.json')

```

Figure 114: Menacho, E.

Ahora que tenemos todos los datos para entrenar la red, lo siguiente será entrenarla para después hacer las predicciones, ya que el proceso de entrenar una red convolucional es algo muy costoso utilizaremos de nuevo Google Colaboratory para que nos permita entrenar nuestro modelo con las gráficas de sus servidores, así podemos ahorrar muchísimo tiempo.

Para esto subimos los dos archivos creados anteriormente en una carpeta de Drive que contenga nuestro cuaderno de Colaboratory. Importamos las siguientes librerías:

```
[ ] #Para cargar y operar con el dataset
    import pandas as pd
    import numpy as np

    #Para entrenar y crear el modelo de la red convolucional
    import tensorflow as tf
    from tensorflow.keras.layers import Dropout
    from tensorflow.keras import layers, models
    from keras.callbacks import EarlyStopping

    #Para imprimir el modelo
    import matplotlib.pyplot as plt
```

Figure 114: Menacho, E.

Conectamos nuestro cuaderno con Google Drive.

```
[ ] print(tf.__version__)
    □ 2.11.0

[ ] from google.colab import drive
    drive.mount('/content/drive')

Drive already mounted at /content/drive
```

Figure 115: Menacho, E.

Importamos los ficheros usando pandas.

```
▶ #Importamos los ficheros con el dataset de las posiciones y si es jaque
train = pd.read_json('drive/MyDrive/Colab Notebooks/Alphaupo-Convolucionales/trainJaquesMap.json')
test = pd.read_json('drive/MyDrive/Colab Notebooks/Alphaupo-Convolucionales/testJaquesMap.json')
```

Figure 116: Menacho, E.

Comprobamos que todo esté en orden:

```
▶ train.head()
```

	posiciones	y	peones_blanco	peones_negro	caballos_blanco	...
0	<code>[[[-4, 0, -3,</code> <code>-5, -6, -3, 0,</code> <code>-4], [0, 0, -1,</code> <code>-1, ...</code>	<code>1</code>	<code>[[0, 0, 0, 0, 0, 0,</code> <code>[0, 0], [0, 0, 0, 0,</code> <code>0, 0, ...</code>	<code>[[0, 0, 0, 0, 0, 0,</code> <code>[0, 0], [0, 0, -1,</code> <code>-1, 0, ...</code>	<code>[[0, 0, 0, 0, 0, 0,</code> <code>[0], [0, 0, 0, 0, 0,</code> <code>...</code>	<code>...</code>
1	<code>[[[-4, 0, -3, 0,</code> <code>-6, -3, 0, -4],</code> <code>[0, 0, -1,</code> <code>-1, ...</code>	<code>1</code>	<code>[[0, 0, 0, 0, 0, 0,</code> <code>[0, 0], [0, 0, 0, 0,</code> <code>0, 0, ...</code>	<code>[[0, 0, 0, 0, 0, 0,</code> <code>[0, 0], [0, 0, -1,</code> <code>-1, 0, ...</code>	<code>[[0, 0, 0, 0, 0, 0,</code> <code>[0], [0, 0, 0, 0, 0,</code> <code>...</code>	<code>...</code>
2	<code>[[0, -4, -3, 0,</code> <code>0, -3, 0, -4],</code> <code>[-1, 0, 0, -1,</code> <code>...,</code>	<code>1</code>	<code>[[0, 0, 0, 0, 0, 0,</code> <code>[0, 0], [0, 0, 0, 0,</code> <code>1, 0, ...</code>	<code>[[0, 0, 0, 0, 0, 0,</code> <code>[0, 0], [-1, 0, 0,</code> <code>-1, 0, ...</code>	<code>[[0, 0, 0, 0, 0, 0,</code> <code>[0], [0, 0, 0, 0, 0,</code> <code>...</code>	<code>...</code>
3	<code>[[[-4, -2, -3,</code> <code>0, 0, -6, 0,</code> <code>-4], [-1, -1,</code> <code>-1, ...</code>	<code>1</code>	<code>[[0, 0, 0, 0, 0, 0,</code> <code>[0, 0], [0, 0, 0, 0,</code> <code>0, 0, ...</code>	<code>[[0, 0, 0, 0, 0, 0,</code> <code>[0, 0], [-1, -1, -1,</code> <code>-1, 0, ...</code>	<code>[[0, 0, 0, 0, 0, 0,</code> <code>[0], [0, 0, 0, 0, 0,</code> <code>...</code>	<code>...</code>
4	<code>[[[-4, 0, 0, 0,</code> <code>0, -3, 0, -4],</code> <code>[-1, 0, 0, 0,</code> <code>-1, ...</code>	<code>1</code>	<code>[[0, 0, 0, 0, 0, 0,</code> <code>[0, 0], [0, 0, 0, 0,</code> <code>0, 0, ...</code>	<code>[[0, 0, 0, 0, 0, 0,</code> <code>[0, 0], [-1, 0, 0,</code> <code>0, -1, 0, ...</code>	<code>[[0, 0, 0, 0, 0, 0,</code> <code>[0], [0, 0, 0, 0, 0,</code> <code>...</code>	<code>...</code>

Figure 117: Menacho, E.

Separamos la X y la Y de los datasets para entrenar y testeo.

```
▶ #Separamos los valores del output (y) y input(x)
trainX = tf.constant(train.drop("y",axis=1).values.tolist())
testX = tf.constant(test.drop("y",axis=1).values.tolist())

trainY = tf.constant(train["y"].values)
testY = tf.constant(test["y"].values)
```

Figure 120: Menacho, E.

Creamos el modelo de la red convolucional usando TensorFlow:

```
▶ #Creamos el modelo de la red convolucional

model = models.Sequential()
model.add(layers.Conv2D(64, (2, 2), activation='relu', input_shape=(13, 8, 8)))
model.add(layers.MaxPooling2D((1, 1)))
model.add(Dropout(0.5))
model.add(layers.Conv2D(128, (2, 2), activation='relu'))
model.add(layers.MaxPooling2D((1, 1)))
model.add(Dropout(0.5))
model.add(layers.Conv2D(256, (2, 2), activation='relu'))
model.add(layers.MaxPooling2D((1, 1)))
model.add(Dropout(0.5))
model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(2))
```

Figure 119: Menacho, E.

Este modelo de TensorFlow es una red neuronal convolucional para clasificación binaria que utiliza una arquitectura secuencial. La red comienza con una capa de convolución de 64 filtros de 2x2 y función de activación ReLU, seguida de una capa de MaxPooling de 1x1 y una capa de Dropout con una tasa de abandono del 50%. Luego, hay dos capas más de convolución, con 128 y 256 filtros respectivamente, cada una seguida de una capa de MaxPooling de 1x1 y con otro Dropout. Después, hay una capa de aplanamiento y dos capas densas, con 512 unidades y

función de activación ReLU en la primera capa y sin función de activación en la última capa.

Esta última capa tiene dos unidades, ya que es una tarea de clasificación binaria.

```
▶ model.summary()

Model: "sequential_2"
-----

| Layer (type)                    | Output Shape       | Param # |
|---------------------------------|--------------------|---------|
| conv2d_6 (Conv2D)               | (None, 12, 7, 64)  | 2112    |
| max_pooling2d_6 (MaxPooling 2D) | (None, 12, 7, 64)  | 0       |
| dropout_6 (Dropout)             | (None, 12, 7, 64)  | 0       |
| conv2d_7 (Conv2D)               | (None, 11, 6, 128) | 32896   |
| max_pooling2d_7 (MaxPooling 2D) | (None, 11, 6, 128) | 0       |
| dropout_7 (Dropout)             | (None, 11, 6, 128) | 0       |
| conv2d_8 (Conv2D)               | (None, 10, 5, 256) | 131328  |
| max_pooling2d_8 (MaxPooling 2D) | (None, 10, 5, 256) | 0       |
| dropout_8 (Dropout)             | (None, 10, 5, 256) | 0       |
| flatten_2 (Flatten)             | (None, 12800)      | 0       |
| dense_4 (Dense)                 | (None, 512)        | 6554112 |
| dense_5 (Dense)                 | (None, 2)          | 1026    |
| <hr/>                           |                    |         |
| Total params: 6,721,474         |                    |         |
| Trainable params: 6,721,474     |                    |         |
| Non-trainable params: 0         |                    |         |

-----
```

Figure 121: Menacho, E.

Compilamos el modelo.

```
#Compilamos el modelo

model.compile(optimizer=tf.keras.optimizers.Adam(),
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Menacho, E.

Entrenamos el modelo con una condición de parada para cuando se pase 5 épocas sin mejorar el resultado de test, de esta forma evitamos un sobreentreno por parte de nuestra red convolucional.

```
#Entrenamos el modelo

# Define early stopping criteria
early_stop = EarlyStopping(monitor='val_loss', patience=5, verbose=1)

# Train the model with early stopping
history = model.fit(trainX, trainY, epochs=500,
                     validation_data=(testX, testY),
                     callbacks=[early_stop])
```

Figure 122: Menacho, E.

Imprimimos por pantalla el entrenamiento.

```
#Hacemos el plot de la precision de entrenamiento y de test

arr = np.full((len(history.history['accuracy'])), ), 0.9)
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(arr, "--", linewidth=1)
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')
```

Figure 123: Menacho, E.

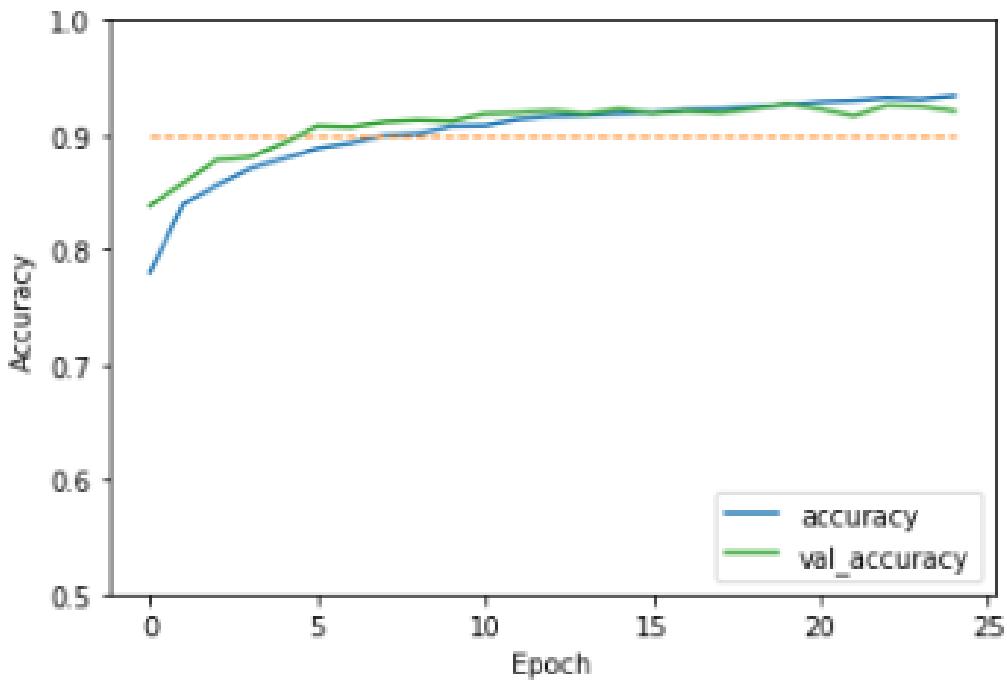
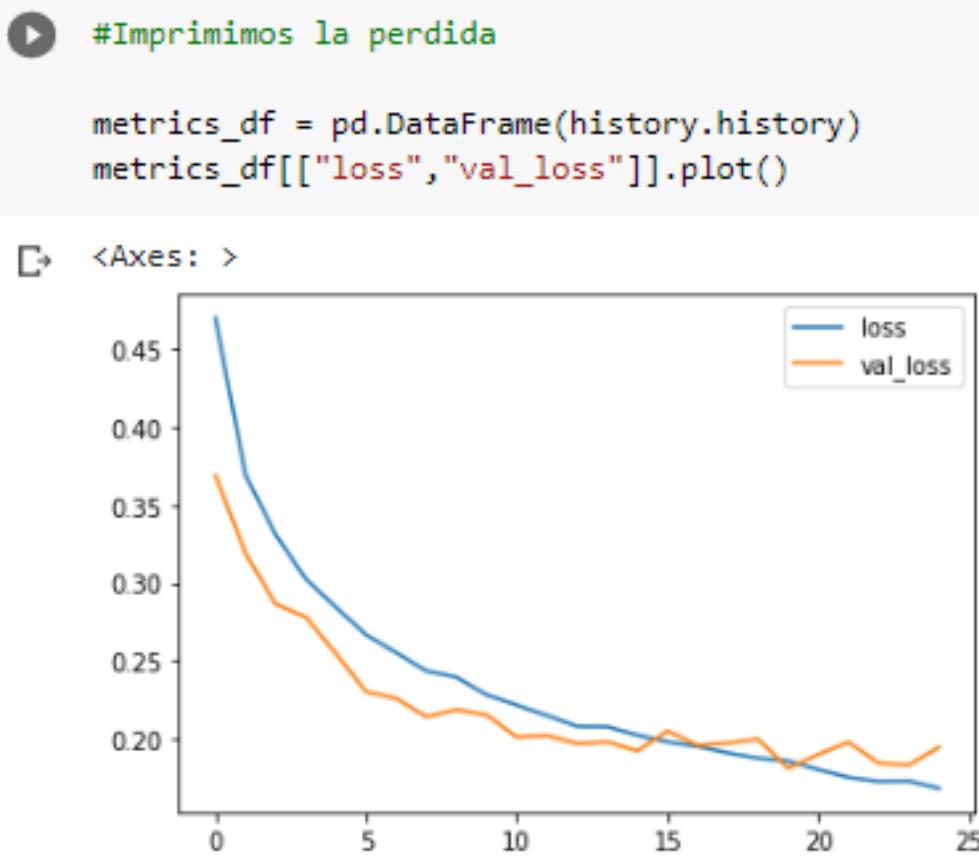


Figure 124: Menacho, E.

Observamos que nuestra red está bien implementada, ya que la precisión aumenta de forma casi similar para ambos casos, tanto para los valores de entrenamiento como para los de validación, esto quiere decir que está entrenando de una forma buena y por lo tanto conseguimos que nuestra red sea capaz de predecir posiciones en las que existan jaques con una precisión bastante alta.



Observamos como nuestro conjunto de test se entrena de la mejor manera posible respecto a los datos de entrada, quizás para conseguir un 100% de acierto debiéramos utilizar otros datos de entrada, pero yo estoy satisfecho con los extraídos en esta red.

Realizamos las predicciones sobre el conjunto de test:

```
▶ #Hacemos las predicciones sobre el conjunto de test

res = model.predict(testX)

print(model.predict(np.array([testX[0]])))
print(res[0])

⇨ 188/188 [=====] - 0s 2ms/step
1/1 [=====] - 0s 76ms/step
[[0.33860248 2.116118 ]]
[0.33860278 2.116119 ]
```

Figure 126: Menacho, E.

Imprimimos la tasa de acierto sobre el conjunto de test:

```
▶ #Imprimimos la tasa de acierto sobre el conjunto de test

a = np.array(testY)
b= np.argmax(res,axis=1)
cont = 0

for i in range(0, len(a)):
    if a[i] == b [i]:
        cont = cont +1

print("Resultado: "+str(cont)+"/"+str(len(a)))
print("Tasa de acierto: "+str((cont/len(a))*100)+"%")

Resultado: 5525/6000
Tasa de acierto: 92.0833333333333%
```

Figure 127: Menacho, E.

Instalamos las librerías para exportar el modelo:

```
▶ !pip install tensorflowjs
import tensorflowjs as tfjs
```

Figure 128: Menacho, E.

Por último, exportamos el modelo a JavaScript y Python.

```
[ ] model.save('drive/MyDrive/Colab Notebooks/Alphaupo-Convolucionales/modeloJaques')
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_cc
• Exportamos el modelo para javascript

[ ] tfjs.converters.save_keras_model(model, "drive/MyDrive/Colab Notebooks/Alphaupo-Convolucionales/ModelosJS/modeloJaques")
```

Figure 129: Menacho, E.

8.2 Redes convolucionales para la clasificación de jaques mate sin conocimiento

Ahora que ya sabemos que podemos entrenar una red para predecir jaques sin conocimientos previos del juego, podemos hacer lo mismo, pero para clasificar si una posición contiene un jaque mate o no. Para ello utilizaremos un dataset de manera idéntica al anterior, solo que esta vez en lugar de usar en y los jaques utilizaremos los jaques mate.

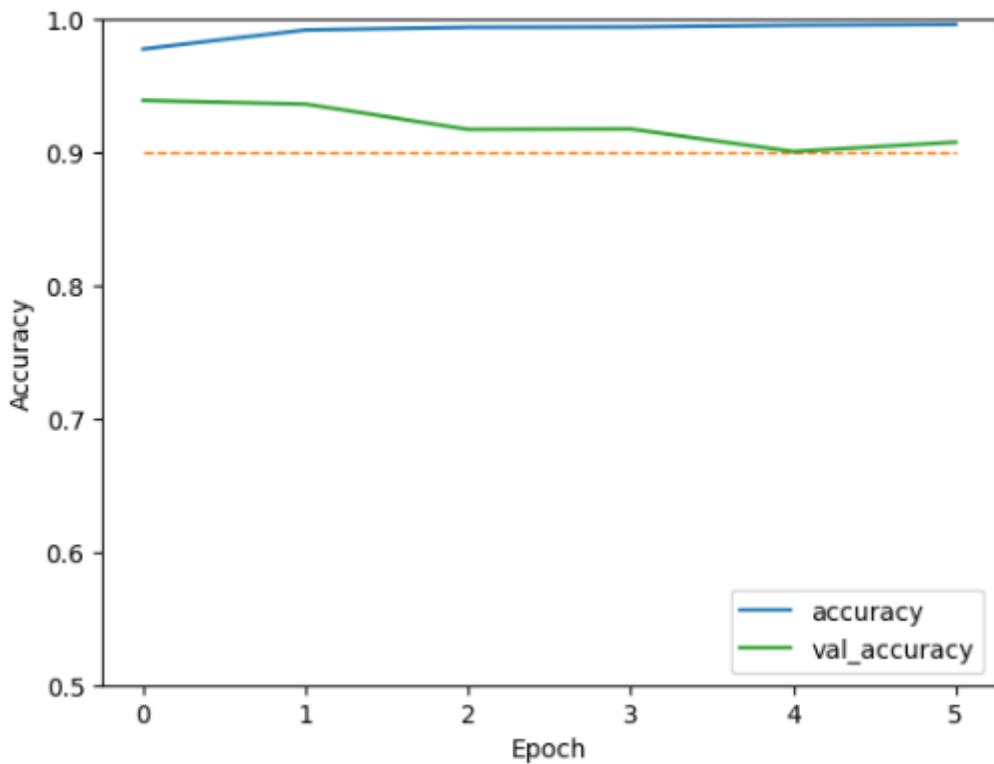


Figure 130: Menacho, E.

En este caso se obtienen resultados bastante acertados a las pocas iteraciones de la red convolucional, ya que estamos utilizando un número menor de filas como entrada de los datos, aun así, se consigue una precisión bastante acertada, aunque la pérdida de nuestro modelo sea alta para el poco número de iteraciones, por eso paramos el entrenamiento.

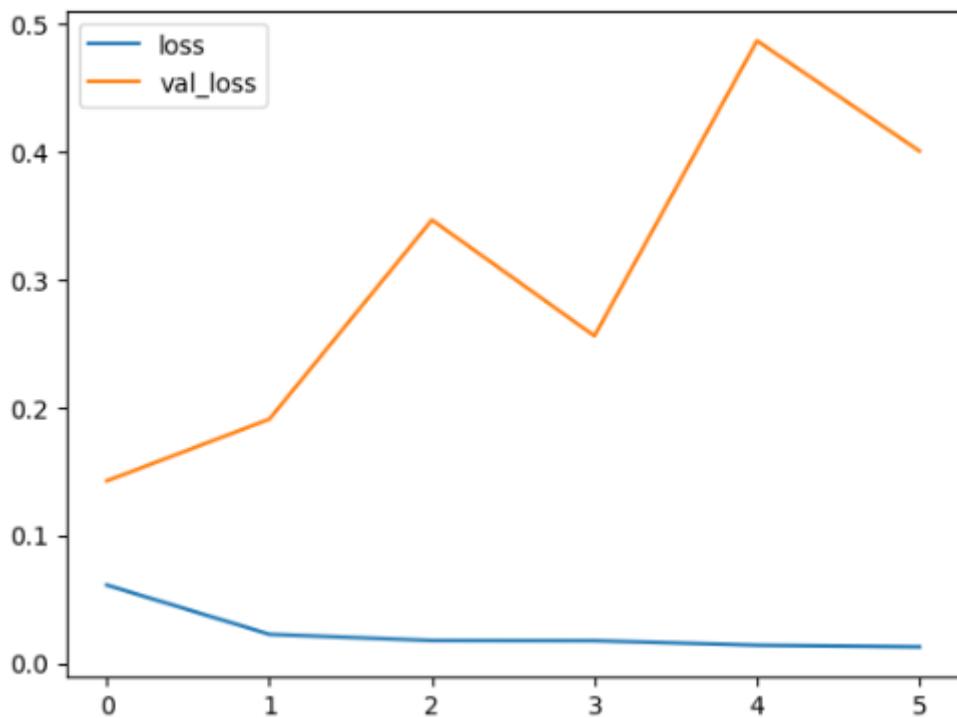


Figure 131: Menacho, E.

Una pérdida bastante alta al comienzo, pero esto se debe a que tenemos que obtener más datos para poder entrenar al modelo de una forma más acertada. Los resultados sobre el conjunto de test son los siguientes:

Resultado: 8578/9450
Tasa de acierto: 90.77248677248677%

Figure 132: Menacho, E.

8.3 Redes convolucionales como función de evaluación

Para utilizar las redes convolucionales como función de evaluación debemos adaptarlas con el fin de que sean capaces de predecir valores en lugar de clasificarlos, luego debemos crear una salida para la entrada de datos que nos permita dar un valor cuantitativo para cada posición de ajedrez. Para ello utilizaremos el motor Stockfish 15, quien nos proporcionará las evaluaciones para cada posición en profundidad 20.

Una vez hemos obtenido nuestro fichero de datos con las evaluaciones del motor, entrenaremos nuestra red con estas partidas, haciendo que se asemeje lo máximo posible a las evaluaciones proporcionadas.

Para ello mapearemos las posiciones de la misma forma que para la predicción de jaques y jaques mate. Por lo tanto, lo que cambiará será la red utilizada y las funciones de optimización para calcular los valores. Este es el modelo que hemos decidido utilizar:

```

Model: "sequential_3"
-----  

Layer (type)          Output Shape       Param #
-----  

conv2d_12 (Conv2D)    (None, 12, 7, 128)   4224  

max_pooling2d_9 (MaxPooling 2D)        (None, 12, 7, 128)   0  

dropout (Dropout)      (None, 12, 7, 128)   0  

conv2d_13 (Conv2D)    (None, 11, 6, 256)    131328  

max_pooling2d_10 (MaxPooling 2D)        (None, 11, 6, 256)   0  

dropout_1 (Dropout)    (None, 11, 6, 256)   0  

conv2d_14 (Conv2D)    (None, 10, 5, 256)    262400  

max_pooling2d_11 (MaxPooling 2D)        (None, 10, 5, 256)   0  

dropout_2 (Dropout)    (None, 10, 5, 256)   0  

conv2d_15 (Conv2D)    (None, 9, 4, 256)    262400  

flatten_3 (Flatten)   (None, 9216)         0  

dense_6 (Dense)       (None, 256)          2359552  

dense_7 (Dense)       (None, 1)            257
-----  

Total params: 3,020,161  

Trainable params: 3,020,161  

Non-trainable params: 0
-----
```

Figure 133: Menacho, E.

El optimizador utilizado, en este caso RMSprop:

```

optimizer = tf.keras.optimizers.RMSprop(0.0001)

"""
history = modelo.compile(optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy'])
"""

model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

Figure 134: Menacho, E.

Estos son los resultados del entrenamiento de la red por unas 30 épocas:

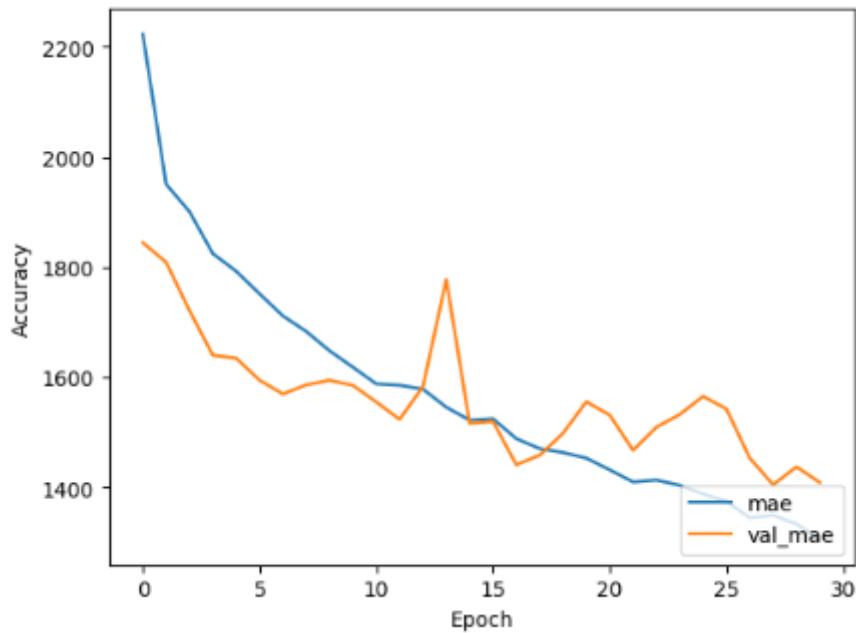


Figure 135: Menacho, E.

Observamos que nuestro modelo aprende de manera efectiva para las pocas posiciones proporcionadas.

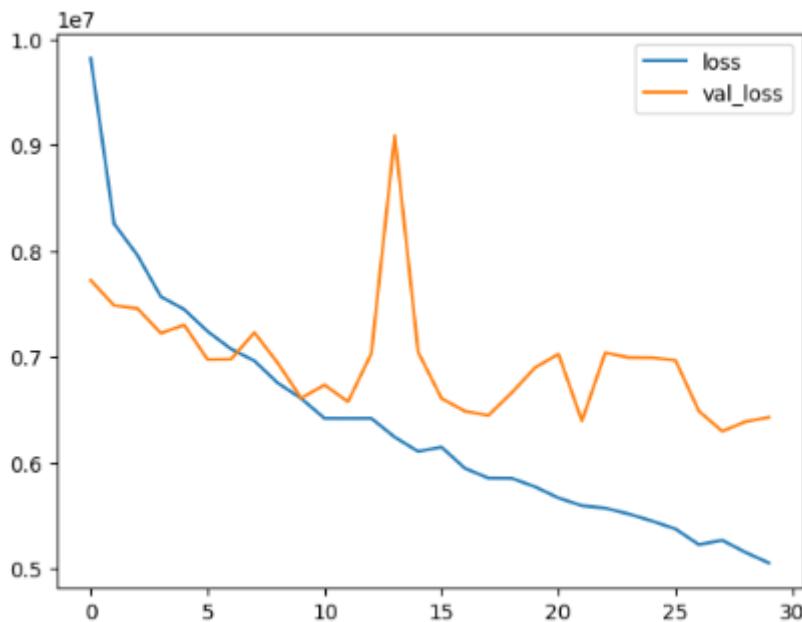


Figure 136: Menacho, E.

La pérdida no aumenta por lo general exceptuando algunos picos que ocurren al intentar predecir y no clasificar. La pérdida al aumentar el número de épocas se hace mayor, esto es debido al sobre entrenamiento de la red.

Con la implementación de este modelo estamos probando el aprendizaje reforzado de un motor de ajedrez, ya que podemos conseguir con el modelo el valor heurístico de la posición. Por lo tanto, podríamos plantear como modelo de aprendizaje reforzado el siguiente.



Figure 137: Menacho, E.

Donde jugamos una partida de ajedrez, después un motor de ajedrez evalúa las posiciones de la partida, para después entrenar a la red convolucional con las nuevas evaluaciones. De esta forma estaríamos creando un modelo basado en aprendizaje reforzado. Para ejecutarlo podremos usar cualquier tipo de búsqueda de árbol.

Para ejecutar los cuadernos implementados durante la investigación acceder al siguiente enlace:

- <https://github.com/nenomg/ALPHAUPO-INVESTIGATION>

9. CONCLUSIONES

Tras realizar esta investigación y profundizar más en la problemática desde el punto de vista informático podemos afirmar que es un juego que todavía no ha sido posible resolverlo por un ordenador. El número de posibilidades a calcular es absolutamente abrumador, un árbol de búsqueda tan enorme que es imposible calcular hoy en día.

Pero, ¿Tan importante es explorar el árbol de búsqueda al completo para resolver el juego? Para responder a esta pregunta debemos ir atrás en el tiempo por la época de Paul Morphy, donde este era implacable en su juego, ganaba la gran mayoría de sus partidas. ¿Por qué? ¿Era capaz de pensar más profundo en el espacio de búsqueda? La respuesta a esto la encontramos cuando enfrentamos al algoritmo de Minimax con Turochamp contra la implementación sin Turochamp. Ambos exploran el árbol de búsqueda en la misma profundidad, sin embargo, el que tenía proporcionado una heurística fue capaz de hacerse con la ventaja y conseguir la victoria. Luego, a lo que quiero hacer alusión es que Paul Morphy es un humano como todos nosotros, simplemente tenía una visión diferente del juego y esto lo hacía un adelantado para su época. Lo mismo ocurre con los algoritmos, una perspectiva distinta como la ponderación Alpha Beta es mucho más eficiente y por lo tanto mejor que la implementación sin ponderar.

Esto lo vemos a lo largo de la historia de los algoritmos que hemos visto a lo largo de la historia. El Deep Blue fue sobrepasado por otros motores como Stockfish, solo por cuestión de perspectiva. Al igual que el anterior, a Stockfish 8 lo superó Alphazero con el aprendizaje reforzado y este ha sido superado por la última implementación de Stockfish NNUE.

Así pues, se demuestra que, si es imposible calcular el juego al completo, tenemos un problema de perspectiva, es decir, la persona o algoritmo que posea una mayor comprensión del juego será la que saldrá victoriosa en un enfrentamiento.

El campo de búsqueda es demasiado amplio y es un problema incluso para los ordenadores de hoy en día, investigar en este campo no solo podría traer beneficios para el juego, sino que

también le extiende la mano indirectamente a la informática, como lo han hecho el algoritmo del Deep Blue o el aprendizaje reforzado de Alphazero. ¡Quién sabe qué será lo siguiente que vendrá! Pero lo que sí tengo claro es que se trata de un juego de comprensión, en el que gana quién comprenda más sobre este enigmático juego.

ANEXO I – PLAN DE PROYECTO

PLAN DE PROYECTO

“ALPHAUPO”

Edición: 1

Fecha: 16-05-2023

CONTROL Y REGISTRO DE CAMBIO DEL DOCUMENTO

CONTROL	
Proyecto	ALPHAUPO WEBSITE
Denominación	Plan de Proyecto del Proyecto ALPHAUPO WEBSITE
Fecha	día de mes de año
Edición	1
Grupo	ALPHAUPO
Autores	Eugenio Menacho de Góngora

REGISTRO DE CAMBIOS		
VERSIÓN	DESCRIPCIÓN DEL CAMBIO	FECHA DEL CAMBIO
1	Añadida Introducción y objetivos del proyecto.	5/03/2023
2	Añadidas metodologías de gestión y la organización del proyecto.	7/03/2023
3	Añadido el programa del trabajo.	18/04/2023
4	Finalización del documento.	10/05/2023

1. INTRODUCCIÓN

Este documento es un plan de desarrollo detallado para el proyecto de "Alphaupo". La plataforma se diseñará como una página web interactiva y contará con una base de datos que almacenará partidas de grandes maestros, una implementación del juego, diversas implementaciones de motores de ajedrez, un sistema de gestión completo para los usuarios y las partidas de maestros y una página a modo de expositor sobre la investigación a realizar.

La aplicación dispondrá de 2 roles, el rol de administrador y el rol de usuario.

Los administradores podrán:

- Acceder al sistema de gestión de usuarios y partidas.
- Acceder a la base de datos.
- Acceder a jugar con los motores de ajedrez.
- Acceder a la investigación.

Los usuarios podrán:

- Acceder a la base de datos.
- Acceder a jugar con los motores de ajedrez.
- Acceder a la investigación.

Los no usuarios podrán:

- Acceder a la investigación.

2. OBJETIVOS DEL PROYECTO

El objetivo principal del proyecto es proporcionar a los usuarios una plataforma de ajedrez en línea completa que permita a los usuarios jugar contra diversas implementaciones del juego.

También queremos proveer a los usuarios con una base de datos intuitiva que permita analizar las partidas, un sistema de gestión para los administradores y un apartado de expositor para la investigación.

Para ello, el equipo de desarrollo se enfocará en diseñar una interfaz de usuario intuitiva, accesible y fácil de usar que pueda adaptarse a una amplia gama de dispositivos y plataformas.

Por lo tanto, identificamos los siguientes objetivos de alto nivel para el desarrollo de la aplicación:

OBJ–1	Investigación acerca de la informática en el ajedrez
Versión	1
Autores	Eugenio Menacho de Góngora
Fuente	Investigación sobre la informática en el ajedrez
Descripción	El objetivo principal del proyecto es entender y comprender más acerca del juego, para ello lo más importante será realizar una investigación sobre como a evolucionado la informática y cómo podríamos adaptarla para implementarla en el ajedrez.
Importancia	Alta
Estado	Aprobado
Comentarios	

OBJ–2	Motor de ajedrez
Versión	1
Autores	Eugenio Menacho de Góngora
Fuente	Investigación sobre la informática en el ajedrez
Descripción	El sistema deberá implementar varios algoritmos de ajedrez y contar con un motor que permita a los usuarios jugar partidas de ajedrez en línea contra la computadora, ofreciendo diferentes niveles de dificultad para adaptarse a las habilidades de cada usuario. Además, el motor de ajedrez deberá ser capaz de realizar movimientos válidos y legales, asegurando una experiencia de juego auténtica y desafiante para los usuarios.
Importancia	Alta
Estado	Aprobado
Comentarios	Se deberá implementar una interfaz que permita jugar al juego.

OBJ-3	Base de datos
Versión	1
Autores	Eugenio Menacho de Góngora
Fuente	Investigación sobre la informática en el ajedrez
Descripción	El sistema debe contar con una base de datos de ajedrez que permita a los usuarios analizar partidas, incluyendo la posibilidad de realizar búsquedas avanzadas por diferentes criterios, tales como jugadores y eventos. Además, la base de datos deberá ser actualizable y permitir la inserción de nuevas partidas y la eliminación de partidas obsoletas por parte de los administradores.
Importancia	Alta
Estado	Aprobado
Comentarios	Inclusión de un modelo de inteligencia artificial implementado anteriormente.

OBJ-4	Sistema de gestión de usuarios y partidas
Versión	1
Autores	Eugenio Menacho de Góngora
Fuente	Investigación sobre la informática en el ajedrez
Descripción	El sistema deberá contar con un sistema de registro de usuarios que permita la creación de perfiles y la gestión de datos personales y partidas jugadas. Permitiendo la modificación y eliminación para todas estas entidades.
Importancia	Alta
Estado	Aprobado
Comentarios	

3. ORGANIZACIÓN DEL PROYECTO

3.1 Diagrama de Organización del proyecto

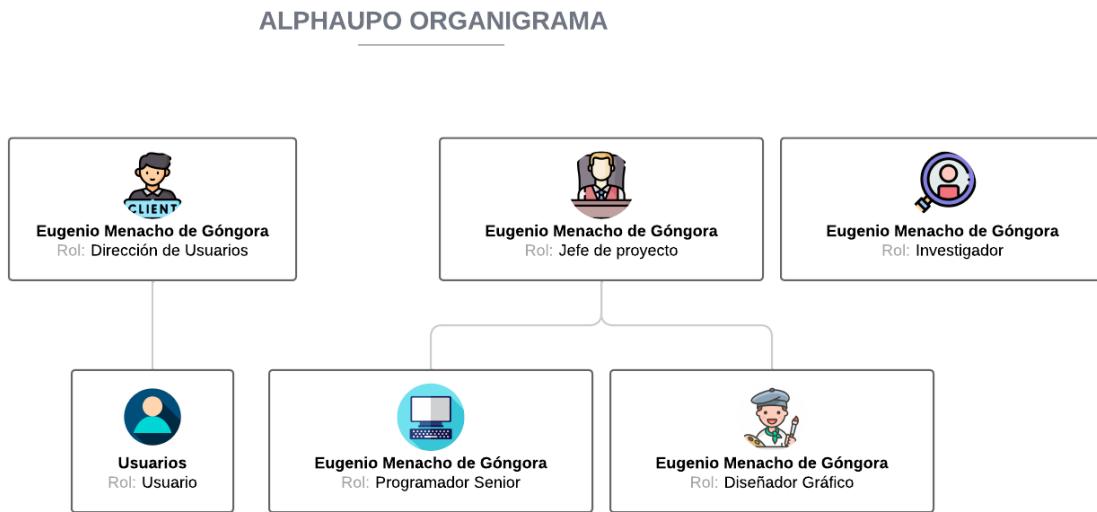


Figure 138: Menacho, E.

- **Jefe de proyecto:** Se encarga de la organización del proyecto, de verificar que los procesos son implementados correctamente por el diseñador gráfico y el programador Senior. También se encargará de tener relación con la dirección de usuarios y de hablar con el investigador tras cumplirse los hitos.
- **Dirección de usuarios:** Se encarga de tener contacto con los clientes para comprobar que se cumplen los requisitos de los interesados.
- **Programador Senior:** Se encarga del desarrollo funcional de la aplicación, el desarrollo del tablero de ajedrez, motor y todas las tareas correspondientes con las entidades y la base de datos.
- **Diseñador gráfico:** Debe estar en contacto constantemente con el programador senior, se encarga única y exclusivamente del diseño de la página, deberá ponerse de acuerdo con el Programador Senior para comenzar cualquier diseño.

- Investigador: Se encargará de investigar acerca de la informática en el ajedrez y de comunicar los datos extraídos al jefe de proyecto.

3.2 Identificación de los interesados

- Jugadores de ajedrez: Tanto personas que acaban de iniciar en el juego como jugadores más avanzados pueden estudiar y aprender de los mejores jugadores del mundo, a través de la base de datos de partidas de maestros y la opción de jugar contra un motor.
- Aficionados del ajedrez: Existen numerosas personas que consideran el ajedrez como una actividad recreativa y mentalmente desafiante. Para estos entusiastas, la opción de enfrentarse a un motor y la posibilidad de examinar y analizar partidas de ajedrez puede resultar atractiva.
- Jugadores profesionales: La base de datos de partidas de ajedrez de alto nivel disponible en la página web puede ser de gran utilidad para los jugadores y entrenadores interesados en mejorar su habilidad en el juego.
- Aficionados a la tecnología: Los amantes de la tecnología se pueden ver interesados en la página web al brindarles la oportunidad de analizar la tecnología detrás del juego y cómo fue implementada en la plataforma. La complejidad del motor de ajedrez también puede resultar fascinante para estos individuos, ya que pueden explorar cómo se desarrolló el motor y cómo funciona para proporcionar un desafío satisfactorio al jugador.
- Investigadores académicos: Los investigadores académicos que tienen interés en el ajedrez y la inteligencia artificial pueden encontrar de gran utilidad la tecnología y los datos disponibles en la página web. Estos recursos pueden ser utilizados para llevar a cabo investigaciones y estudios relacionados con el ajedrez y la IA, y para explorar los

diferentes enfoques que pueden ser aplicados para mejorar el juego y la eficiencia del motor de ajedrez.

3.3 Responsabilidades y funciones de los interesados

Los jugadores de ajedrez y amantes de la tecnología serían nuestros principales interesados, ya que les brindamos la combinación de un software y el mismo juego. Aunque el enfoque de la página web está orientado a cualquier público, no sólo a los interesados mencionados anteriormente, el objetivo principal es brindar información al público acerca de la informática en este juego.

Luego, los usuarios de la aplicación serán los interesados y aquellas personas que quieran entrar en la web.

Los administradores podrán modificar la página cuando se desee, con la finalidad de supervisar y gestionar los datos de la aplicación.

El jefe de proyecto se encarga de la supervisión del programador senior y el diseñador gráfico, al mismo tiempo este se encarga de establecer contacto con la dirección de los usuarios.

El programador senior y el diseñador gráfico serán los que lleven a cabo el desarrollo de la página web y llevarán a cabo la implementación al completo.

4. METODOLOGÍA DE GESTIÓN DEL PROYECTO

La metodología Agile es una forma de abordar el desarrollo de proyectos que se centra en la flexibilidad, la entrega temprana de resultados y la colaboración continua. Para una investigación de informática en ajedrez y una página web a modo de expositor, la metodología Agile puede ser beneficiosa porque permite adaptarse a los cambios a medida que surgen, entregar resultados valiosos de manera frecuente, fomentar la colaboración y mantener al equipo motivado al ver el progreso del proyecto en iteraciones continuas. En resumen, la metodología Agile es una forma efectiva de abordar proyectos complejos y en constante evolución.

Opté por la metodología Agile porque creo que es una forma efectiva de abordar proyectos complejos e innovadores. Esta metodología se centra en la entrega temprana y frecuente de resultados con valor para el cliente o usuario final, lo que permite obtener retroalimentación rápida y hacer cambios en el proyecto en consecuencia. Además, la metodología Agile fomenta la colaboración y la comunicación continua entre los miembros del equipo y los usuarios del proyecto, lo que puede ayudar a asegurar que los resultados se presenten de manera clara y útil. En el caso de mi proyecto de investigación en informática en ajedrez, la metodología Agile es especialmente adecuada debido a la naturaleza cambiante de la investigación. La metodología me permitirá realizar ajustes continuos al enfoque de los resultados en la página web.

Además, la metodología Agile también se adapta bien a mi estilo de trabajo y mi equipo de investigación.

En resumen, opté por la metodología Agile porque creo que es una forma efectiva de abordar proyectos innovadores y complejos, y es especialmente adecuada para mi investigación en informática en el debido a la naturaleza cambiante de la investigación y las necesidades del usuario.



Figure 139: Agile Methodology

5. PROGRAMA DE TRABAJO

5.1 Alcance y objetivos del programa de trabajo

El objetivo principal del programa de trabajo es gestionar la investigación para poder coordinarla con el desarrollo de la página web. Para ello se ha elaborado un plan de tareas para la página web y la investigación, de forma que se pueda trabajar en ambas partes al mismo tiempo.

A continuación, se muestran las tareas que se deberán desarrollar a lo largo del proyecto.

5.2 Plan de tareas

PLAN DE TAREAS ALPHAUPO

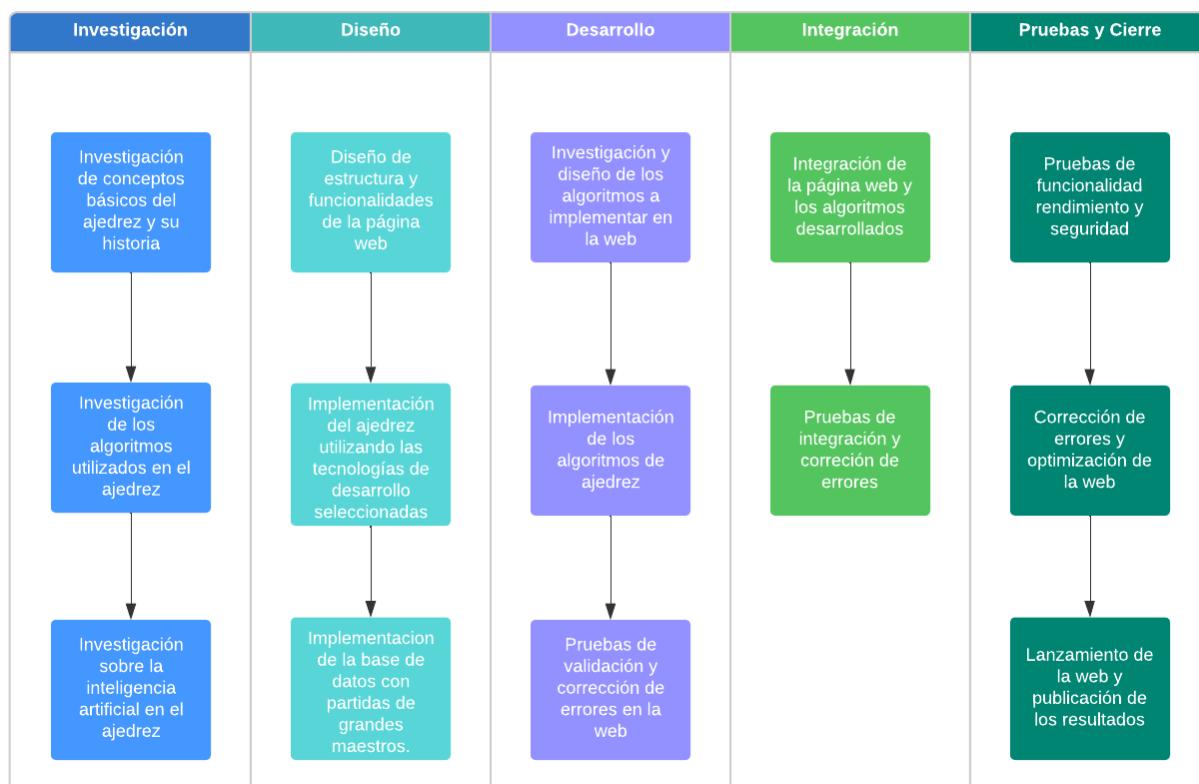


Figure 140: Menacho, E.

También mostramos la planificación del proyecto junto con un diagrama de Gantt, para ello hemos utilizado la herramienta de Microsoft Project:

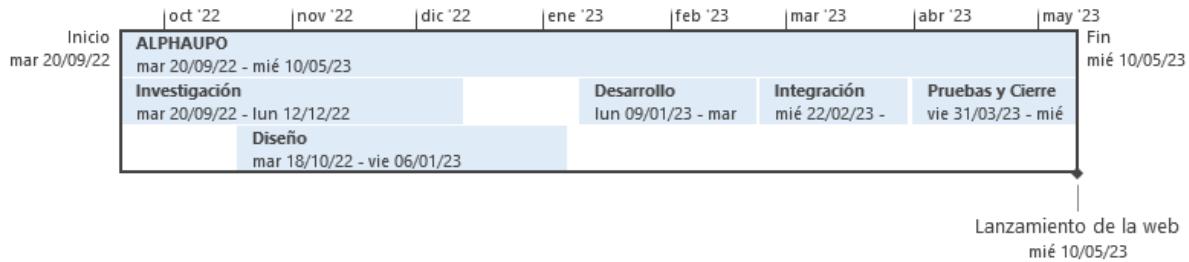


Figure 141: Menacho, E.

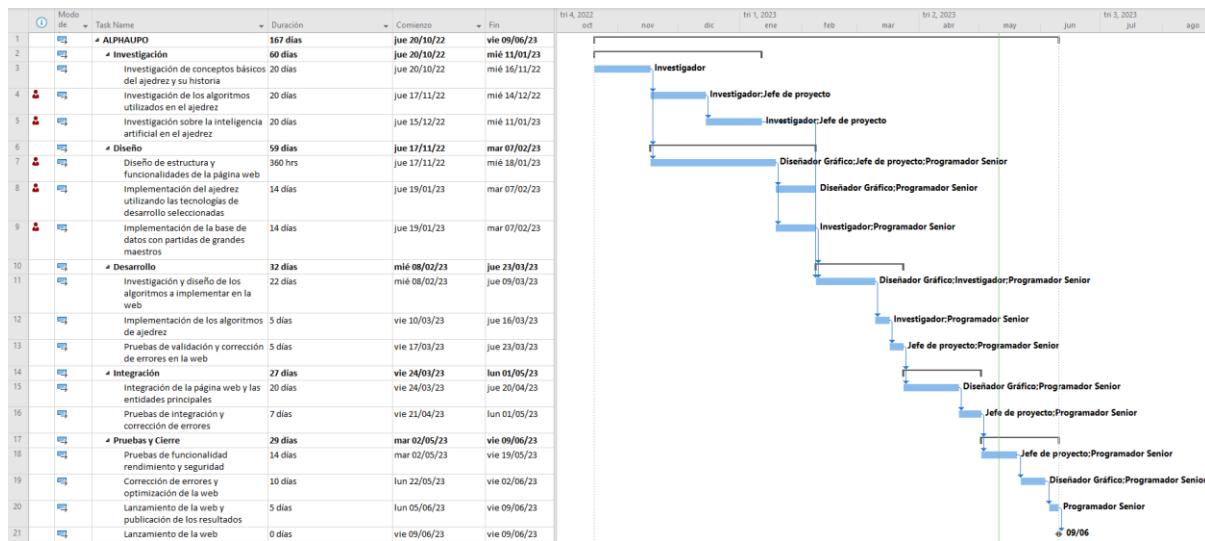


Figure 142: Menacho, E.

Para los recursos utilizaremos las siguientes abreviaturas:

- Jefe de proyecto: JP
- Diseñador gráfico: DG
- Investigador: I
- Programador senior: PS

En lo que respecta a cada tarea, se detallan a continuación, incorporando la responsabilidad de llevarla a cabo y la estimación de cada una de ellas, teniendo en cuenta la distribución de las tareas por fases:

5.2.1 Investigación

Código	Nombre	Responsable	Estimación (días)
1.1.1	Investigación de conceptos básicos del ajedrez y su historia	I	20
Descripción			
<ul style="list-style-type: none"> Se realizará una investigación acerca de la informática en el ajedrez, con el objetivo de aprender más acerca de las soluciones previamente implementadas. <p>El objetivo de la tarea de investigación de conceptos básicos del ajedrez y su historia es proporcionar una comprensión general de los principios y fundamentos del juego de ajedrez, así como su evolución y desarrollo a lo largo de la historia.</p> <p>En resumen, esta tarea tiene como objetivo proporcionar una visión general de los aspectos esenciales del ajedrez, lo que puede ser de gran utilidad para cualquier persona interesada en el juego.</p>			
Observaciones			
N/A			

Código	Nombre	Responsable	Estimación (días)
1.1.2	Investigación de los algoritmos utilizados en el ajedrez	I, JP	20
Descripción			
<ul style="list-style-type: none"> El objetivo principal de esta tarea es comprender cómo los métodos de cálculo y estrategia han evolucionado y se han mejorado con el tiempo para permitir una mejor toma de decisiones en el juego de ajedrez. 			
Observaciones			
N/A			

Código	Nombre	Responsable	Estimación (días)
1.1.3	Investigación sobre la inteligencia artificial en el ajedrez	I, JP	20
Descripción			
<ul style="list-style-type: none"> En esta tarea tratamos de comprender cómo la tecnología ha mejorado la capacidad de la máquina para jugar al ajedrez y cómo se ha utilizado la inteligencia artificial en el ajedrez. 			
Observaciones			
N/A			

5.2.2 Diseño

Código	Nombre	Responsable	Estimación (días)
1.2.1	Diseño de estructuras y funcionalidades de la página web	DG, JP, PS	20
Descripción			
<ul style="list-style-type: none"> El objetivo de la tarea es crear una plataforma en línea que sea útil y fácil de usar para los jugadores y aficionados del ajedrez, para ello se hará el diseño de las entidades principales de la web a la vez que se realiza la investigación. 			
Observaciones			
N/A			

Código	Nombre	Responsable	Estimación (días)
1.2.2	Implementación del ajedrez usando las tecnologías seleccionadas	DG, PS	20
Descripción			
<ul style="list-style-type: none"> En esta tarea implementaremos un módulo capaz de jugar al ajedrez y reconocer las reglas del juego, para ello se diseñará una interfaz sencilla y fácil de usar que permita a los usuarios jugar al ajedrez. 			
Observaciones			
N/A			

Código	Nombre	Responsable	Estimación (días)
1.2.3	Implementación de la base de datos con partidas de grandes maestros	I, PS	20
Descripción			
<ul style="list-style-type: none"> El objetivo de esta tarea es brindar a los usuarios la posibilidad de reproducir partidas de grandes maestros a lo largo de la historia, para ello se hará el filtrado correspondiente para obtener los datos y la carga en la base de datos de prueba. 			
Observaciones			
N/A			

5.2.3 Desarrollo

Código	Nombre	Responsable	Estimación (días)
1.3.1	Investigación y diseño de los algoritmos a implementar en la web	DG, I, PS	20
Descripción			
<ul style="list-style-type: none"> Con esta tarea buscamos diseñar uno o varios algoritmos que sean capaces de jugar al ajedrez contra los usuarios de la aplicación. Para ello se pondrá de acuerdo el programador con la investigación previamente realizada. También se realizará el modelado de la plataforma para ejecutar los algoritmos. 			
Observaciones			
N/A			

Código	Nombre	Responsable	Estimación (días)
1.3.2	Implementación de los algoritmos a implementar en la web	I, PS	20
Descripción			
<ul style="list-style-type: none"> Se implementarán los algoritmos previamente diseñados en la página web con las tecnologías previamente seleccionadas. 			
Observaciones			
N/A			

Código	Nombre	Responsable	Estimación (días)
1.3.3	Pruebas de validación y corrección de errores en la web	JP, PS	20
Descripción			
<ul style="list-style-type: none"> En esta tarea se realizarán las pruebas necesarias para confirmar que nuestros algoritmos no tienen problemas de rendimiento que puedan perjudicar la funcionalidad de la página web. 			
Observaciones			
N/A			

5.2.4 Integración

Código	Nombre	Responsable	Estimación (días)
1.4.1	Integración de la página web y las entidades principales	DG, PS	20

Descripción

- En esta tarea, integraremos los roles de administrador, de usuario y para los visitantes de la página. También se le creará el sistema de gestión para las entidades principales.

Observaciones

N/A

Código	Nombre	Responsable	Estimación (días)
1.4.2	Pruebas de integración y corrección de errores	JP, PS	20

Descripción

- El objetivo de esta tarea es verificar que todas las entidades tienen acceso a sus respectivos recursos y corregir los posibles errores en caso de que los hubiera.

Observaciones

N/A

5.2.5 Pruebas y Cierre

Código	Nombre	Responsable	Estimación (días)
1.5.1	Pruebas de funcionalidad rendimiento y seguridad	JP, PS	20

Descripción

- En esta tarea se realizará la comprobación de que todas las funcionalidades de la página están en orden y que no existan problemas que puedan comprometer a nuestra aplicación.

Observaciones

N/A

Código	Nombre	Responsable	Estimación (días)
1.5.2	Corrección de errores y optimización de la web	DG, PS	20

Descripción

- El objetivo principal de esta tarea es corregir los errores que hallamos encontrado a lo largo de la tarea anterior y realizar las posibles optimizaciones que surjan.

Observaciones

N/A

Código	Nombre	Responsable	Estimación (días)
1.5.3	Lanzamiento de la web y publicación de los resultados	PS	20

Descripción

- Se lanzará la aplicación para que la pueda ejecutar el público y se publicaran los resultados obtenidos en la investigación.

Observaciones

N/A

5.3 Asignación de recursos

A continuación, mostraremos una tabla con el número de horas que deberá trabajar cada recurso en el proyecto y el coste asociado a cada uno de ellos.

Recurso	Coste/Hora	Nº total de horas por recurso	Coste total por recurso
Investigador	15 €/h	808 hrs	12120 €
Jefe de Proyecto	18 €/h	888 hrs	15984 €
Programador Senior	12 €/h	1288 hrs	15456 €
Diseñador Gráfico	10 €/h	888 hrs	8880 €

5.4 Asignación de tareas

A continuación, se muestra una tabla con la asignación de cada recurso a cada tarea del proyecto.

Tarea	Recurso	Esfuerzo en horas
Investigación		
Investigación de conceptos básicos del ajedrez y su historia	Investigador	160 horas
Investigación de los algoritmos utilizados en el ajedrez	Jefe de Proyecto	320 horas
	Investigador	320 horas
Investigación sobre la inteligencia artificial en el ajedrez	Jefe de Proyecto	320 horas
	Investigador	320 horas
Diseño		
Diseño de estructura y funcionalidades de la página web	Diseñador Gráfico	1080 horas
	Programador Senior	1080 horas
	Jefe de Proyecto	1080 horas
	Diseñador Gráfico	224 horas

Implementación del ajedrez utilizando las tecnologías de desarrollo seleccionadas	Programador Senior	224 horas
Implementación de la base de datos con partidas de grandes maestros	Investigador	224 horas
	Programador Senior	224 horas
Desarrollo		
Investigación y diseño de los algoritmos a implementar en la web	Diseñador Gráfico	528 horas
	Programador Senior	528 horas
	Investigador	528 horas
Implementación de los algoritmos de ajedrez	Investigador	80 horas
	Programador Senior	80 horas
Pruebas de validación y corrección de errores en la web	Jefe de proyecto	80 horas
	Programador Senior	80 horas
Integración		
Integración de la página web y las entidades principales	Diseñador Gráfico	320 horas
	Programador Senior	320 horas

Pruebas de integración y corrección de errores	Jefe de Proyecto	112 horas
	Programador Senior	112 horas
Pruebas y Cierre		
Pruebas de funcionalidad, rendimiento y seguridad	Jefe de Proyecto	224 horas
	Programador Senior	224 horas
Corrección de errores y optimización de la web	Diseñador Gráfico	160 horas
	Programador Senior	160 horas
Lanzamiento de la web y publicación de los resultados	Programador Senior	40 horas

6. EVALUACIÓN Y PLANIFICACIÓN DE RIESGOS

Para evaluar los diferentes riesgos seguiremos el siguiente esquema, para tener en cuenta tanto su impacto como su probabilidad:

		PROBABILIDAD		
		BAJA	MEDIA	ALTA
SEVERIDAD	BAJA	MUY LEVE	LEVE	MODERADO
	MEDIA	LEVE	MODERADO	GRAVE
	ALTA	MODERADO	GRAVE	MUY GRAVE

A continuación, mostramos los riesgos identificados para el proyecto:

Origen del Riesgo	ID	Descripción del Riesgo	Probabilidad	Severidad	Plan de Contingencia / Acciones Propuestas	Prioridad
Aspectos tecnológicos	R01	Cambio de tecnología por algún error en la escogida anteriormente.	Baja	Alta	Realizar una copia de seguridad para realizar la migración a posteriori.	Moderada

Riesgo de seguridad	R02	La página web podría ser vulnerable a ataques informáticos o robos de información de los usuarios.	Alta	Alta	Para mitigar este riesgo, es importante implementar medidas de seguridad robustas, como encriptación de datos, autenticación de usuario y monitoreo constante de la actividad en la página web.	Grave
Riesgo de rendimiento	R03	Existe la posibilidad de que la página web no funcione correctamente o se ralentice debido a una carga de tráfico o uso intensivo de recursos del servidor. Este riesgo podría surgir si la página web no se diseña para manejar grandes cantidades de tráfico o si los recursos del servidor no son	Media	Baja	Para mitigar el riesgo de rendimiento, es importante llevar a cabo pruebas de rendimiento y de carga en la página web antes de su lanzamiento.	Moderada

		suficientes para manejar la demanda.				
--	--	--------------------------------------	--	--	--	--

7. TEMAS ABIERTOS Y DECISIONES PENDIENTES

Al ser un trabajo de investigación queda pendiente todavía los algoritmos a utilizar para la implementación de la página web, así como la información a contrastar en la misma.

8. OTROS ASPECTOS DEL PROYECTO

8.1 Ubicación del equipo de desarrollo

Nuestro equipo trabajará desde casa con todas las herramientas instaladas en sus dispositivos.

8.2 Entorno de desarrollo y herramientas

Nuestro entorno de desarrollo estará compuesto por:

- Programador senior, jefe de proyecto, diseñador gráfico y el investigador.
- Utilizaremos “Github” para servir a los usuarios con las aplicaciones.

8.3 Equipos

Los equipos del programador senior, el jefe de proyecto y el investigador deben disponer de:

- Visual Studio Code: Como entorno de desarrollo principal.
- XAMPP: Para crear una base de datos local y poder hacer las pruebas con la web.
- Laravel: Será el Framework principal para desarrollar la página web.
- Python: Será el lenguaje de programación a utilizar para el desarrollo de la investigación.
- Google Drive: También deberán tener acceso a Google Drive para usar la herramienta de Google Colaboratory.

El diseñador gráfico deberá disponer de:

- Laravel: Para el diseño de la página web
- Visual Studio Code
- Google Drive

**ANEXO II – PLANTILLA DOCUMENTO
ANÁLISIS**

DOCUMENTO DE ANÁLISIS

“ALPHAUPO”

Edición: 1

Fecha: 16-05-2023

CONTROL Y REGISTRO DE CAMBIO DEL DOCUMENTO

CONTROL	
Proyecto	ALPHAUPO WEBSITE
Denominación	Documento de análisis para ALPHAUPO
Fecha	día de mes de año
Edición	1
Grupo	ALPHAUPO
Autores	Eugenio Menacho de Góngora

REGISTRO DE CAMBIOS		
VERSIÓN	DESCRIPCIÓN DEL CAMBIO	FECHA DEL CAMBIO
1	Creado el modelo de dominio, puntos de definición del sistema y especificación de requisitos.	18/04/2023
2	Añadido el diseño de interfaces y finalización del documento.	10/05/2023

1. DEFINICIÓN DEL SISTEMA

1.1 Alcance del sistema

Se pretende realizar una investigación a la informática en el ajedrez para realizar a posteriori una página web que demuestre los conocimientos adquiridos. Esta aplicación contará con una implementación del juego del ajedrez y una base de datos con partidas de grandes maestros. A su vez, la página web tendrá un sistema de gestión para los usuarios y las partidas de grandes maestros.

OBJ-1		Investigación acerca de la informática en el ajedrez
Versión	1	
Autores	Eugenio Menacho de Góngora	
Fuente	Investigación sobre la informática en el ajedrez	
Descripción	El objetivo principal del proyecto es entender y comprender más acerca del juego, para ello lo más importante será realizar una investigación sobre cómo ha evolucionado la informática y cómo podríamos adaptarla para implementarla en el ajedrez.	
Importancia	Alta	
Estado	Aprobado	
Comentarios		

OBJ-2		Motor de ajedrez
Versión	1	
Autores	Eugenio Menacho de Góngora	
Fuente	Investigación sobre la informática en el ajedrez	
Descripción	El sistema deberá implementar varios algoritmos de ajedrez y contar con un motor que permita a los usuarios jugar partidas de ajedrez en línea contra la computadora, ofreciendo diferentes niveles de dificultad para adaptarse a las habilidades de cada usuario. Además, el motor de ajedrez deberá ser capaz de realizar movimientos válidos y legales, asegurando una experiencia de juego auténtica y desafiante para los usuarios.	
Importancia	Alta	
Estado	Aprobado	
Comentarios	Se deberá implementar una interfaz que permita jugar al juego.	

OBJ-3	Base de datos
Versión	1
Autores	Eugenio Menacho de Góngora
Fuente	Investigación sobre la informática en el ajedrez
Descripción	El sistema debe contar con una base de datos de ajedrez que permita a los usuarios analizar partidas, incluyendo la posibilidad de realizar búsquedas avanzadas por diferentes criterios, tales como jugadores y eventos. Además, la base de datos deberá ser actualizable y permitir la inserción de nuevas partidas y la eliminación de partidas obsoletas por parte de los administradores.
Importancia	Alta
Estado	Aprobado
Comentarios	Inclusión de un modelo de inteligencia artificial implementado anteriormente.

OBJ-4	Sistema de gestión de usuarios y partidas
Versión	1
Autores	Eugenio Menacho de Góngora
Fuente	Investigación sobre la informática en el ajedrez
Descripción	El sistema deberá contar con un sistema de registro de usuarios que permita la creación de perfiles y la gestión de datos personales y partidas jugadas. Permitiendo la modificación y eliminación para todas estas entidades.
Importancia	Alta
Estado	Aprobado
Comentarios	

A continuación, se mostrará el modelo de dominio que usaremos para codificar el juego:

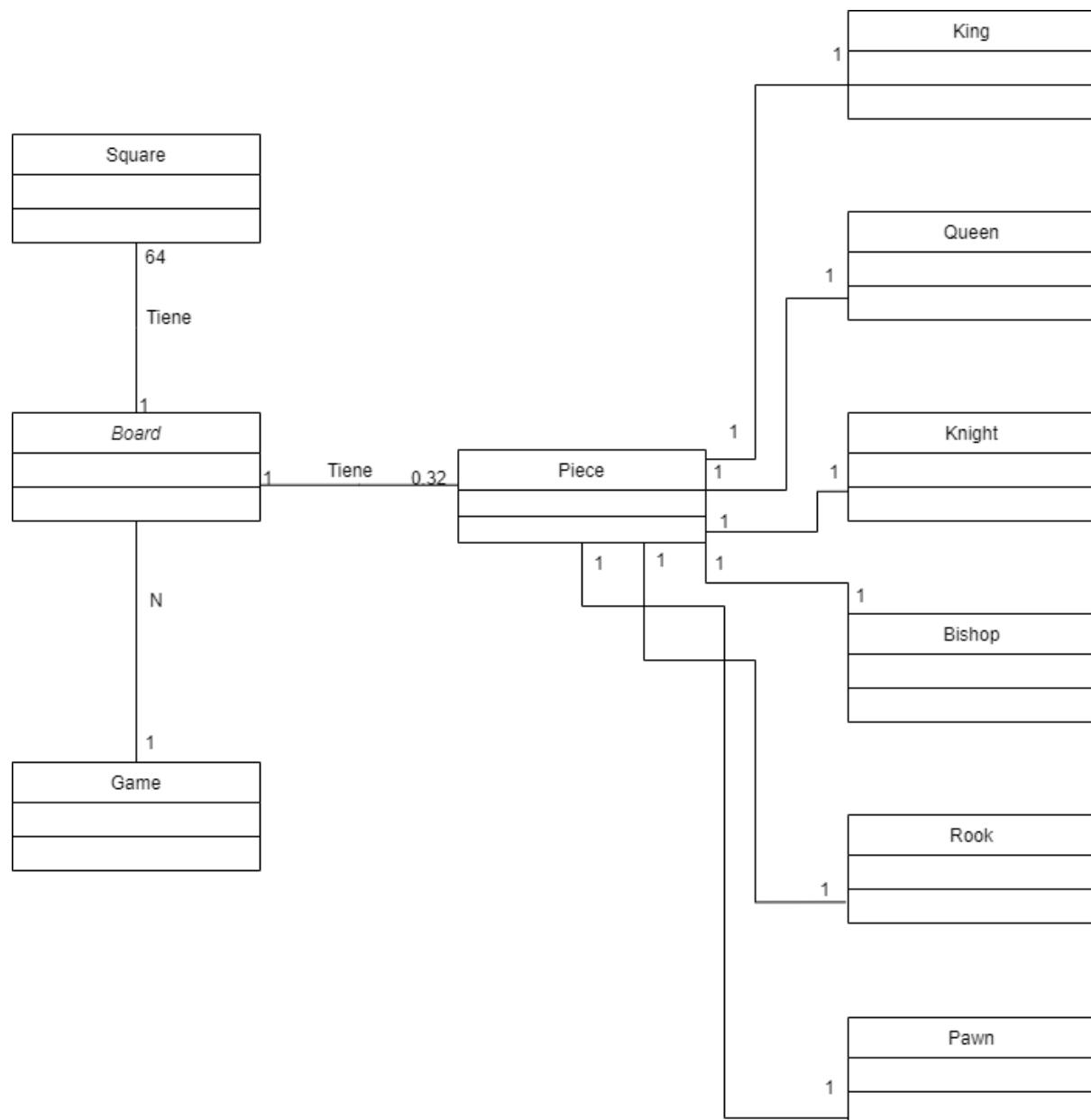


Figure 143: Menacho, E.

2. ESPECIFICACIÓN DE REQUISITOS

2.1 Catálogo de requisitos

A continuación, mostraremos los requisitos funcionales identificados para el proyecto:

2.1.1 Requisitos funcionales

RF-01	Implementación del juego de ajedrez
Versión	1
Autores	Eugenio Menacho de Góngora
Fuentes	Investigación sobre la informática en el ajedrez
Objetivos asociados	<ul style="list-style-type: none"> • OBJ-1 • OBJ-2 • OBJ-3 2.
Descripción	Los usuarios podrán usar un motor de ajedrez previamente implementado para jugar contra las implementaciones algorítmicas del juego y para reproducir las partidas desde la base de datos.
Actores	Actor Usuario
Comentarios	

RF-02	Jugar contra algoritmos de ajedrez				
Versión	1				
Autores	Eugenio Menacho de Góngora				
Fuentes	Investigación sobre la informática en el ajedrez				
Objetivos asociados	<ul style="list-style-type: none"> • OBJ-1 • OBJ-2 • OBJ-3 <p>3.</p>				
Descripción	Los usuarios podrán usar el motor de ajedrez previamente implementado para jugar partidas contra diversas implementaciones algorítmicas que han sido extraídas de la investigación				
Actores	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="background-color: #cccccc; width: 50%;">Actor</td> <td style="width: 50%;"></td> </tr> <tr> <td style="background-color: #cccccc;">Usuario</td> <td></td> </tr> </table>	Actor		Usuario	
Actor					
Usuario					
Comentarios					

RF-03	Acceso y filtrado en la base de datos de grandes maestros
Versión	1
Autores	Eugenio Menacho de Góngora
Fuentes	Investigación sobre la informática en el ajedrez
Objetivos asociados	<ul style="list-style-type: none"> • OBJ-2 • OBJ-3 4.
Descripción	Los usuarios de la aplicación podrán acceder a la base de datos para reproducir las partidas de grandes maestros con el motor de ajedrez diseñado. También se deberá permitir el filtrado de las partidas por eventos y jugadores.
Actores	Actor
	Usuario
Comentarios	

RF-04	Gestión de usuarios y partidas		
Versión	1		
Autores	Eugenio Menacho de Góngora		
Fuentes	Investigación sobre la informática en el ajedrez		
Objetivos asociados	<ul style="list-style-type: none"> • OBJ-4 5. 		
Descripción	Los administradores podrán gestionar la base de datos al completo, permitiéndoles cambiar cualquier característica tanto de las partidas como de los usuarios. Se permitirán la eliminación y modificación de las partidas y lo mismo para los usuarios de la plataforma		
Actores	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="background-color: #cccccc; width: 10%;">Actor</td> </tr> <tr> <td>Administradores</td> </tr> </table>	Actor	Administradores
Actor			
Administradores			
Comentarios			

RF-05	Registro y acceso
Versión	1
Autores	Eugenio Menacho de Góngora
Fuentes	Investigación sobre la informática en el ajedrez
Objetivos asociados	<ul style="list-style-type: none"> • OBJ-4
Descripción	Se deberá poder registrarse como usuario de la aplicación siendo un visitante de la misma. También se permitirá el acceso a la aplicación con un usuario ya registrado en la plataforma.
Actores	Actor
	Visitante
Comentarios	

RF-06	Investigación sobre la informática en el ajedrez
Versión	1
Autores	Eugenio Menacho de Góngora
Fuentes	Investigación sobre la informática en el ajedrez
Objetivos asociados	<ul style="list-style-type: none">● OBJ-1 6.
Descripción	Se realizará una investigación sobre la influencia algorítmica en el ajedrez y los aspectos teóricos más relevantes a lo largo de la historia, así como los jugadores más importantes a lo largo de las últimas épocas.
Actores	Actor
	Visitante
Comentarios	

2.1.2 Requisitos no funcionales

RNF-01	Alto rendimiento
Versión	1
Autores	Eugenio Menacho de Góngora
Fuentes	Investigación sobre la informática en el ajedrez
Objetivos asociados	<ul style="list-style-type: none">• OBJ-2• OBJ-3 <p>7.</p>
Descripción	El sistema deberá ejecutar los algoritmos de ajedrez sin comprometer el rendimiento del servidor que se encargue de manejar la web. Tampoco se puede ver comprometido el rendimiento de la interfaz ejecutada por parte de los usuarios.
Comentarios	

RNF-02	Usabilidad
Versión	1
Autores	Eugenio Menacho de Góngora
Fuentes	Investigación sobre la informática en el ajedrez
Objetivos asociados	<ul style="list-style-type: none">• OBJ-2• OBJ-3 <p>8.</p>
Descripción	La página web debe ser fácil de usar y navegar para los usuarios. Esto incluye una interfaz de usuario intuitiva, una buena organización de la información y la capacidad de acceder rápidamente a las diferentes funciones y características de la página web.
Comentarios	

RNF-03	Seguridad
Versión	1
Autores	Eugenio Menacho de Góngora
Fuentes	Investigación sobre la informática en el ajedrez
Objetivos asociados	<ul style="list-style-type: none">• OBJ-2• OBJ-3
Descripción	La página debe ser segura y proteger la privacidad de los usuarios. Esto incluye la implementación de medidas de seguridad, como la encriptación de datos y la autenticación de usuarios, así como la protección contra ataques informáticos y la privacidad de la información de los usuarios.
Comentarios	

Matriz de Objetivos-Requisitos:

\Objetivos Requisitos	OBJ-01	OBJ-02	OBJ-03	OBJ-04
RF-01	X	X	X	
RF-02	X	X	X	
RF-03		X	X	
RF-04				X
RF-05				X
RF-06	X			

2.2 Especificación de los casos de uso

2.2.1 Identificación y definición de los actores

Para esta aplicación identificaremos como actores a aquellas personas que accedan a nuestra aplicación, es decir, cualquier usuario de la misma, para ello distinguimos tres roles principales, el administrador, el usuario y por último los visitantes.

Actores	Descripción
Act01: Usuarios	Son aquellas personas que acceden a la aplicación con un registro previo en la misma.
Act02: Administradores	Son aquellos usuarios que poseen privilegios para hacer modificaciones en la base de datos.
Act03: Visitantes	Son aquellas personas que acceden a la página sin registro.

2.2.2 Diagrama de Casos de Uso

A continuación, mostraremos los casos de uso a desarrollar para la página web. Representa los posibles casos de uso una vez iniciada la aplicación.

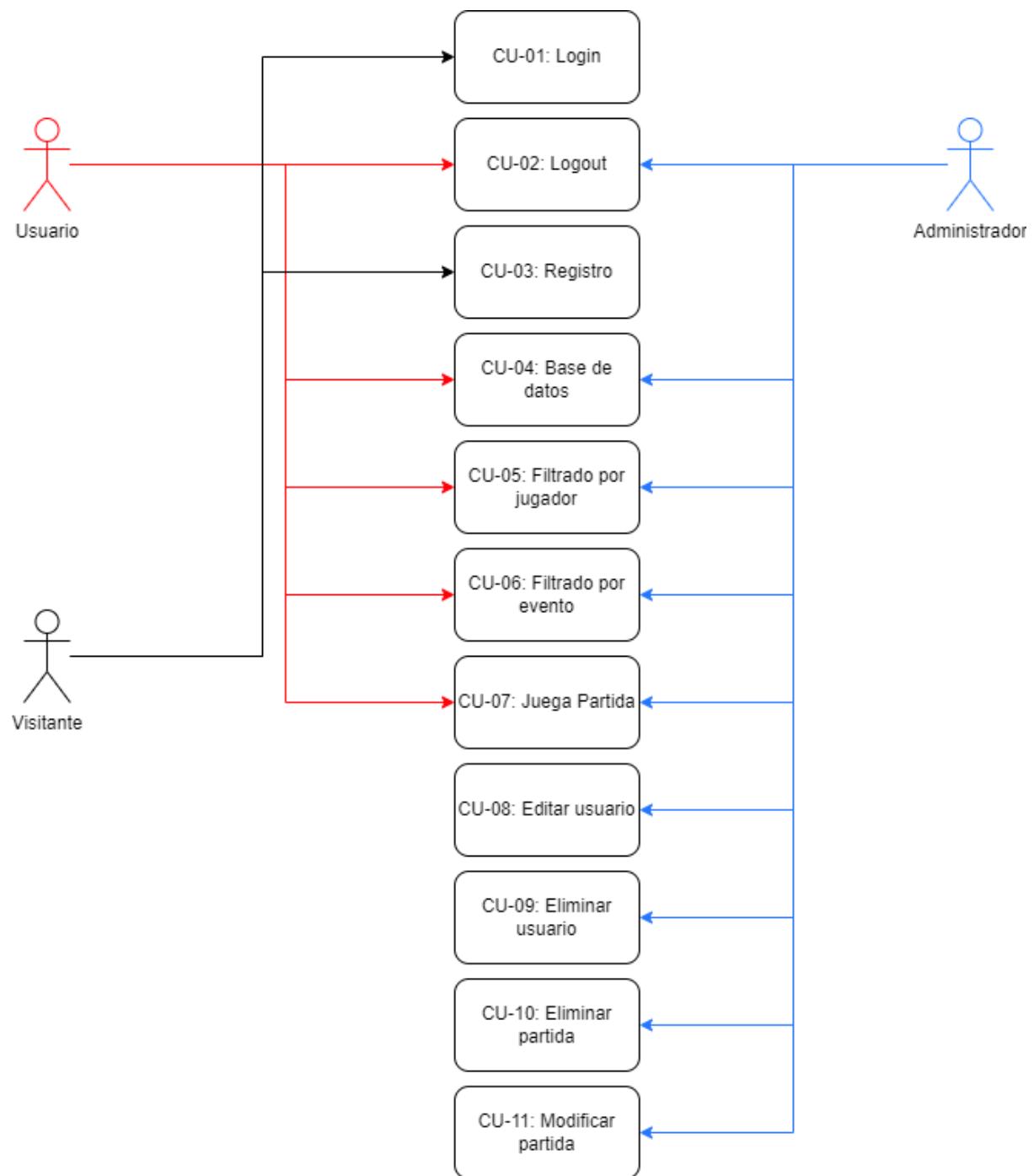


Figure 144: Menacho, E.

2.2.3 Especificación de Casos de Uso

CU-01	Login	
Descripción	El sistema debe permitir a los usuarios y administradores autenticarse en el sistema con su nombre de usuario y contraseña.	
Actores	Visitante	
Precondición	El usuario no puede estar logueado en el sistema.	
Postcondición	El usuario quedará logueado en la aplicación	
Flujo Normal	PASO	ACCIÓN
	1	El visitante selecciona la opción de login.
	2	El sistema solicita una dirección de correo y una contraseña.
	3	El visitante introduce las credenciales. (Nombre, Correo y Contraseña).
	4	El sistema comprueba que las credenciales introducidas son correctas y redirige al usuario a la pantalla principal de la aplicación, ya con acceso de usuario o administrador.
	5	El sistema finaliza el caso de uso
Flujos Alternativos	PASO	ACCIÓN
	4	Las credenciales introducidas no son válidas, así que el sistema avisa al usuario de ello y vuelve al paso 2.

CU-02	Logout	
Descripción	El sistema debe permitir a los usuarios cerrar sesión.	
Actores	Administrador, Usuario	
Precondición	Sesión iniciada	
Postcondición	Se eliminará la sesión previamente abierta	
Flujo Normal	PASO	ACCIÓN
	1	El alumno/profesor/administrador selecciona la opción de logout.
	2	El sistema cierra la sesión del usuario logeado y lo redirige a la página principal.

CU-03	Register	
Descripción	El sistema debe permitir a los visitantes registrarse en la web para poder acceder a las funcionalidades de la aplicación que corresponden al usuario.	
Actores	Visitante	
Precondición	El usuario no está registrado en el sistema.	
Postcondición	El usuario queda registrado en la aplicación y puede acceder a las funcionalidades que le corresponden dependiendo del rol.	
Flujo Normal	PASO	ACCIÓN
	1	El visitante selecciona la opción de registro en la página web.
	2	El sistema solicita la información del usuario
	3	El visitante introduce las credenciales.
	4	El sistema verifica la información ingresada y crea una cuenta para el usuario.
	5	El sistema redirige al usuario a la página de inicio con la sesión iniciada.
	6	El caso de uso finaliza.
Flujos Alternativos	PASO	ACCIÓN
	4	La información ingresada es incorrecta o incompleta, el sistema notifica al visitante y solicita que corrija la información.

CU-04	Base de Datos	
Descripción	El sistema debe permitir a los usuarios acceder a la base de datos de partidas de grandes maestros para reproducir partidas.	
Actores	Usuario, Administrador	
Precondición	El usuario está registrado en el sistema.	
Postcondición		
Flujo Normal	PASO	ACCIÓN
	1	El usuario selecciona la opción de Base de Datos.
	2	El sistema confirma que el usuario ha iniciado sesión.
	3	El sistema devuelve la tabla con la base de datos y la posibilidad de reproducir las partidas.
	4	El sistema finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	Si el usuario no ha iniciado sesión devuelve la pantalla de Login.

CU-05	Filtrado por Evento	
Descripción	El sistema debe permitir a los usuarios filtrar las partidas de la base de datos por los eventos relacionados a las partidas.	
Actores	Usuario, Administrador	
Precondición	El usuario está registrado en el sistema.	
Postcondición		
Flujo Normal	PASO	ACCIÓN
	1	El usuario selecciona la opción de Filtrado por Evento.
	2	El sistema confirma que el usuario ha iniciado sesión.
	3	El sistema devuelve los eventos a seleccionar desde la base de datos.
	4	El usuario introduce el evento.
	5	El sistema devuelve la tabla de las partidas en ese evento.
	6	El sistema finaliza el caso de uso
Flujos Alternativos	PASO	ACCIÓN
	2	Si el usuario no ha iniciado sesión devuelve la pantalla de Login.

CU-06	Filtrado por Jugador	
Descripción	El sistema debe permitir a los usuarios filtrar las partidas de la base de datos por los jugadores relacionados a las partidas.	
Actores	Usuario, Administrador	
Precondición	El usuario está registrado en el sistema.	
Postcondición		
Flujo Normal	PASO	ACCIÓN
	1	El usuario selecciona la opción de Filtrado por Jugador.
	2	El sistema confirma que el usuario ha iniciado sesión.
	3	El sistema devuelve los jugadores a seleccionar desde la base de datos.
	4	El usuario introduce el jugador.
	5	El sistema devuelve la tabla de las partidas en ese evento.
	6	El sistema finaliza el caso de uso
Flujos Alternativos	PASO	ACCIÓN
	2	Si el usuario no ha iniciado sesión devuelve la pantalla de Login.

CU-07	Juega Partida	
Descripción	El sistema debe permitir a los usuarios jugar una partida de ajedrez contra los algoritmos implementados.	
Actores	Usuario, Administrador	
Precondición	El usuario está registrado en el sistema.	
Postcondición		
Flujo Normal	PASO	ACCIÓN
	1	El usuario selecciona la opción de Juega Partida.
	2	El sistema confirma que el usuario ha iniciado sesión.
	3	El sistema devuelve la interfaz con la partida desde la posición inicial.
	4	El usuario selecciona el algoritmo y la profundidad.
	5	El usuario juega la partida.
	6	El sistema finaliza el caso de uso
Flujos Alternativos	PASO	ACCIÓN
	2	Si el usuario no ha iniciado sesión devuelve la pantalla de Login.

CU-08	Modificar Usuario	
Descripción	El sistema debe permitir a los administradores modificar a los usuarios existentes.	
Actores	Administrador	
Precondición	El usuario está registrado en el sistema y es administrador.	
Postcondición	Se actualiza en la base de datos el usuario seleccionado	
Flujo Normal	PASO	ACCIÓN
	1	El administrador selecciona gestión de usuarios.
	2	El sistema confirma que el usuario ha iniciado sesión y que es administrador.
	3	El sistema devuelve una tabla con los usuarios.
	4	El administrador selecciona el usuario a modificar.
	5	El sistema devuelve una interfaz con los datos a modificar.
	6	El administrador los datos del usuario a modificar.
	7	El sistema actualiza la información.
	8	El sistema devuelve la pantalla de gestión de usuarios.
	9	El sistema finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	Si el usuario no ha iniciado sesión devuelve la pantalla de Login.
	6	Si los datos introducidos son incorrectos el sistema muestra la información errónea en la interfaz de modificación para el usuario.

CU-09	Eliminar Usuario	
Descripción	El sistema debe permitir a los administradores eliminar a los usuarios existentes.	
Actores	Administrador	
Precondición	El usuario está registrado en el sistema y es administrador.	
Postcondición	Se elimina de la base de datos el usuario seleccionado.	
Flujo Normal	PASO	ACCIÓN
	1	El administrador selecciona gestión de usuarios.
	2	El sistema confirma que el usuario ha iniciado sesión y que es administrador.
	3	El sistema devuelve una tabla con los usuarios.
	4	El administrador selecciona el usuario a eliminar.
	5	El sistema elimina al usuario de la base de datos.
	6	El sistema devuelve la pantalla de gestión de usuarios.
	7	El sistema finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	Si el usuario no ha iniciado sesión devuelve la pantalla de Login.

CU-10	Modificar Partida	
Descripción	El sistema debe permitir a los administradores modificar a las partidas existentes.	
Actores	Administrador	
Precondición	El usuario está registrado en el sistema y es administrador.	
Postcondición	Se actualiza en la base de datos la partida seleccionada.	
Flujo Normal	PASO	ACCIÓN
	1	El administrador selecciona gestión de partidas.
	2	El sistema confirma que el usuario ha iniciado sesión y que es administrador.
	3	El sistema devuelve una tabla con las partidas.
	4	El administrador selecciona la partida a modificar.
	5	El sistema devuelve una interfaz con los datos a modificar.
	6	El administrador los datos de la partida a modificar.
	7	El sistema actualiza la información.
	8	El sistema devuelve la pantalla de gestión de partidas.
9	El sistema finaliza el caso de uso.	
Flujos Alternativos	PASO	ACCIÓN
	2	Si el usuario no ha iniciado sesión devuelve la pantalla de Login.
	6	Si los datos introducidos son incorrectos el sistema muestra la información errónea en la interfaz de modificación para la partida.

CU-11	Eliminar Partida	
Descripción	El sistema debe permitir a los administradores eliminar las partidas existentes.	
Actores	Administrador	
Precondición	El usuario está registrado en el sistema y es administrador.	
Postcondición	Se elimina de la base de datos la partida seleccionada.	
Flujo Normal	PASO	ACCIÓN
	1	El administrador selecciona gestión de partidas.
	2	El sistema confirma que el usuario ha iniciado sesión y que es administrador.
	3	El sistema devuelve una tabla con las partidas.
	4	El administrador selecciona la partida a eliminar.
	5	El sistema elimina la partida de la base de datos.
	6	El sistema devuelve la pantalla de gestión de partidas.
	7	El sistema finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	Si el usuario no ha iniciado sesión devuelve la pantalla de Login.

3. SUBSISTEMAS DE ANÁLISIS

3.1 Identificación de los subsistemas de análisis

Para el desarrollo de la página web hemos identificado los siguientes subsistemas:

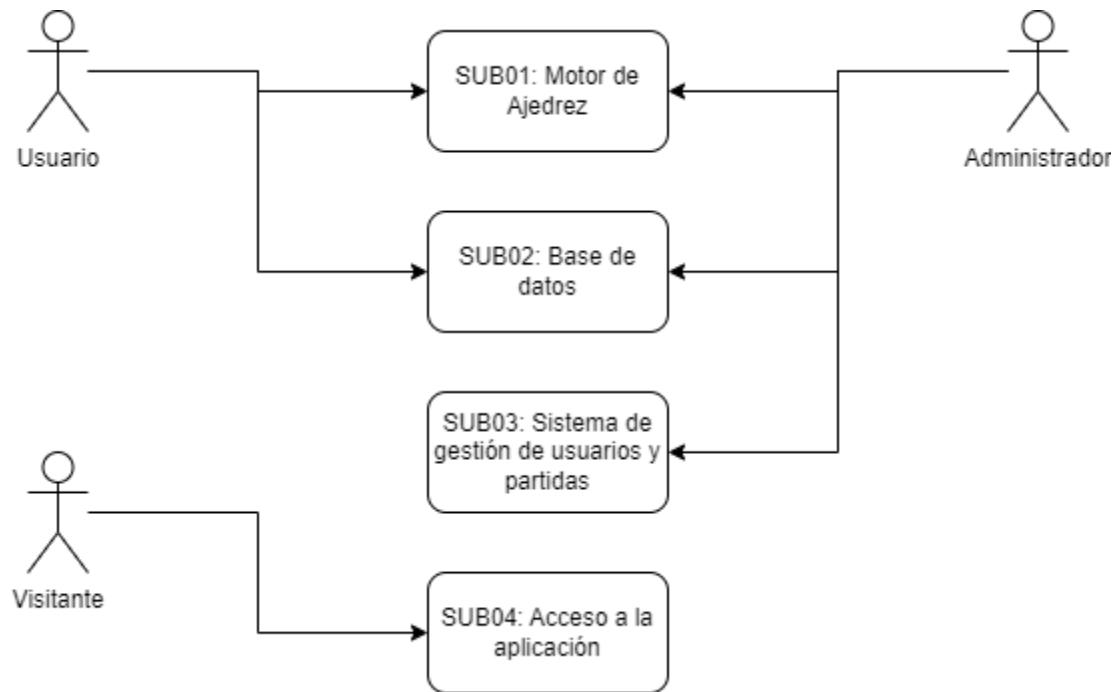


Figure 145: Menacho, E.

En el primer subsistema agrupamos el desarrollo del motor de ajedrez y los algoritmos que se ejecutarán con este motor. En el subsistema de la base de datos incluimos el acceso a la base de datos de partidas de maestros y los filtrados de búsqueda realizados sobre las partidas. En el subsistema de gestión de usuarios y partidas agrupamos todas las herramientas incluidas para la modificación de los usuarios y las partidas. Por último, hemos identificado un subsistema para la gestión del acceso a la aplicación, es decir, los registros, los inicios de sesión y los cierres de sesión.

3.2 Relaciones entre subsistemas de Análisis

Ninguno de los subsistemas indicados anteriormente está relacionado, exceptuando el Motor de Ajedrez y el de la Base de datos, ya que la base de datos requiere del motor para poder visualizar y analizar las partidas.

3.3 Matriz de Trazabilidad

Subsistemas / Casos de Uso	SUB-01	SUB-02	SUB-03	SUB-04
CU-01				X
CU-02				X
CU-03				X
CU-04	X	X		
CU-05		X		
CU-06		X		
CU-07	X			
CU-08			X	

CU-09			X	
CU-10			X	
CU-11			X	

4. ANÁLISIS DE LAS CLASES POR SUBSISTEMAS

4.1 Paquetes de clases de negocio

4.1.1 Subsistema “SUB01 – Motor de ajedrez”

A continuación, se muestra el diagrama de clases correspondiente a este subsistema:

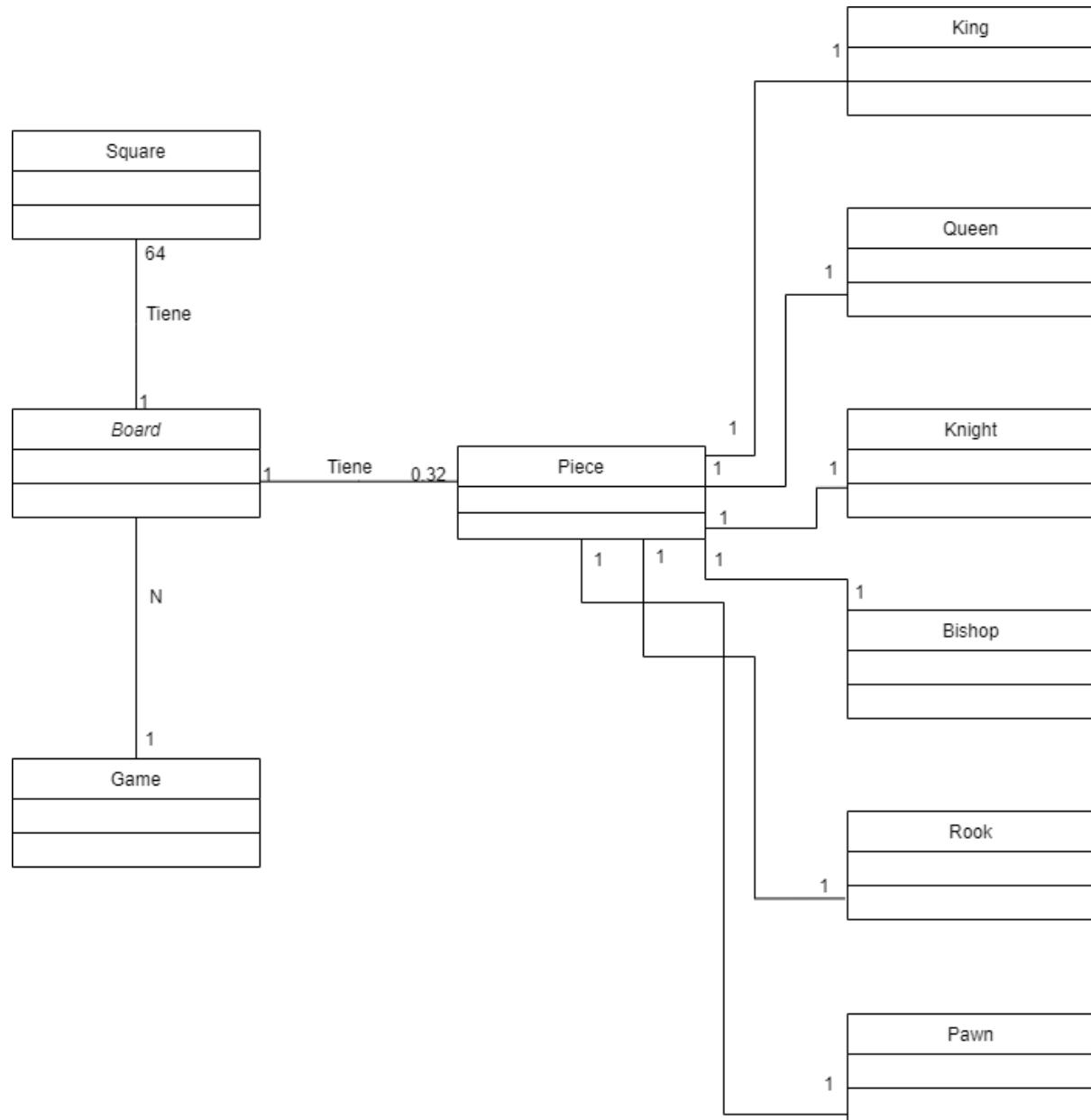


Figure 145: Menacho, E.

CN-0001: Game			
Responsabilidades	PlayGame		
	CreateBoard		
Atributos	Nombre	Tipo	Descripción
Participación en Casos de Uso	CU-07: Jugar al Ajedrez – Se inicia un “Game” cada vez que se juega una partida.		

CN-0002: Board			
Responsabilidades	SetUpBoard		
	MakeMove		
	RevertMove		
Atributos	Nombre	Tipo	Descripción
Participación en Casos de Uso	CU-07: Jugar al Ajedrez – Cuando queremos iniciar una partida creamos un tablero que incluya las piezas necesarias.		
	CU-04: Base de Datos – En la base de datos se creará una partida cuando queramos analizar una partida y por lo tanto un tablero.		

CN-0003: Square			
Responsabilidades			
Atributos	Nombre	Tipo	Descripción
Participación en Casos de Uso	CU-07: Jugar al Ajedrez – Cuando queremos iniciar una partida creamos un tablero con una matriz de posibles posiciones.		

CN-0004: Piece			
Responsabilidades			
Atributos	Nombre	Tipo	Descripción
	Color	int	Color de la pieza, 1 para negras 0 para blancas.
	Pos	Int[2]	Un array con la posición de la pieza.
Participación en Casos de Uso	CU-07: Jugar al Ajedrez – Cuando queremos iniciar una partida creamos un tablero con una matriz que contenga las piezas en la posición inicial		

CN-0005: King			
Responsabilidades	getMoves		
Atributos	Nombre	Tipo	Descripción
Participación en Casos de Uso	CU-07: Jugar al Ajedrez – Cuando queremos iniciar una partida creamos un tablero con una matriz que contenga las piezas en la posición inicial		

CN-0006: Queen			
Responsabilidades	getMoves		
Atributos	Nombre	Tipo	Descripción
Participación en Casos de Uso	CU-07: Jugar al Ajedrez – Cuando queremos iniciar una partida creamos un tablero con una matriz que contenga las piezas en la posición inicial		

CN-0007: Rook			
Responsabilidades	getMoves		
Atributos	Nombre	Tipo	Descripción
Participación en Casos de Uso	CU-07: Jugar al Ajedrez – Cuando queremos iniciar una partida creamos un tablero con una matriz que contenga las piezas en la posición inicial		

CN-0008: Bishop			
Responsabilidades	getMoves		
Atributos	Nombre	Tipo	Descripción
Participación en Casos de Uso	CU-07: Jugar al Ajedrez – Cuando queremos iniciar una partida creamos un tablero con una matriz que contenga las piezas en la posición inicial		

CN-0009: Knight			
Responsabilidades	getMoves		
Atributos	Nombre	Tipo	Descripción
Participación en Casos de Uso	CU-07: Jugar al Ajedrez – Cuando queremos iniciar una partida creamos un tablero con una matriz que contenga las piezas en la posición inicial		

CN-0010: Pawn			
Responsabilidades	getMoves		
Atributos	Nombre	Tipo	Descripción
Participación en Casos de Uso	CU-07: Jugar al Ajedrez – Cuando queremos iniciar una partida creamos un tablero con una matriz que contenga las piezas en la posición inicial		

4.1.2 Subsistema “SUB02 – Base de Datos”

Este subsistema debe contener las partidas de la base de datos. Las consultas se harán a una entidad llamada “Partida” que contendrá las partidas de grandes maestros.

CN-0001: Partida			
Responsabilidades	getPartidas getEventos getJugadores getPartidasByJugador getPartidasByEvento		
Atributos	Nombre	Tipo	Descripción
	ID	int	Identificador de la partida.
	Black	str	Código de nombre de jugador de piezas negras.
	BlackElo	int	Ranking en puntuación fide del jugador con piezas negras.
	Date	Date	Fecha de la partida.
	ECO	str	Código de enciclopedia de aperturas.
	Event	str	Evento en el que tuvo lugar la partida.
	Game	str	Partida en notación algebraica.
	Result	str	Resultado de la partida.
	Round	int	Ronda del enfrentamiento en el evento
	Site	str	Página en la que tuvo lugar.
	White	str	Código de nombre de jugador de piezas blancas.
	WhiteElo	int	Ranking en puntuación fide del jugador con piezas blancas.

Participación en Casos de Uso	CU-07: Jugar al Ajedrez – Cuando queremos reproducir una partida accederemos al motor de ajedrez para que pueda simular la notación dada.
	CU-04: Base de Datos – Accederemos a la entidad de partida cada vez que queramos cumplir el caso de uso.
	CU-05: Filtrado por Evento – Cuando queramos hacer el filtrado de un evento lo haremos sobre esta entidad.
	CU-06: Filtrado por Jugador – Cuando queramos hacer el filtrado de un jugador lo haremos sobre esta entidad.

4.1.3 Subsistema “SUB03 – “Subsistema de gestión de partidas y usuarios”

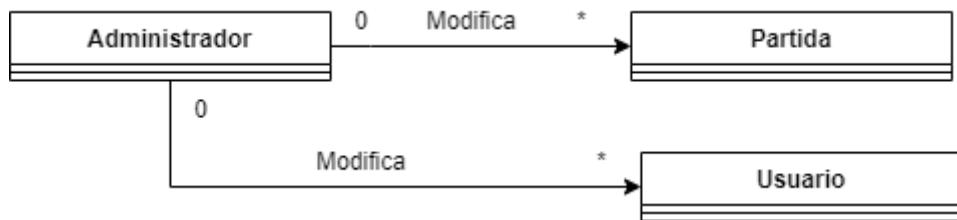


Figure 146: Menacho, E.

CN-0001: Partida			
Responsabilidades	isAdmin		
Atributos	Nombre	Tipo	Descripción
	ID	int	Identificador de la partida.
	Black	str	Código de nombre de jugador de piezas negras.
	BlackElo	int	Ranking en puntuación fide del jugador con piezas negras.
	Date	Date	Fecha de la partida.
	ECO	str	Código de enciclopedia de aperturas.
	Event	str	Evento en el que tuvo lugar la partida.
	Game	str	Partida en notación algebraica.
	Result	str	Resultado de la partida.
	Round	int	Ronda del enfrentamiento en el evento
	Site	str	Página en la que tuvo lugar.
	White	str	Código de nombre de jugador de piezas blancas.
Participación en Casos de Uso	CU-10: Modificar partida – Cuando queremos modificar una partida lo haremos sobre esta entidad.		
	CU-11: Eliminar partida – Cuando queremos eliminar una partida lo haremos sobre esta entidad.		

CN-0002: Usuario			
Responsabilidades	isAdmin		
Atributos	Nombre	Tipo	Descripción
	ID	int	Identificador de la partida.
	Name	str	Código de nombre de jugador de piezas negras.
	Email	int	Ranking en puntuación fide del jugador con piezas negras.
	password	Date	Fecha de la partida.
	Created_at	str	Código de enciclopedia de aperturas.
Participación en Casos de Uso	CU-08: Modificar usuario – Cuando queremos modificar un usuario lo haremos sobre esta entidad.		
	CU-09: Eliminar usuario – Cuando queremos eliminar un usuario lo haremos sobre esta entidad.		

4.1.4 Subsistema “SUB04 – “Acceso a la aplicación”

Para el acceso a la aplicación utilizaremos la entidad de “Usuario” como principal.

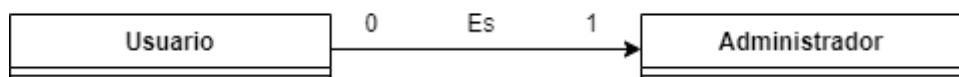


Figure 147: Menacho, E.

CN-0001: Usuario			
Responsabilidades	Login Logout Register		
Atributos	Nombre	Tipo	Descripción
	ID	int	Identificador de la partida.
	Name	str	Código de nombre de jugador de piezas negras.
	Email	int	Ranking en puntuación fide del jugador con piezas negras.
	password	Date	Fecha de la partida.
	Created_at	str	Código de enciclopedia de aperturas.
Participación en Casos de Uso	CU-01: Login – Cuando iniciamos sesión comprobamos que existe el usuario y coincide la contraseña.		
	CU-02: Logout – Cuando queremos eliminar la sesión del usuario accedemos a la entidad.		
	CU-03: Register– Cuando registramos a un usuario accedemos a la entidad.		

5. INTERFACES DE USUARIO

5.1 Principios de Diseño de la Interfaz de Usuario

Nuestra página web contará con una interfaz clara y simple que permita a cualquier visitante realizar un registro en la página de manera intuitiva. También los usuarios registrados en la página podrán navegar entre las diferentes funcionalidades de la página, de esta manera conseguimos que la web sea más fácil de usar. Para ello proponemos el siguiente diseño para todas las páginas:

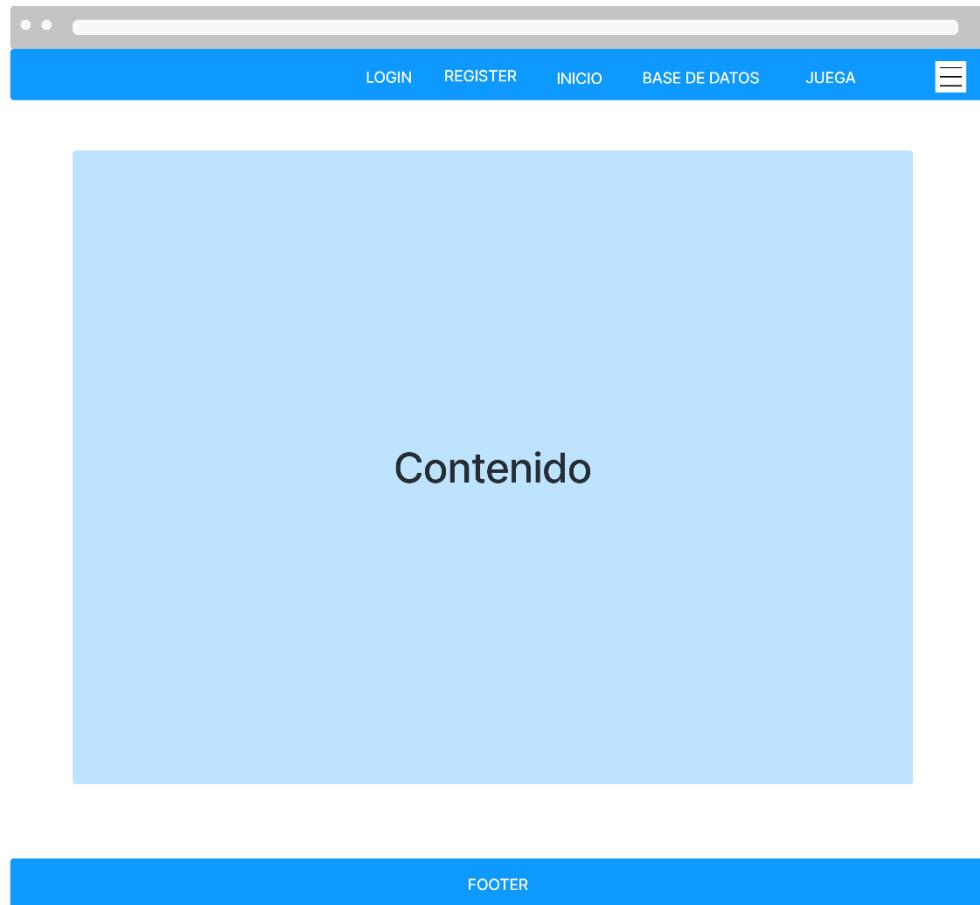


Figure 148: Menacho, E.

Esta será la plantilla a utilizar en todas las interfaces de usuario, para ello usaremos un “header” que contenga los enlaces a las funcionalidades principales del sistema. Y un “Footer” que permita al usuario navegar si este se encuentra más abajo en la web.

5.2 Interfaces principales aplicación

5.2.1 Interfaz del módulo IU-0001 – Juega

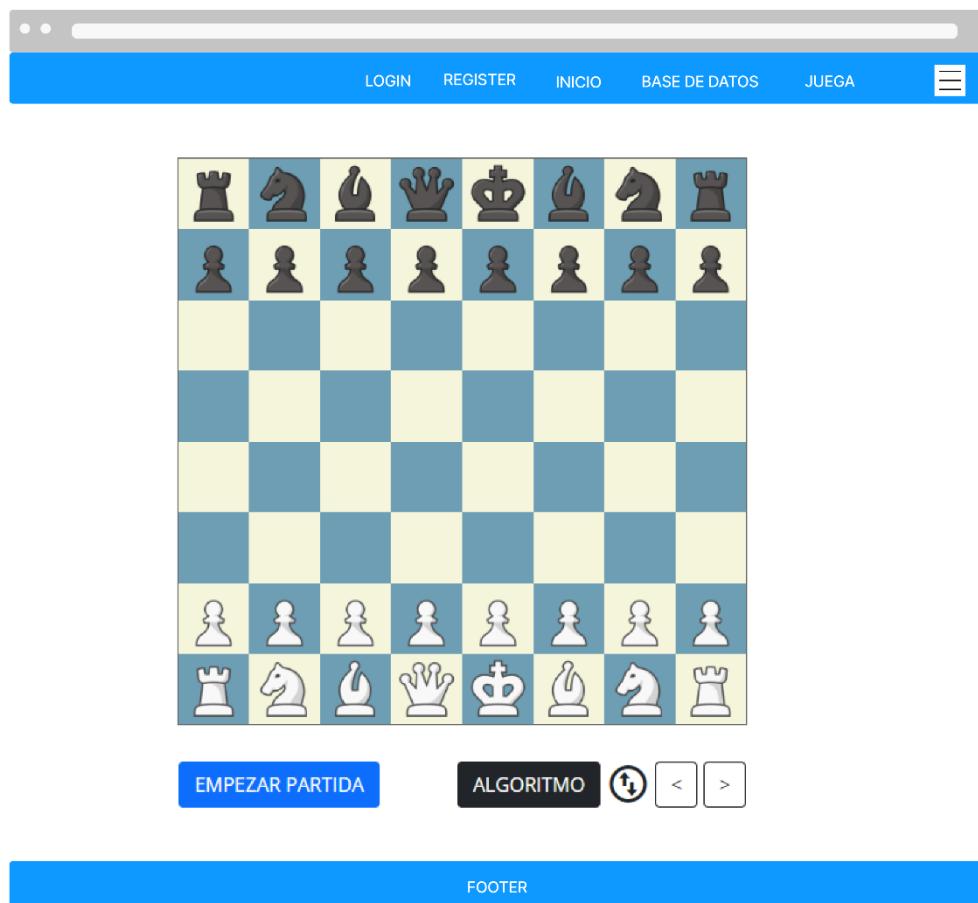


Figure 149: Menacho, E.

IU-0001: Juega						
Descripción	Muestra la interfaz desde la que podremos jugar a las partidas.					
Campos	Nombre	Tipo Datos	Editable/ Consulta	Oblig.	Descripción	
Botones /Enlaces	Nombre		Acción			
	Botón Empezar Partida		Crea un tablero con una posición inicial de ajedrez listo para jugar una partida.			
	Botón Algoritmo		Abre una ventana para seleccionar el algoritmo y la profundidad.			
	Botón Cambiar lado		Rota el tablero y realiza un movimiento por parte del algoritmo seleccionado.			
	Botón Deshacer movimiento		Vuelve hacia el movimiento anterior.			
	Botón volver al movimiento		Hace el movimiento ya realizado si existiera.			

5.2.2 Interfaz del módulo IU-0002 – Base de Datos

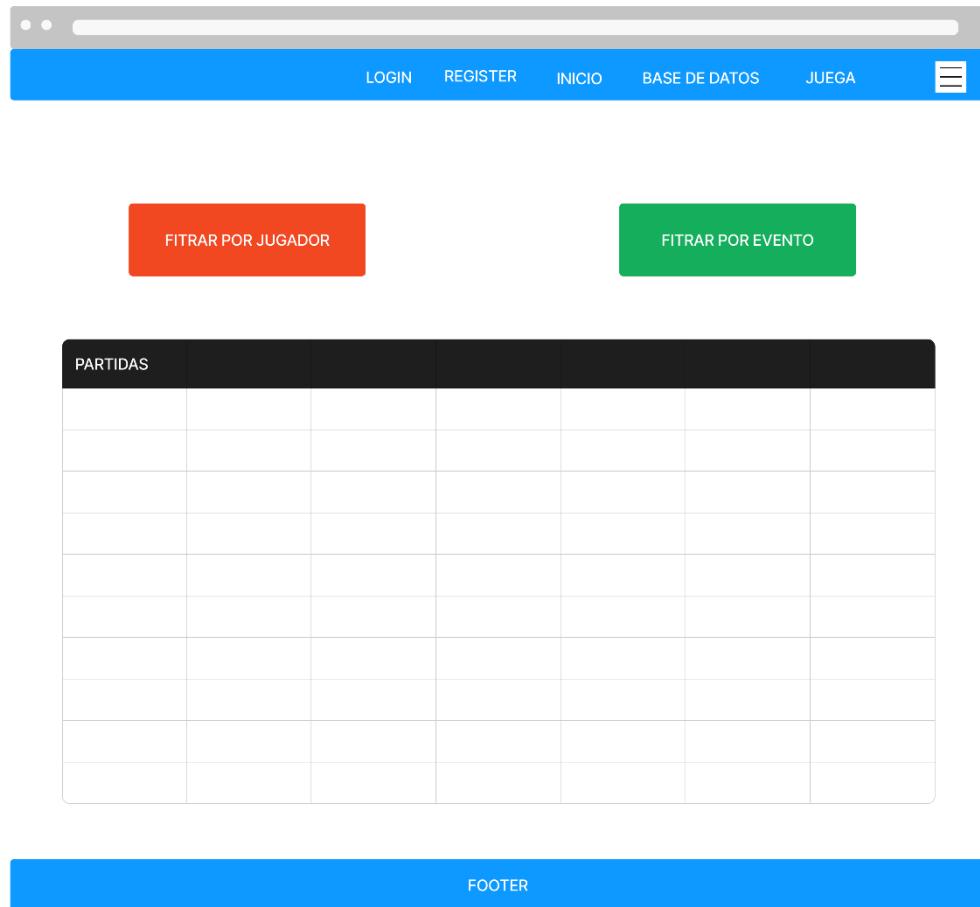


Figure 150: Menacho, E.

IU-0002: Base de datos					
Descripción	Muestra la interfaz con la base de datos				
Campos	Nombre	Tipo Datos	Editable/ Consulta	Oblig.	Descripción
	Partidas	Cadena	Consulta	Si	Muestra la información necesaria a mostrar para las partidas
Botones /Enlaces	Nombre		Acción		
	Botón Filtrar por Jugador		Permite filtrar la tabla por los eventos disponibles.		
	Botón Filtrar por Evento		Permite filtrar la tabla por los jugadores disponibles.		
	Botón Analizar		Permite analizar cada partida en la tabla.		

5.2.3 Interfaz del módulo IU-0003 – Analizar partida

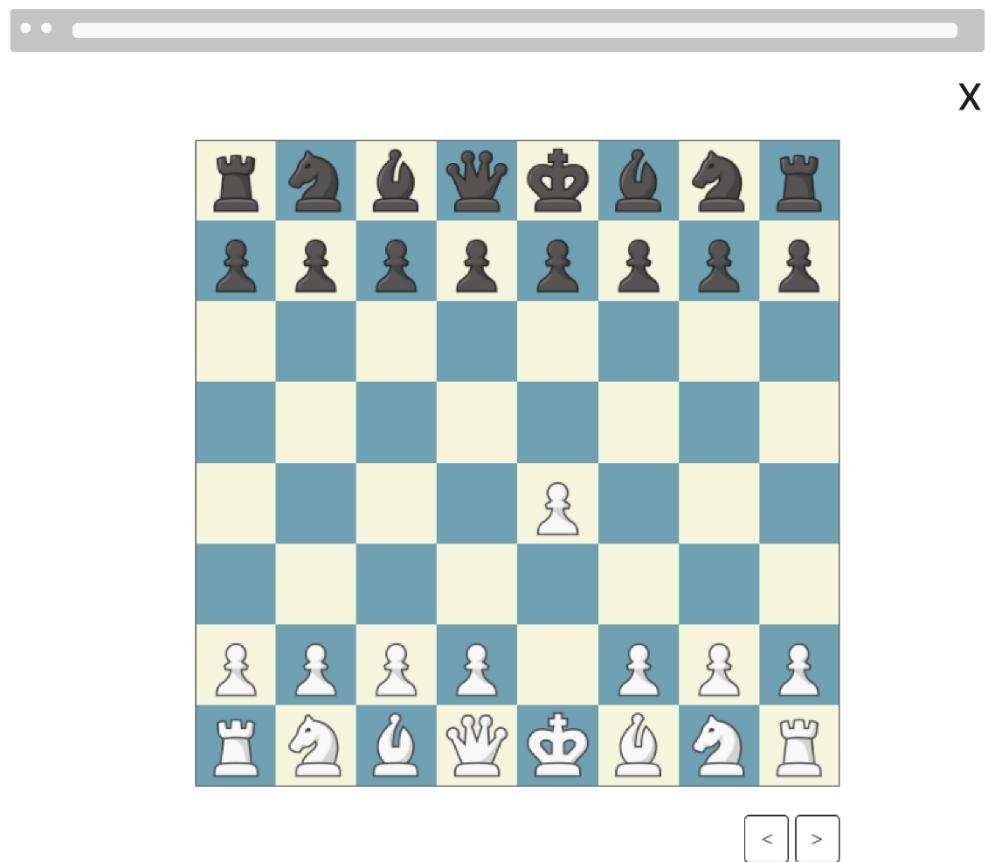


Figure 151: Menacho, E.

IU-0003: Analizar partida					
Descripción	Muestra la interfaz cuando analizamos una partida de la base de datos.				
Campos	Nombre	Tipo Datos	Editable/ Consulta	Oblig.	Descripción
Botones /Enlaces	Nombre		Acción		
	Botón Retroceder movimiento		Retrocede un movimiento según la notación de la partida.		
	Botón Adelantar movimiento		Avanza un movimiento según la notación de la partida.		
	Botón cerrar		Cierra la interfaz con la partida seleccionada.		

5.2.4 Interfaz del módulo IU-0004 – Gestión de usuarios y partidas

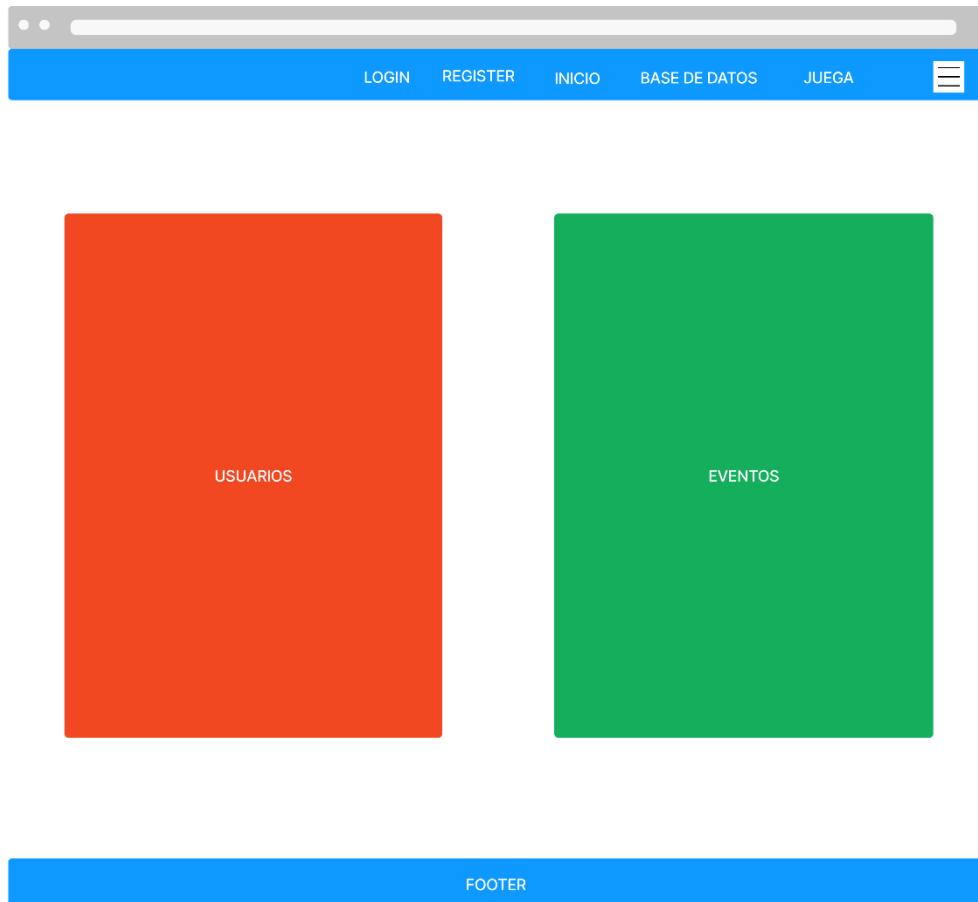


Figure 152: Menacho, E.

IU-0004: Gestión de usuarios y partidas					
Descripción	Muestra la interfaz para los administradores cuando quieren acceder al portal de administrador.				
Campos	Nombre	Tipo Datos	Editable/ Consulta	Oblig.	Descripción
Botones /Enlaces	Nombre		Acción		
	Botón Usuarios		Muestra la interfaz IU-0005		
	Botón Eventos		Muestra la interfaz IU-0007		

5.2.5 Interfaz del módulo IU-0005 – Gestión de usuarios

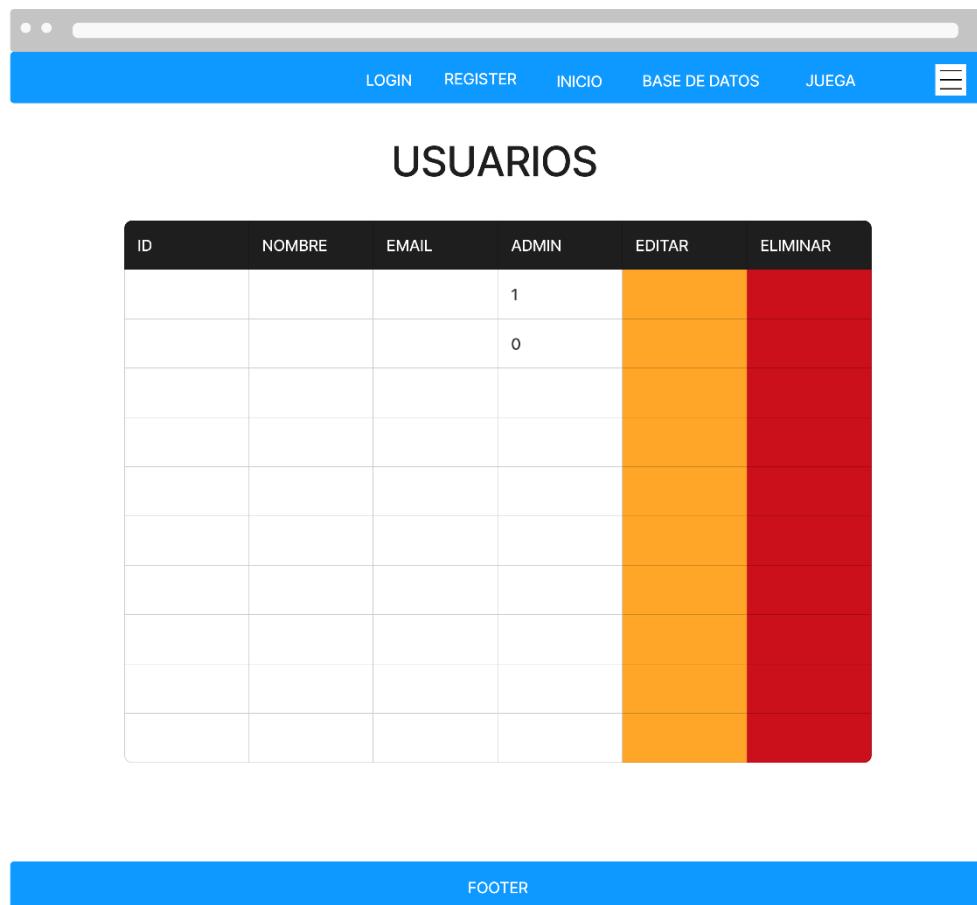


Figure 153: Menacho, E.

IU-0005: Gestión de usuarios					
Descripción	Muestra la interfaz para gestionar los usuarios de la aplicación.				
Campos	Nombre	Tipo Datos	Editable/Consulta	Oblig.	Descripción
	Usuarios	Cadena	Editable	Si	Recibe una lista con los usuarios de la aplicación.
Botones /Enlaces	Nombre		Acción		
	Botón Editar		Muestra la interfaz IU-0006.		
	Botón Eliminar		Elimina a un usuario de la aplicación.		

5.2.6 Interfaz del módulo IU-0006 – Editar usuario

The screenshot shows a web application interface. At the top, there is a blue header bar with navigation links: LOGIN, REGISTER, INICIO, BASE DE DATOS, JUEGA, and a menu icon (three horizontal lines). Below the header is a dark header section labeled 'EDITAR USUARIO'. The main content area contains three input fields: 'NAME', 'EMAIL', and 'ADMIN'. The 'ADMIN' field has a checked checkbox with an 'X' inside. Below the input fields is a red 'SUBMIT' button. At the bottom of the page is a blue footer bar labeled 'FOOTER'.

Figure 154: Menacho, E.

IU-0006: Editar Usuario					
Descripción	Muestra la interfaz para editar un usuario.				
Campos	Nombre	Tipo Datos	Editable/Consulta	Oblig.	Descripción
	Nombre	Cadena	Editable	Si	Nombre del usuario.
	Email	Cadena	Editable	Si	Correo del usuario.
	Admin	Boolean	Editable	Si	Variable que define si un usuario es administrador o no.
Botones /Enlaces	Nombre		Acción		
	Botón Submit		Envía el formulario llenado y edita el usuario.		

5.2.7 Interfaz del módulo IU-0007 – Gestionar partidas



Figure 155: Menacho, E.

IU-0007: Gestionar partidas					
Descripción	Muestra la interfaz para gestionar las partidas de la aplicación.				
Campos	Nombre	Tipo Datos	Editable/ Consulta	Oblig.	Descripción
	Partidas	Cadena	Editable	Si	Recibe una lista con las partidas de la aplicación.
Botones /Enlaces	Nombre		Acción		
	Botón Eliminar		Elimina una partida de la aplicación.		

5.2.8 Interfaz del módulo IU-0008 – Login

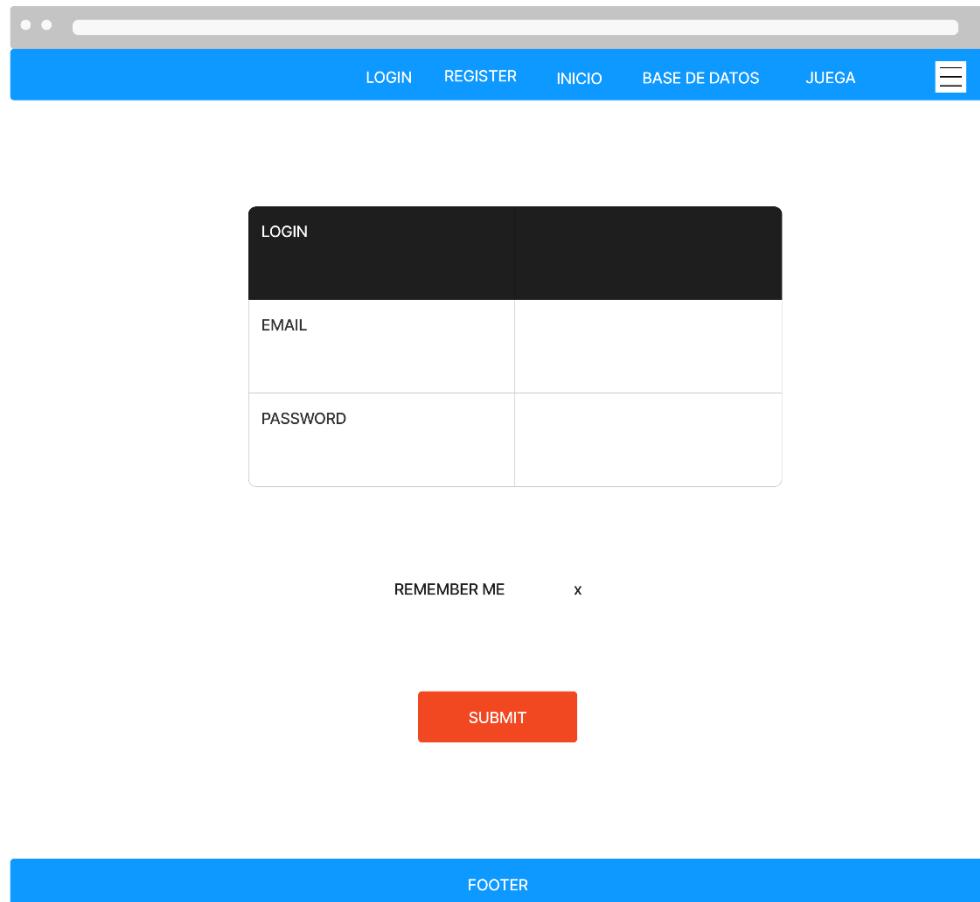


Figure 156: Menacho, E.

IU-0008: Login					
Descripción	Muestra la interfaz para gestionar las partidas de la aplicación.				
Campos	Nombre	Tipo Datos	Editable/ Consulta	Oblig. . .	Descripción
	Email	Cadena	Consulta	Si	Usuario que quiere iniciar sesión.
	Password	Cadena	Consulta	Si	Contraseña del usuario que quiere iniciar sesión.
	Remember me	Boolean	Editable	No	Recuerda la sesión del usuario si ha marcado la casilla
Botones /Enlaces	Nombre		Acción		
	Botón Submit		Envía el formulario.		

5.2.9 Interfaz del módulo IU-0009 – Register

The wireframe illustrates the 'REGISTER' interface. At the top, there's a navigation bar with links for LOGIN, REGISTER, INICIO, BASE DE DATOS, JUEGA, and a menu icon. The main area is titled 'REGISTER' and contains four input fields: 'NAME', 'EMAIL', 'PASSWORD', and 'CONFIRM PASSWORD'. Below these fields is a large blue 'SUBMIT' button. At the bottom of the page is a blue footer bar with the word 'FOOTER'.

Figure 157: Menacho, E.

IU-0009: Register					
Descripción	Muestra la interfaz para gestionar las partidas de la aplicación.				
Campos	Nombre	Tipo Datos	Editable/ Consulta	Oblig . .	Descripción
	Email	Cadena	Editable	Si	Correo del usuario que quiere registrarse.
	Password	Cadena	Editable	Si	Contraseña del usuario que quiere registrarse.
	Confirm Password	Cadena	Editable	Si	Confirmación de la contraseña antes del registro.
	Name	Cadena	Editable	Si	Nombre del usuario.
Botones /Enlaces	Nombre		Acción		
	Botón Submit		Envía el formulario.		

DESARROLLO PÁGINA WEB

DESARROLLO ALPHAUPO WEB

“ALPHAUPO”

1. TECNOLOGÍAS SELECCIONADAS

A continuación, mostraremos las tecnologías que se han utilizado para el desarrollo de la aplicación web.

1.1 Laravel:

Laravel es un popular marco de trabajo para desarrollar aplicaciones web en PHP, el cual ofrece una variedad de ventajas y características que lo hacen una opción atractiva para construir aplicaciones web modernas y escalables. Las principales razones para considerar Laravel como un buen marco de trabajo son:

- Sintaxis fácil de entender: Laravel tiene una sintaxis clara y expresiva que hace que el código sea fácil de entender y escribir, lo que acelera el proceso de desarrollo.
- Programación orientada a objetos: Laravel se basa en la programación orientada a objetos (POO), lo que facilita la reutilización del código y hace que la aplicación sea más modular y fácil de mantener.
- Patrón Modelo-Vista-Controlador (MVC): Laravel sigue el patrón MVC, lo que permite separar la lógica de negocio de la presentación y facilita la escalabilidad y mantenimiento del código.
- Enrutamiento fácil: Laravel tiene un sistema de enrutamiento fácil de usar que permite definir rutas de forma clara y sencilla.
- ORM Eloquent: Laravel utiliza Eloquent, un ORM intuitivo y potente que permite interactuar con bases de datos de forma más fácil y eficiente.
- Seguridad integrada: Laravel cuenta con medidas de seguridad integradas para proteger la aplicación de ataques comunes como la inyección SQL y la falsificación de solicitudes.

- Comunidad activa: Laravel cuenta con una comunidad activa de desarrolladores que contribuyen a mejorar el marco de trabajo y ofrecen soporte en línea.

9.

En resumen, Laravel es un marco de trabajo bien estructurado y fácil de aprender con numerosas características que lo hacen una excelente opción para construir aplicaciones web escalables, seguras y de alta calidad.

En concreto estaremos trabajando con la versión “Laravel Framework 8.83.7” con una versión de “PHP 8.1.6”. Para instalar PHP instalaremos el siguiente software.

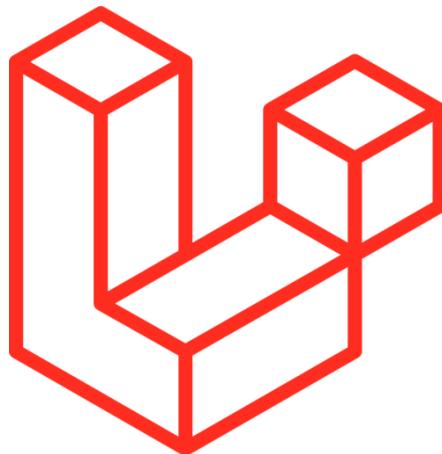


Figure 158: Laravel Icon

1.2 XAMPP:

Usaremos XAMPP para trabajar con una base de datos local y poder probar el funcionamiento de nuestra aplicación en local.

XAMPP es un paquete de software gratuito y de código abierto que contiene todas las herramientas necesarias para ejecutar un servidor web local en Windows, Linux o Mac,

incluyendo Apache, MySQL y PHP. Viene con una interfaz gráfica de usuario fácil de usar y herramientas útiles como phpMyAdmin. Además, XAMPP es portátil. En resumen, XAMPP es una solución completa y fácil de usar para el desarrollo web local.

En concreto usaremos la versión 3.3.0 de XAMPP.



Figure 159: Xampp Icon

1.3 Visual Studio Code:

Visual Studio Code es una excelente alternativa como editor de código debido a su potencia, flexibilidad y facilidad de uso. Algunas de las razones por las que es conveniente para desarrollar la web son las siguientes:

- Soporte para múltiples lenguajes de programación: Visual Studio Code es capaz de manejar una amplia gama de lenguajes de programación, incluyendo PHP, JavaScript, HTML y CSS, lo que lo hace ideal para proyectos de Laravel que requieren trabajar con múltiples archivos y lenguajes de programación.

- Extensiones para Laravel: Visual Studio Code cuenta con una amplia gama de extensiones que facilitan el trabajo con Laravel. Estas extensiones incluyen Laravel Blade Snippets, Laravel Extension Pack y muchas más. Estas extensiones ayudan a mejorar la productividad al ofrecer características específicas para Laravel, como resaltado de sintaxis, autocompletado, sugerencias y validaciones.



Figure 160: Visual Studio Code

2. DISEÑO DEL MOTOR DE AJEDREZ

Para diseñar el motor de ajedrez de la página web de ajedrez hemos utilizado como lenguaje de programación Javascript, ya que, al tratarse de un algoritmo de ajedrez, queremos que se ejecute desde la parte del cliente y no del servidor, ya que tener un servidor procesando las peticiones de los clientes es inviable. Luego es mucho más eficiente que el motor se ejecute desde la parte del usuario.

Para ello, lo primero fue implementar un motor capaz de realizar movimientos legales, revertirlos e ir hacia adelante en la partida. Para esto, se creó un paquete de funciones llamado Alphaupo.js que contiene todas las funcionalidades del tablero para poder jugar una partida y utilizar los algoritmos a posteriori.

Este paquete usará el siguiente modelo de dominio:

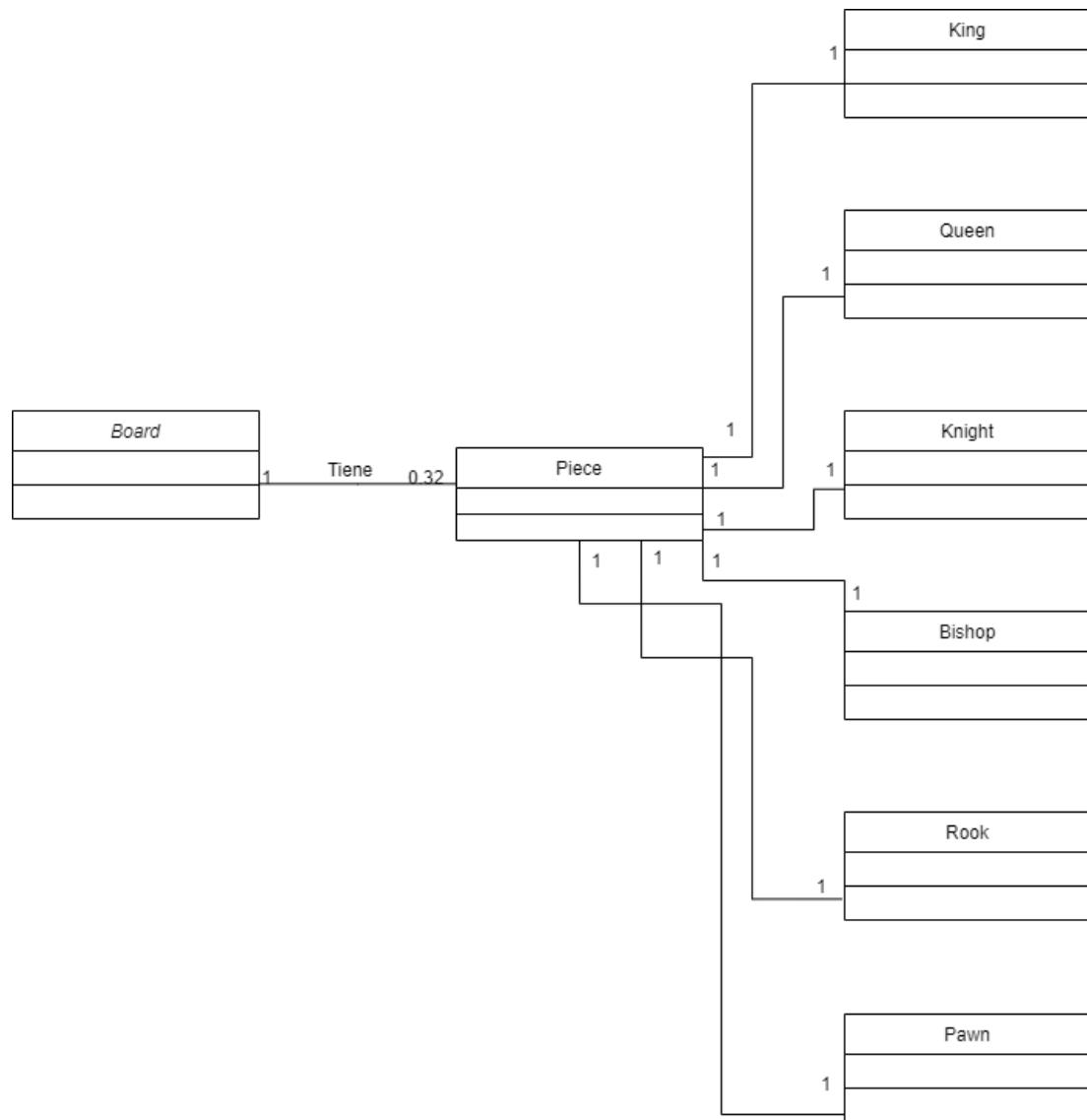


Figure 161: Menacho, E.

El resultado de codificar el juego de ajedrez fue el siguiente:

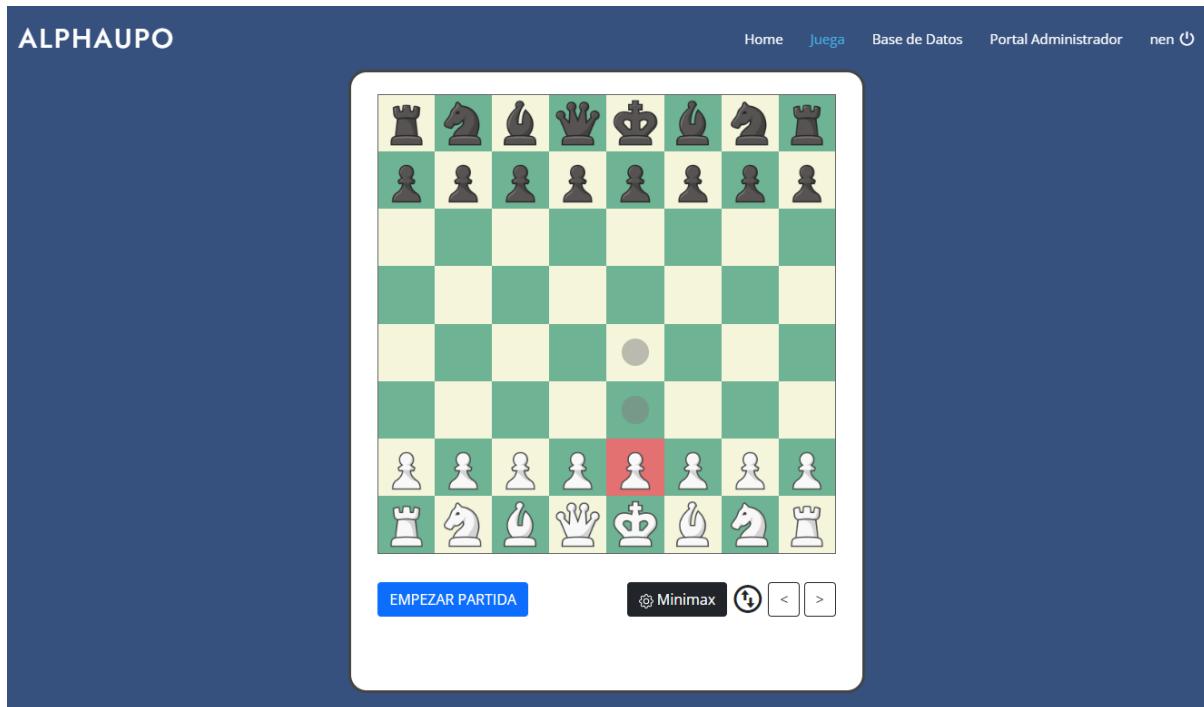


Figure 162: Menacho, E.

Una interfaz sencilla y simple que nos permite jugar contra los algoritmos que hemos hablado con anterioridad en la investigación. Ahora podemos crear una partida y reconocer los movimientos legales, los jaques, los jaques mate, etc... Es decir, todas las reglas del ajedrez.

3. DISEÑO DE LOS ALGORITMOS DE AJEDREZ

Para implementar los algoritmos de ajedrez contra los que podemos jugar usaremos la información extraída en la investigación e implementaremos el algoritmo de Minimax y Minimax con Alpha y Beta. Para esto, usaremos como tablero para guardar los estados la implementación anterior contenida en “Alphaupo.js”.

Los algoritmos contra los que podemos jugar en la aplicación son los siguientes:

Minimax:

```
function minimax(board, depth) {
    if (depth === 0 || board.is_checkmate() || board.is_stalemate()) {
        return [this.fitness(board), null];
    } else {
        var toMove = board.toMove;
        var moves = board.getLegalMoves(board.toMove);

        //Maximizamos las blancas
        if (toMove === 0) {
            var bestScore = -9999999999;
            var bestMove = false;

            for (var i = 0; i < moves.length; i++) {
                board.makeMove(moves[i]);
                var res = minimax(board, depth - 1);
                board.revertLastMove(false);

                if (res[0] > bestScore) {
                    bestScore = res[0];
                    bestMove = moves[i];
                } else if(res[0] == bestScore){
                    if(Math.random()>0.5){
                        bestScore = res[0];
                        bestMove = moves[i];
                    }
                }
            }
            return [bestScore, bestMove];
        } else {
            var bestScore = 9999999999;
            var bestMove = false;

            for (var i = 0; i < moves.length; i++) {
                board.makeMove(moves[i]);
                var res = minimax(board, depth - 1);
                board.revertLastMove(false);

                if (res[0] < bestScore) {
                    bestScore = res[0];
                    bestMove = moves[i];
                } else if(res[0] == bestScore){
                    if(Math.random()>0.5){
                        bestScore = res[0];
                        bestMove = moves[i];
                    }
                }
            }
            return [bestScore, bestMove];
        }
    }
}
```

Figure 163: Menacho, E.

Para la implementación de Minimax hemos utilizado como base el algoritmo previamente implementado con Python, pero en este caso haremos que seleccione un movimiento al azar para el mejor movimiento en caso de que los evalúe con la misma puntuación.

Como heurística usaremos el valor de las piezas mencionado anteriormente junto con la posibilidad de reconocer un jaque mate en la posición.

```
function fitness(board) {
    var res = board.material_value();

    if (board.is_checkmate()) {
        if (board.toMove === 1) {
            return 999999;
        } else {
            return -999999;
        }
    }

    return res;
}
```

Figure 164: Menacho, E.

Minimax Alpha Beta:

```

function minimaxAlphaBetaPruning(board, depth, alpha, beta) {
    if (depth === 0 || board.is_checkmate() || board.is_stalemate()) {
        return [this.fitness(board), null];
    } else {
        var toMove = board.toMove;
        var moves = board.getLegalMoves(board.toMove);
        //Maximizamos las blancas
        if (toMove === 0) {
            var bestMove = false;
            var bestVal = -99999999;

            for (var i = 0; i < moves.length; i++) {
                board.makeMove(moves[i]);
                var res = minimaxAlphaBetaPruning(board, depth - 1, alpha, beta);
                board.revertLastMove(false);

                if (bestVal < res[0]) {
                    bestMove = moves[i];
                    bestVal = res[0];
                }
            }

            alpha = Math.max(alpha, bestVal);

            if (alpha >= beta) {
                return [bestVal, bestMove];
            }
        }
        return [bestVal, bestMove];
    } else {
        var bestMove = false;
        var bestVal = 99999999;

        for (var i = 0; i < moves.length; i++) {
            board.makeMove(moves[i]);
            var res = minimaxAlphaBetaPruning(board, depth - 1, alpha, beta);
            board.revertLastMove(false);

            if (bestVal > res[0]) {
                bestMove = moves[i];
                bestVal = res[0];
            }
        }

        beta = Math.min(beta, bestVal);

        if (beta<=alpha) {
            return [bestVal, bestMove];
        }
    }
    return [bestVal, bestMove];
}
}

```

Figure 165: Menacho, E.

Para la implementación de los algoritmos con ponderación Alpha y beta hemos utilizado el mismo código que el mencionado anteriormente en Python y como heurística utilizamos la misma que para el Minimax.

Minimax Alpha Beta + Database:

Aprovechando la base de datos de grandes maestros hemos creado nuestro propio libro de aperturas, de forma que nuestro algoritmo realiza movimientos hasta no encontrar más partidas en la base de datos. A la hora de seleccionar los movimientos selecciona las aperturas con mayor posibilidad de victoria según la base de datos, para ello hemos creado dos funciones en el controlador de las partidas:

```

public function getNextDatabaseMove($game){

    $aux = array("White" => DB::select("
        SELECT LEFT(SUBSTRING(Game,LENGTH('%'.$game.'%')), InStr(SUBSTRING(Game,LENGTH(
            '%'.$game.'%'),' ') - 1) as n_move, count(*) as number FROM `partida` 
        WHERE Game LIKE '%'.$game.'%' AND Result = '1-0'
        GROUP BY n_move
        ORDER BY number DESC LIMIT 3;
    "));

    $aux2 = array("Black" => DB::select("
        SELECT LEFT(SUBSTRING(Game,LENGTH('%'.$game.'%')), InStr(SUBSTRING(Game,LENGTH(
            '%'.$game.'%'),' ') - 1) as n_move, count(*) as number FROM `partida` 
        WHERE Game LIKE '%'.$game.'%' AND Result = '0-1'
        GROUP BY n_move
        ORDER BY number DESC LIMIT 3;
    "));
    echo json_encode(array_merge($aux, $aux2));
}

public function firstDatabaseMove(){

    $aux = array("White" => DB::select("
        SELECT REPLACE(LEFT(SUBSTRING(Game,LENGTH('%%')), InStr(SUBSTRING(Game,
            LENGTH('%%')), ' ') - 1), '.', '') as n_move, count(*) as number FROM `partida` 
        WHERE Game LIKE '%%' AND Result = '1-0'
        GROUP BY n_move
        ORDER BY number DESC LIMIT 3;
    "));

    $aux2 = array("Black" => DB::select("
        SELECT REPLACE(LEFT(SUBSTRING(Game,LENGTH('%%')), InStr(SUBSTRING(Game,
            LENGTH('%%')), ' ') - 1), '.', '') as n_move, count(*) as number FROM `partida` 
        WHERE Game LIKE '%%' AND Result = '0-1'
        GROUP BY n_move
        ORDER BY number DESC LIMIT 3;
    "));
    echo json_encode(array_merge($aux, $aux2));
}

```

Figure 166: Menacho, E.

En la segunda función hacemos una consulta para obtener el primer movimiento de la partida, en la primera hacemos la misma consulta, pero para un movimiento que no sea el primero. Para ello seleccionamos las victorias con las blancas y las victorias con las negras en función de la notación del transcurso de la partida.

Minimax Alpha y Beta 2:

Para este motor se utilizará el algoritmo de Minimax Alpha y Beta implementado anteriormente, a excepción de que usaremos una heurística diferente. En esta, ponderaremos el valor de las casillas con el objetivo de que el algoritmo realice aperturas más coherentes y sea capaz de ganar espacio en el tablero, para ello ponderaremos el tablero de la siguiente forma para cada pieza:

Peones Blancos:

```
var wp = [
  [0., 0., 0., 0., 0., 0., 0., 0.],
  [5., 5., 5., 5., 5., 5., 5., 5.],
  [1., 1., 2., 3., 3., 2., 1., 1.],
  [0.5, 0.5, 1., 2.5, 2.5, 1., 0.5, 0.5],
  [0., 0., 0., 2., 2., 0., 0., 0.],
  [0.5, -0.5, -1., 0., 0., -1., -0.5, 0.5],
  [0.5, 1., 1., -2., -2., 1., 1., 0.5],
  [0., 0., 0., 0., 0., 0., 0., 0.]
];
```

Figure 167: Menacho, E.

Caballos Blancos:

```
var wn = [
  [-5., -4., -3., -3., -3., -3., -4., -5.],
  [-4., -2., 0., 0., 0., -2., -4.],
  [-3., 0., 1., 1.5, 1.5, 1., 0., -3.],
  [-3., 0.5, 1.5, 2., 2., 1.5, 0.5, -3.],
  [-3., 0., 1.5, 2., 2., 1.5, 0., -3.],
  [-3., 0.5, 1., 1.5, 1.5, 1., 0.5, -3.],
  [-4., -2., 0., 0.5, 0.5, 0., -2., -4.],
  [-5., -4., -3., -3., -3., -3., -4., -5.]
];
```

Figure 168: Menacho, E.

Alfiles Blancos:

```
var wb = [
    [-2., -1., -1., -1., -1., -1., -1., -2.],
    [-1., 0., 0., 0., 0., 0., 0., -1.],
    [-1., 0., 0.5, 1., 1., 0.5, 0., -1.],
    [-1., 0.5, 0.5, 1., 1., 0.5, 0.5, -1.],
    [-1., 0., 1., 1., 1., 0., -1.],
    [-1., 1., 1., 1., 1., 1., -1.],
    [-1., 0.5, 0., 0., 0., 0., 0.5, -1.],
    [-2., -1., -1., -1., -1., -1., -1., -2.]
];
```

Figure 169: Menacho, E.

Torres Blancas:

```
var wr = [
    [0., 0., 0., 0., 0., 0., 0., 0.],
    [0.5, 1., 1., 1., 1., 1., 1., 0.5],
    [-1., 0., 0.5, 1., 1., 0.5, 0., -0.5],
    [-1., 0.5, 0.5, 1., 1., 0.5, 0.5, -0.5],
    [-1., 0., 1., 1., 1., 1., 0., -0.5],
    [-1., 1., 1., 1., 1., 1., 1., -0.5],
    [-1., 0.5, 0., 0., 0., 0., 0.5, -0.5],
    [0., 0., 0., 0.5, 0.5, 0., 0., 0.]
];
```

Figure 170: Menacho, E.

Reinas Blancas:

```
var wq = [
    [-2, -1, -1, -0.5, -0.5, -1, -1, -2],
    [-1, 0., 0., 0., 0., 0., 0., -1.],
    [-1, 0., 0.5, 0.5, 0.5, 0.5, 0., -1.],
    [-0.5, 0., 0.5, 0.5, 0.5, 0.5, 0., -0.5],
    [0., 0., 0.5, 0.5, 0.5, 0.5, 0., -0.5],
    [-1, 0.5, 0.5, 0.5, 0.5, 0.5, 0., -1.],
    [-1, 0., 0.5, 0., 0., 0., 0., -1.],
    [-2, -1, -1, -0.5, -0.5, -1, -1, -2]
];
```

Figure 171: Menacho, E.

Rey Blanco:

```
var wk = [
    [-3, -4, -4, -5, -5, -4, -4, -3],
    [-3, -4, -4, -5, -5, -4, -4, -3],
    [-3, -4, -4, -5, -5, -4, -4, -3],
    [-3, -4, -4, -5, -5, -4, -4, -3],
    [-2, -3, -3, -4, -4, -3, -3, -2],
    [-1., -2, -2, -2, -2, -2, -2, -1],
    [2, 2, 0., 0., 0., 2, 2],
    [2, 3, 1, 0, 0, 1, 3, 2]
];
```

Figure 172: Menacho, E.

Para las piezas negras utilizaremos las matrices de posición de las piezas blancas y las rotaremos utilizando la siguiente función:

```
function negativeRotateArray(arr) {
    var res = arr.slice();
    var tam = arr.length;

    for (var i = 0; i < tam; i++) {
        for (var j = 0; j < tam; j++) {
            res[i][tam - 1 - j] = arr[tam - 1 - i][j] * (-1);
        }
    }
    return res;
}
```

Figure 173: Menacho, E.

El objetivo de estas matrices es evaluar la posición de una manera más acertada dependiendo de la posición de cada pieza en el tablero, a continuación, mostraremos la heurística al completo:

```

function fitness2(board) {
    var res = board.material_value();

    if (board.is_checkmate()) {
        if (board.toMove === 1) {
            return 999999;
        } else {
            return -999999;
        }
    }

    for (var i = 0; i < board.pieces.length; i++) {
        var pc = board.pieces[i];
        if (pc.name === "wp") {
            res += wp[pc.pos[0]][pc.pos[1]];
        } else if (pc.name === "bp") {
            res += bp[pc.pos[0]][pc.pos[1]];
        } else if (pc.name === "wn") {
            res += wn[pc.pos[0]][pc.pos[1]];
        } else if (pc.name === "bn") {
            res += bn[pc.pos[0]][pc.pos[1]];
        } else if (pc.name === "wb") {
            res += wb[pc.pos[0]][pc.pos[1]];
        } else if (pc.name === "bb") {
            res += bb[pc.pos[0]][pc.pos[1]];
        } else if (pc.name === "wr") {
            res += wr[pc.pos[0]][pc.pos[1]];
        } else if (pc.name === "br") {
            res += br[pc.pos[0]][pc.pos[1]];
        } else if (pc.name === "wq") {
            res += wq[pc.pos[0]][pc.pos[1]];
        } else if (pc.name === "bq") {
            res += bq[pc.pos[0]][pc.pos[1]];
        } else if (pc.name === "wk") {
            res += wk[pc.pos[0]][pc.pos[1]];
        } else if (pc.name === "bk") {
            res += bk[pc.pos[0]][pc.pos[1]];
        }
    }

    return res;
}

```

Figure 174: Menacho, E.

Hemos utilizado la heurística anterior solo que esta vez calculamos el valor de la posición de cada pieza en el tablero.

Tras la codificación de los algoritmos nos dimos cuenta de que en Javascript, al ser un lenguaje que cuenta con un único hilo de ejecución era imposible interactuar con la página mientras calculaba el algoritmo el siguiente movimiento, es por esto que se propuso realizar las implementaciones fijando una promesa al algoritmo, para ello utilizamos los “Workers” o trabajadores en Español.

Los workers se ejecutan en un entorno separado y no tienen acceso directo al hilo principal de la aplicación ni a las variables o funciones definidas allí. En su lugar, los workers se comunican con el hilo principal mediante una promesa.

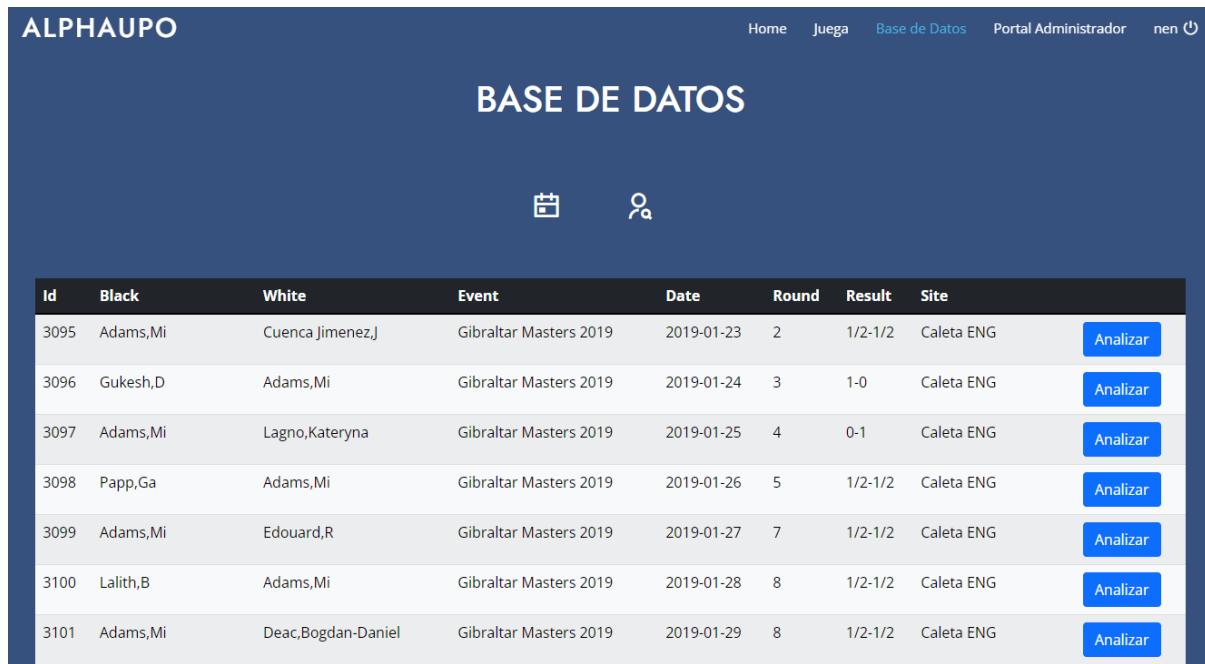
Para ello enviamos una promesa al “Worker” y este calcula la solución del algoritmo mientras que intercala el algoritmo para poder seguir ejecutando la página.



Figure 175: Web Workers

4. INTERFAZ BASE DE DATOS

Para la Base de Datos hemos optado por una interfaz sencilla que nos permite realizar los filtrados de las partidas por eventos y por jugador seleccionado.



The screenshot shows a dark-themed web application for managing a chess database. At the top, there's a navigation bar with links: Home, Juega, Base de Datos (which is highlighted in blue), Portal Administrador, and a user icon. Below the navigation, the title "BASE DE DATOS" is centered. Underneath the title are two small icons: a white square with a black pawn and a white circle with a black pawn. The main content is a table listing chess games:

Id	Black	White	Event	Date	Round	Result	Site	Analizar
3095	Adams,Mi	Cuenca Jimenez,J	Gibraltar Masters 2019	2019-01-23	2	1/2-1/2	Caleta ENG	Analizar
3096	Gukesh,D	Adams,Mi	Gibraltar Masters 2019	2019-01-24	3	1-0	Caleta ENG	Analizar
3097	Adams,Mi	Lagno,Kateryna	Gibraltar Masters 2019	2019-01-25	4	0-1	Caleta ENG	Analizar
3098	Papp,Ga	Adams,Mi	Gibraltar Masters 2019	2019-01-26	5	1/2-1/2	Caleta ENG	Analizar
3099	Adams,Mi	Edouard,R	Gibraltar Masters 2019	2019-01-27	7	1/2-1/2	Caleta ENG	Analizar
3100	Lalith,B	Adams,Mi	Gibraltar Masters 2019	2019-01-28	8	1/2-1/2	Caleta ENG	Analizar
3101	Adams,Mi	Deac,Bogdan-Daniel	Gibraltar Masters 2019	2019-01-29	8	1/2-1/2	Caleta ENG	Analizar

Figure 176: Menacho, E.

También contará con una interfaz capaz de analizar las partidas y reproducirlas:



Figure 177: Menacho, E.

Para reproducir las partidas utilizamos el motor de ajedrez implementado anteriormente para jugar contra los algoritmos. Para evaluar las partidas, hemos exportado el modelo que hacía uso de las redes convolucionales a Javascript, para ello hemos utilizado TensorFlow.js, un paquete que nos permite ejecutar nuestros modelos previamente implementados con Python y exportados. En este caso la red convolucional recibe una posición de ajedrez y clasifica si las negras ganan, si la posición está empatada o si la posición es victoriosa para las piezas blancas.



Figure 178: TensorflowJS.

5. GESTIÓN DE USUARIOS

Id		Name	Email	Admin		
2		user	user@itsolutionstuff.com	x	Eliminar	Editar
6		Pepe	pepe@mail.com	x	Eliminar	Editar
7		admin	admin@itsolutionstuff.com	✓	Eliminar	Editar
9		nen	nenitomg@gmail.com	✓	Eliminar	Editar
10		Tensifusion	tensifusion@gmail.com	x	Eliminar	Editar

Figure 178: Menacho, E.

6. GESTIÓN DE PARTIDAS

Id		Black	White	Event	Date	Round	Result	Site	
3095	Adams,Mi	Cuenca Jimenez,J	Gibraltar Masters 2019	2019-01-23	2		1/2-1/2	Caleta ENG	Eliminar
3096	Gukesh,D	Adams,Mi	Gibraltar Masters 2019	2019-01-24	3		1-0	Caleta ENG	Eliminar
3097	Adams,Mi	Lagno,Kateryna	Gibraltar Masters 2019	2019-01-25	4		0-1	Caleta ENG	Eliminar
3098	Papp,Ga	Adams,Mi	Gibraltar Masters 2019	2019-01-26	5		1/2-1/2	Caleta ENG	Eliminar
3099	Adams,Mi	Edouard,R	Gibraltar Masters 2019	2019-01-27	7		1/2-1/2	Caleta ENG	Eliminar
3100	Lalith,B	Adams,Mi	Gibraltar Masters 2019	2019-01-28	8		1/2-1/2	Caleta ENG	Eliminar
3101	Adams,Mi	Deac,Bogdan-Daniel	Gibraltar Masters 2019	2019-01-29	8		1/2-1/2	Caleta ENG	Eliminar

Figure 179: Menacho, E.

MANUAL DE USUARIO

MANUAL DE USUARIO

“ALPHAUPO”

1. GOOGLE COLABORATORY

Para el uso de los códigos proporcionados durante la investigación, hemos decidido usar Jupyter Notebook para poder crear un cuaderno y ejecutarlo a posteriori. Podemos encontrar todos los cuadernos implementados desde el siguiente enlace:

<https://github.com/nenomg/ALPHAUPO-INVESTIGATION>

Los cuadernos implementados son los siguientes:

1. Minimax: Contiene los algoritmos de Minimax y Minimax Alpha y Beta codificados en Python. <https://github.com/nenomg/ALPHAUPO-INVESTIGATION/releases/download/ALPHAUPO-INVESTIGATION/Minimax.rar>
2. Turochamp: Contiene la heurística de Alan Turin para evaluar una posición codificada en Python: <https://github.com/nenomg/ALPHAUPO-INVESTIGATION/releases/download/ALPHAUPO-INVESTIGATION/Turochamp.rar>
3. Modelo Jaques: Contiene las redes convolucionales para la clasificación de jaques sin conocimiento previo del juego. <https://github.com/nenomg/ALPHAUPO-INVESTIGATION/releases/download/ALPHAUPO-INVESTIGATION/Modelo.Jaques.rar>
4. Modelo Jaques Mate: Contiene las redes convolucionales para la clasificación de jaques mate sin conocimiento previo del juego. <https://github.com/nenomg/ALPHAUPO-INVESTIGATION/releases/download/ALPHAUPO-INVESTIGATION/Modelo.Jaques.Mate.rar>
5. Modelo heurística: Contiene la red convolucional que aprende del motor Stockfish 15 para predicción de evaluaciones en una posición. <https://github.com/nenomg/ALPHAUPO-INVESTIGATION/releases/download/ALPHAUPO-INVESTIGATION/Modelo.Heuristica.rar>

INVESTIGATION/releases/download/ALPHAUPO-

INVESTIGATION/Modelo.Heuristica.rar

1.1 Instrucciones

Para ejecutar los cuadernos de Minimax y Turochamp debemos hacer lo siguiente:

- Subir el cuaderno a Google Drive:

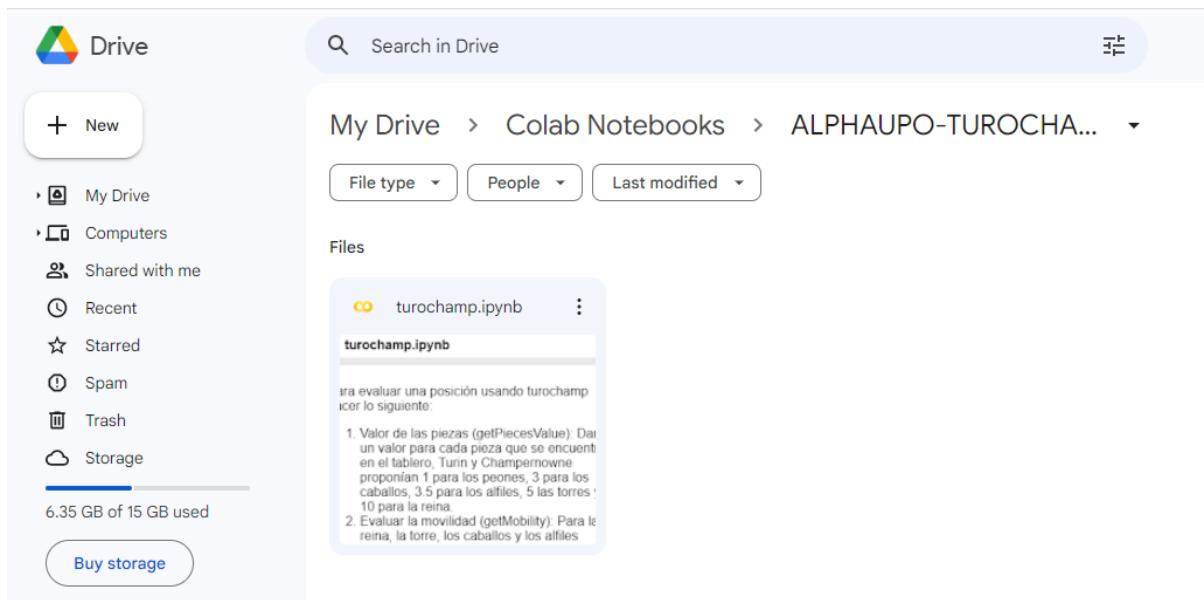


Figure 180: Menacho, E.

- Hacemos doble click y abrimos el entorno de Google Colaboratory:

The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** turochamp.ipynb
- Toolbar:** File, Edit, View, Insert, Runtime, Tools, Help, Last edited on May 7
- Code Cells:**
 - Cell 1:** Text: "Para evaluar una posición usando turochamp hacer lo siguiente:" followed by a numbered list from 1 to 8 describing evaluation rules.
 - Cell 2:** Code: `!pip install chess` followed by the output of the package installation command.
 - Cell 3:** Code: `import chess` followed by a chessboard visualization.
- Output:** A 8x8 chessboard diagram with pieces placed on it.

Figure 181: Menacho, E.

- Ejecutamos celda a celda el código:

The screenshot shows a Jupyter Notebook cell containing the command `!pip install chess` and its execution output, which includes the package download and installation process.

```

!pip install chess
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting chess
  Downloading chess-1.9.4-py3-none-any.whl (149 kB)
                                             149.1/149.1 kB 6.8 MB/s eta 0:00:00
Installing collected packages: chess
Successfully installed chess-1.9.4

```

Figure 182: Menacho, E.

Para el resto de cuadernos que dependen de ficheros para su ejecución haremos lo siguiente, primero subiremos el cuaderno y los ficheros como lo hacemos con los anteriores, luego una vez subido el cuaderno debemos hacer lo siguiente:

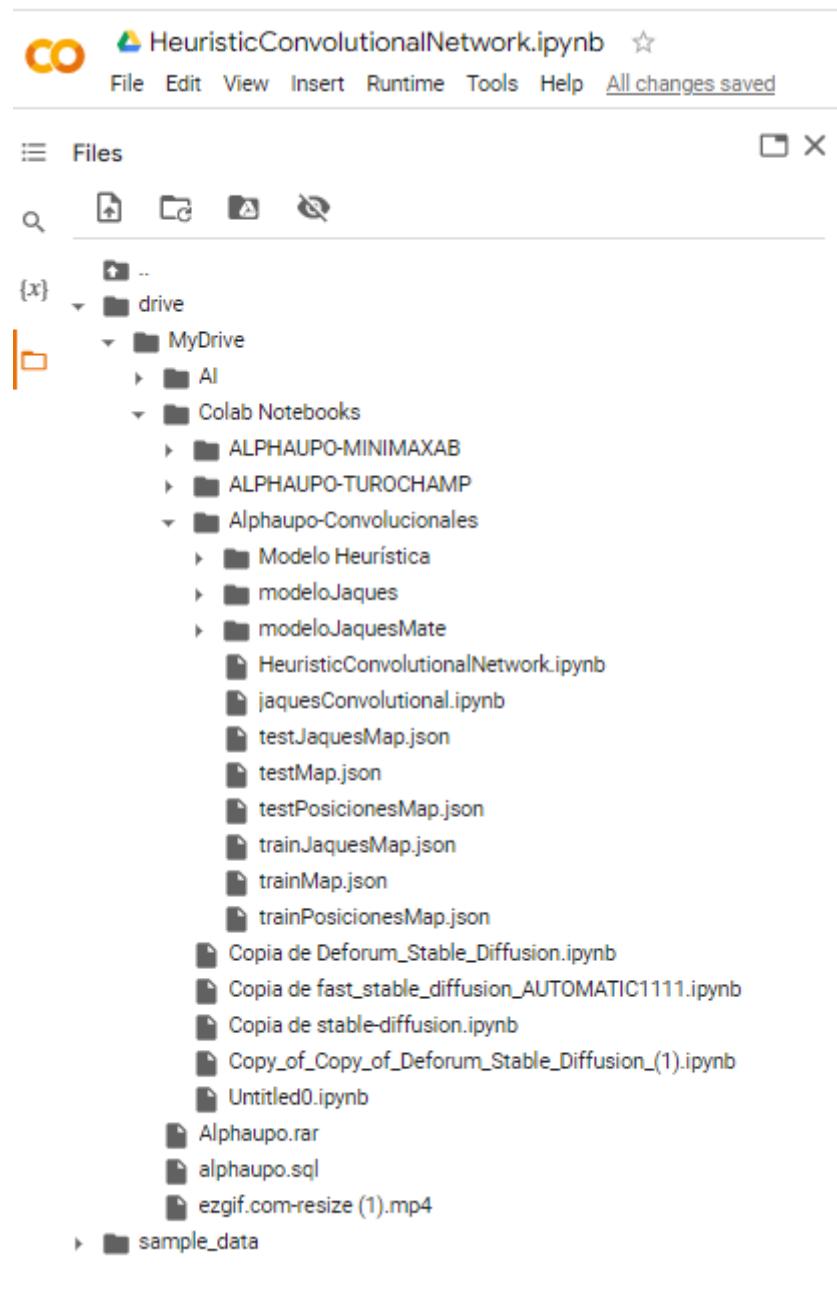


Figure 183: Menacho, E.

Vemos desde la parte izquierda de la aplicación los ficheros de nuestro drive, seleccionamos el archivo que queremos importar en el código, hacemos click derecho y seleccionamos copiar ruta y pegamos la dirección en la siguiente celda de código. También podríamos subir los archivos al entorno de ejecución y hacer las referencias desde ahí.

- Cargamos el dataset, en caso de no estar incluido, incluirlo en el entorno de ejecución

```
[ ] train = pd.read_json('drive/MyDrive/Colab Notebooks/Alphaupo-Convolucionales/trainPosicionesMap.json')
      test = pd.read_json('drive/MyDrive/Colab Notebooks/Alphaupo-Convolucionales/testPosicionesMap.json')
```

Figure 184: Menacho, E.

Por último, las redes convolucionales son muy costosas de entrenar, para ello recomendamos seleccionar entorno de ejecución, para ello vamos a Runtime → Change Runtime Type y seleccionamos la GPU en lugar de la CPU, de esta forma entrenaremos el modelo mucho más rápido.

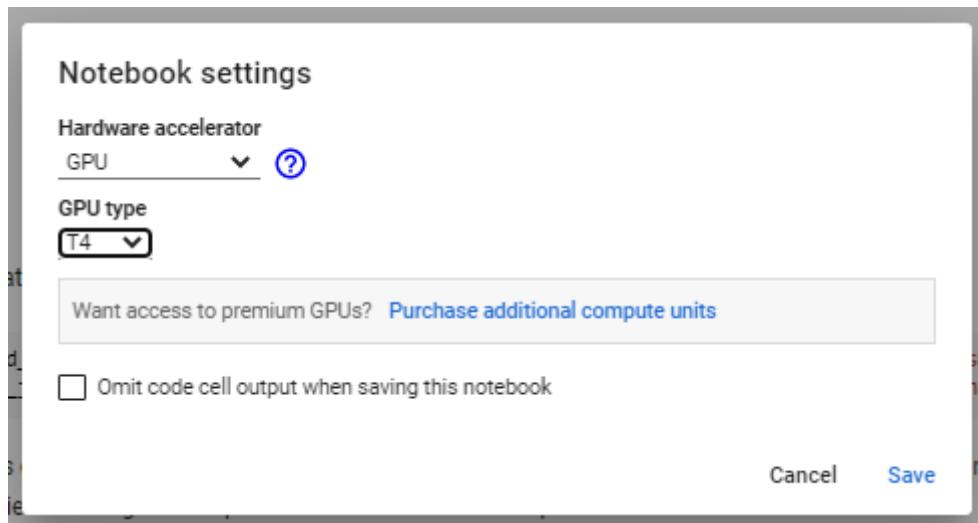


Figure 185: Menacho, E.

2. PÁGINA WEB EN LOCAL

Para instalar la página web en local debemos instalar una versión de PHP similar a la nuestra, en este caso estamos utilizando la versión siguiente:

```
C:\Users\nenit>php --version
PHP 8.1.6 (cli) (built: May 11 2022 08:55:59) (ZTS Visual C++ 2019 x64)
Copyright (c) The PHP Group
Zend Engine v4.1.6, Copyright (c) Zend Technologies
```

Figure 186: Menacho, E.

Para instalarla hemos utilizado XAMPP, en este caso hemos utilizado la versión 3.3.0 de XAMPP:

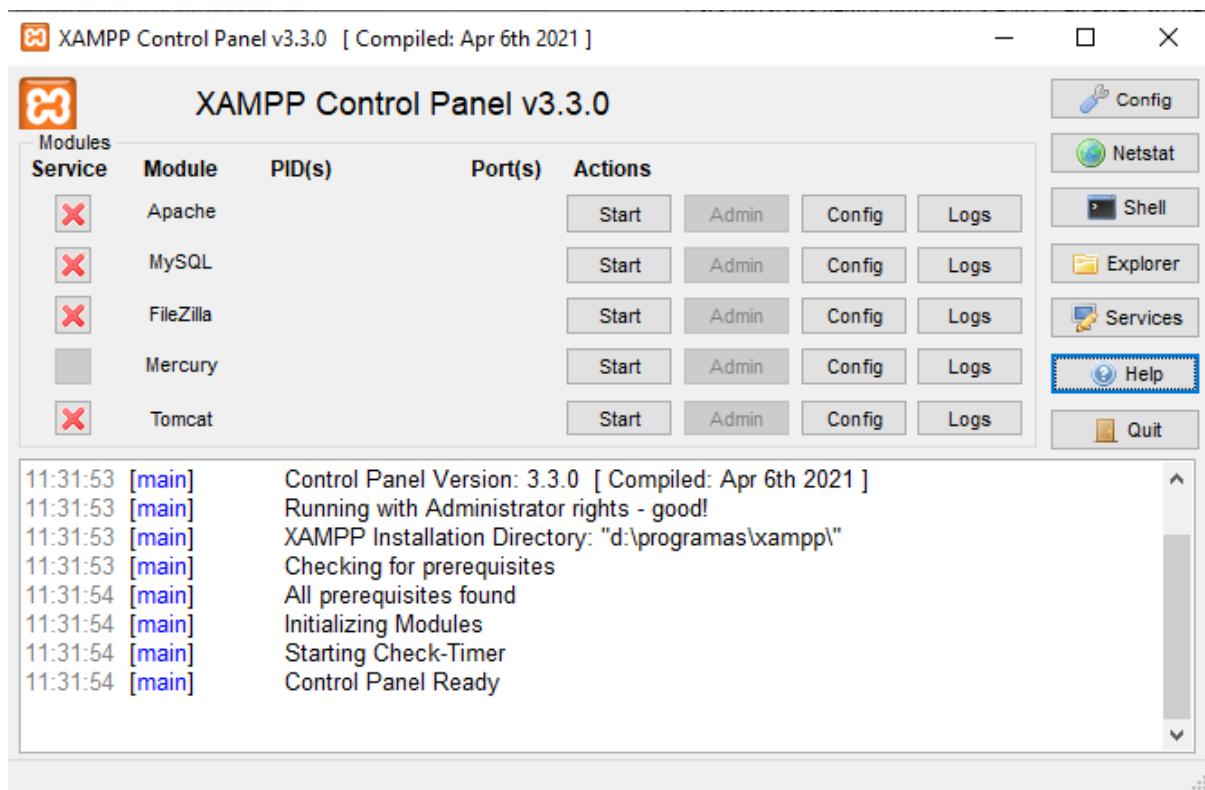


Figure 187: Menacho, E.

Una vez instalado XAMPP debemos importar la base de datos de la aplicación, para ello levantamos los servidores y vamos a phpmyadmin:

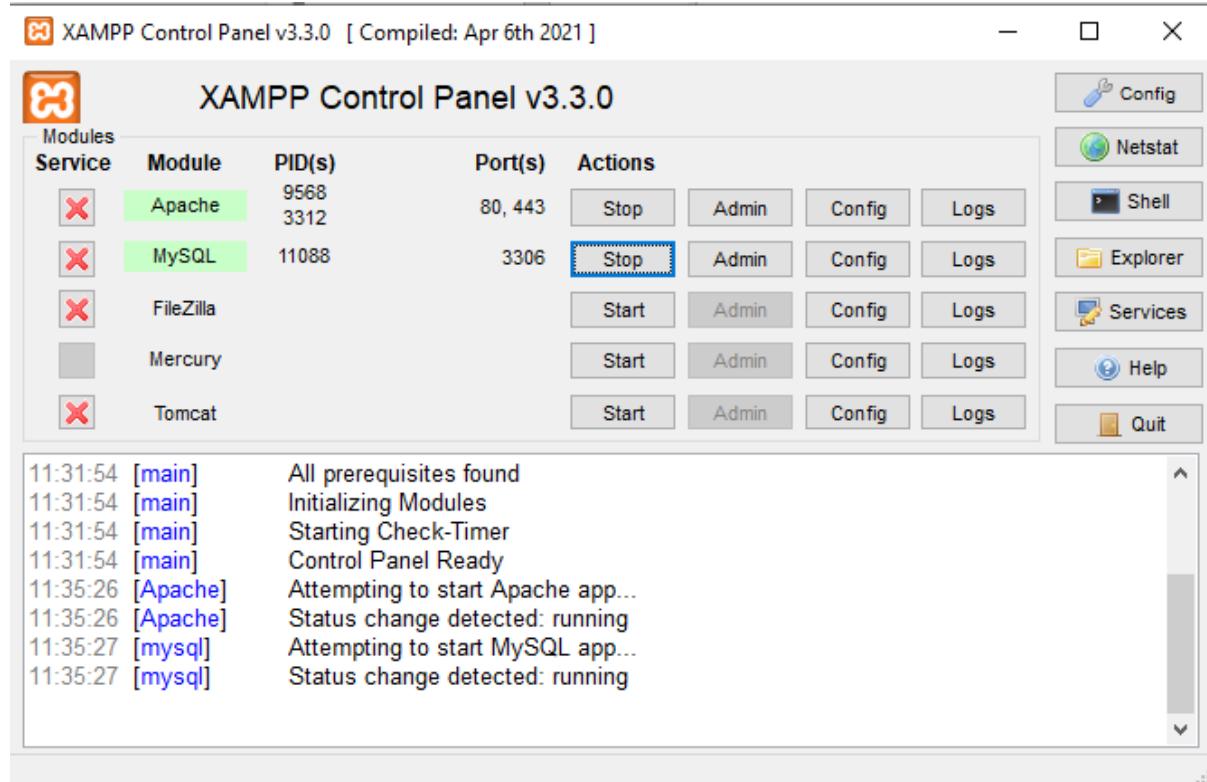


Figure 188: Menacho, E.

Una vez en phpmyadmin seleccionamos importar y importamos la base de datos correspondiente:

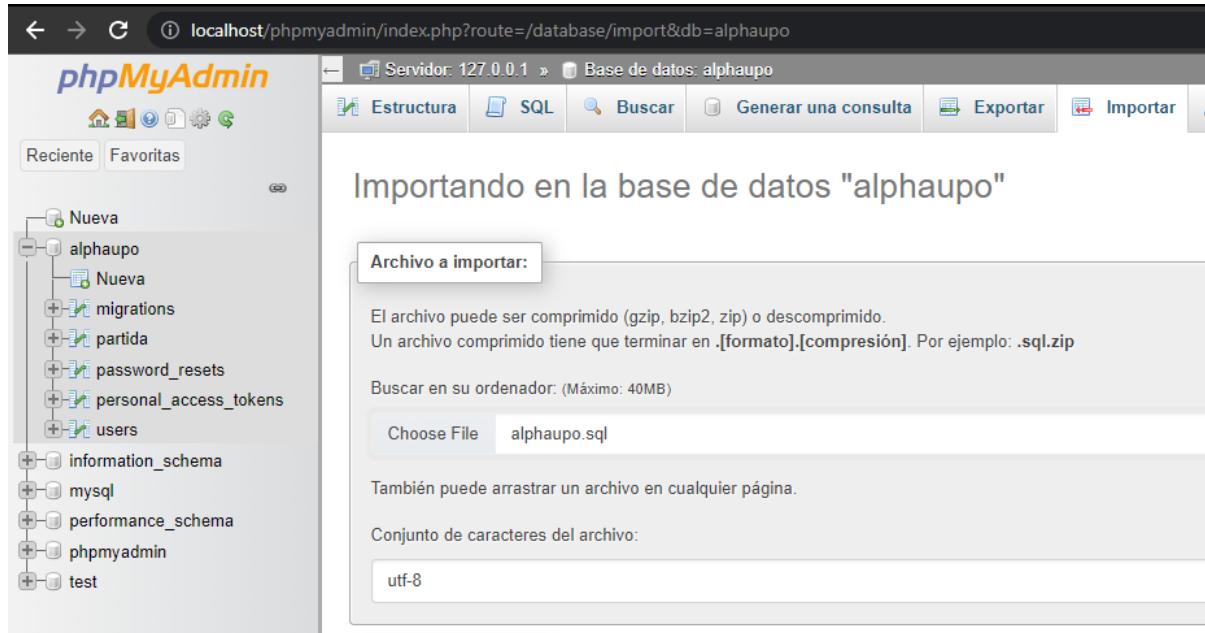


Figure 189: Menacho, E.

Por último, levantamos la aplicación, para ello vamos al directorio en el que se encuentran los archivos de la página y abrimos el terminal, luego ejecutamos el siguiente comando:

```
Microsoft Windows [Versión 10.0.19044.2965]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\nenit\Documents\Laravel\Alphaupo>php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
[Mon May 15 11:46:54 2023] PHP 8.1.6 Development Server (http://127.0.0.1:8000) started
```

Figure 190: Menacho, E.

Ya estaría levantada la página y lista para usarse.



Figure 191: Menacho, E.

3. FUNCIONALIDADES PÁGINA WEB

3.1 Portada



Figure 192: Menacho, E.

This screenshot shows a 'QUIENES SOMOS' (Who We Are) page. At the top, there are logos for 'Pablo Olavide', 'HTML CSS JS', 'Laravel', and 'python'. Below these, the title 'QUIENES SOMOS' is centered. A paragraph of text follows, stating: 'Alphaupo es un trabajo de investigación sobre la informática en el ajedrez, incluyendo varios puntos de vista para computar el ajedrez y nuevas implementaciones para otros aspectos del juego aplicando técnicas de inteligencia artificial.' Underneath this, there is a section titled 'Investigaciones' with a list of items, each preceded by a small video camera icon. The items are: 'Algoritmo de Minimax', 'Algoritmo de Minimax con ponderación alpha y beta', 'Modificaciones sobre Minimax', 'Turochamp, algoritmo de Alan Turin como heurística', 'Redes Convolucionales para la predicción de jaques y jaques mate', and 'Redes Convolucionales como heurística en el ajedrez'. At the bottom right of the page is a small circular arrow icon.

Figure 193: Menacho, E.



Figure 194: Menacho, E.

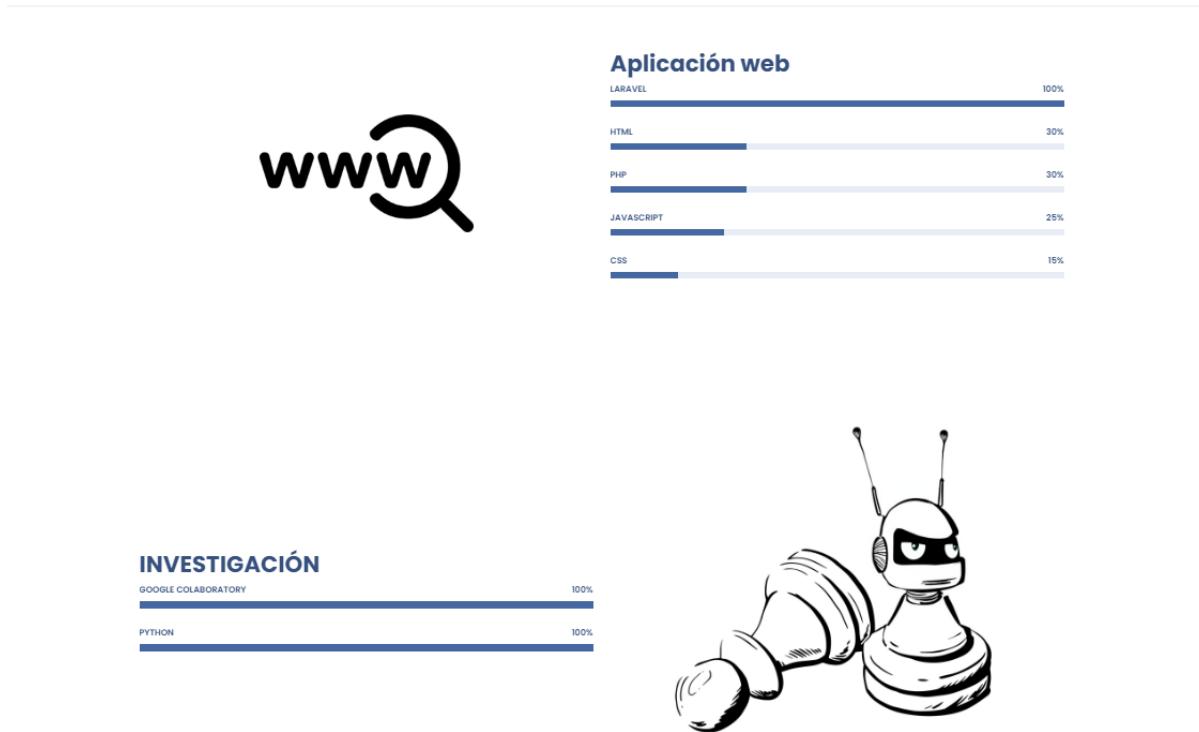


Figure 195: Menacho, E.

EQUIPO

Somos un equipo altamente colaborativo y comprometido, con habilidades complementarias y una pasión compartida por lograr los mejores resultados en todo lo que hacemos.



Eugenio Menacho de Góngora
Estudiante, desarrollador
Estudiante de ingeniería informática de sistemas de información en la universidad Pablo de Olavide, Escuela Politécnica Superior.



Francisco Martínez Álvarez
Catedrático de universidad, Tutor
Profesor en la universidad Pablo de Olavide, Escuela Politécnica Superior, Ingeniería informática de sistemas de información.



Alicia Troncoso Lora
Catedrática de universidad, Coordinadora
Profesora en la universidad Pablo de Olavide, Escuela Politécnica superior, Ingeniería informática de sistemas de información.

Figure 196: Menacho, E.

INVESTIGACIÓN

El resultado de las investigaciones hemos decidido plasmarlo en cuadernos de Jupyter con el objetivo de que cualquier persona pueda ejecutarlos y probar los códigos proporcionados, accede a la codificación desde aquí: <https://github.com/nenomg/ALPHAUPO-INVESTIGATION>

Investigaciones

- 🕒 Algoritmo de Minimax
- 🕒 Algoritmo de Minimax con ponderación alpha y beta
- 🕒 Modificaciones sobre Minimax
- 🕒 Turochamp, algoritmo de Alan Turing como heurística
- 🕒 Redes Convolucionales para la predicción de jaques y jaques mate
- 🕒 Redes Convolucionales como heurística en el ajedrez

ALPHAUPO

Ctra. de Utrera, 1, 41013
Sevilla

Email: emende@alu.upo.es

LINKS

- > Home
- > Juega
- > Base de Datos
- > Login
- > Register

SOCIAL

Échale un ojo a nuestros métodos de comunicación.



Figure 197: Menacho, E.

3.2 Login

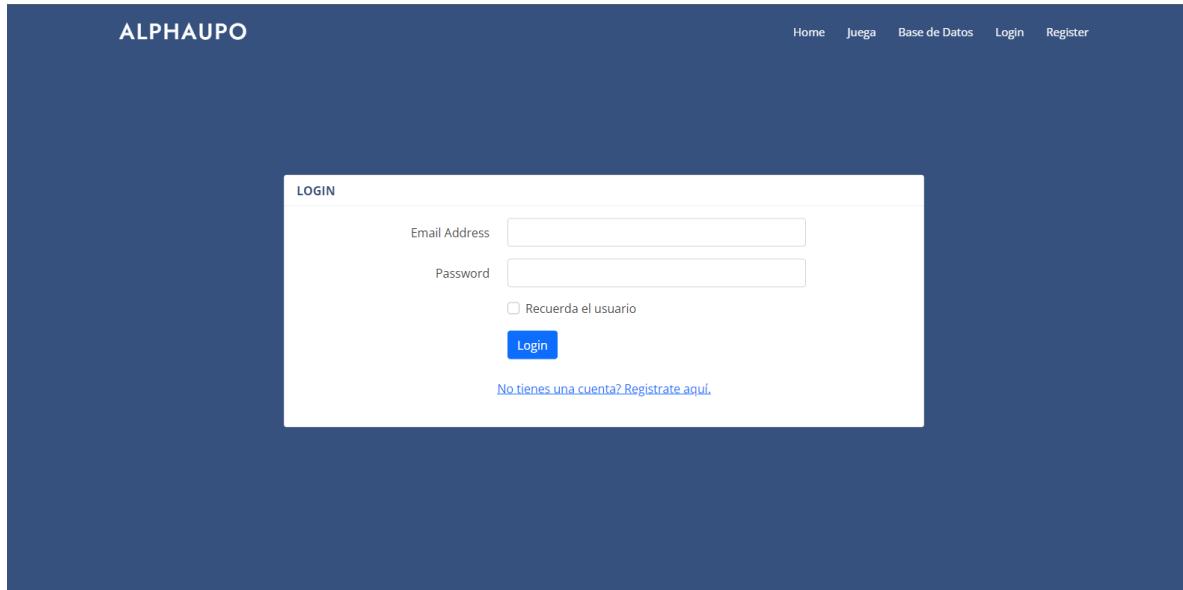


Figure 198: Menacho, E.

3.3 Register

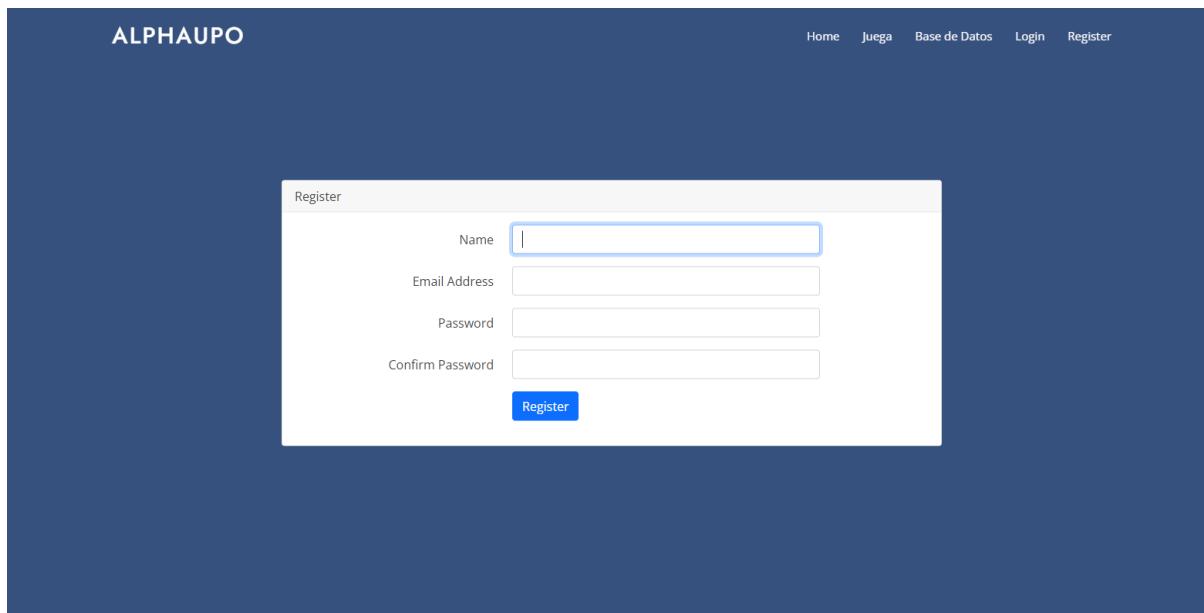


Figure 199: Menacho, E.

3.4 Logout



3.5 Juega

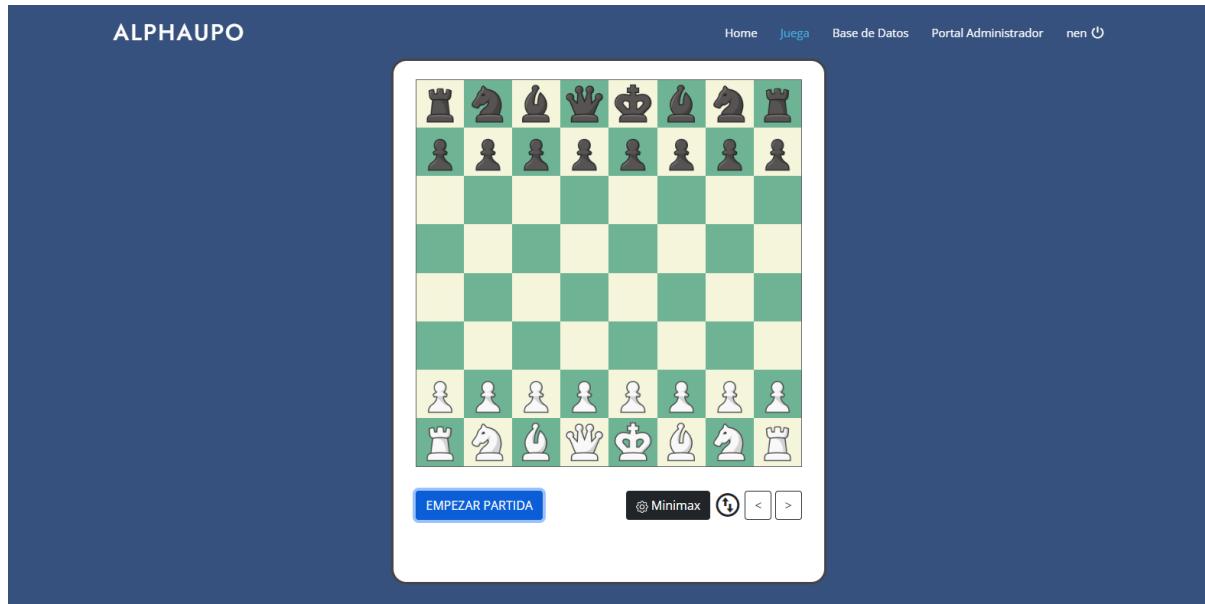


Figure 200: Menacho, E.

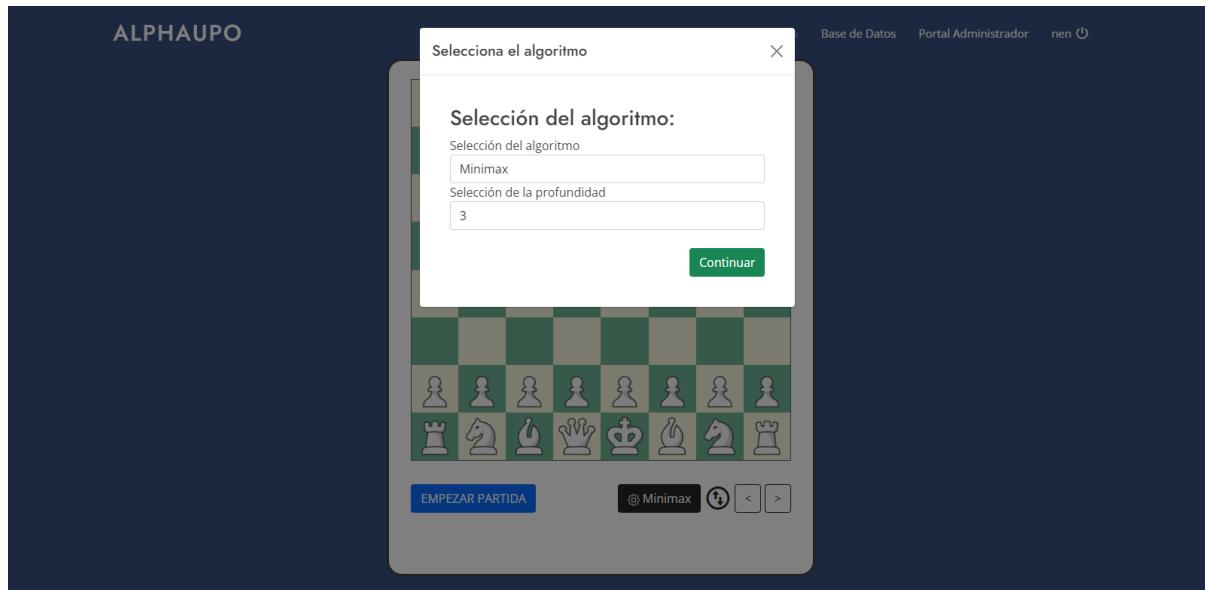


Figure 201: Menacho, E.

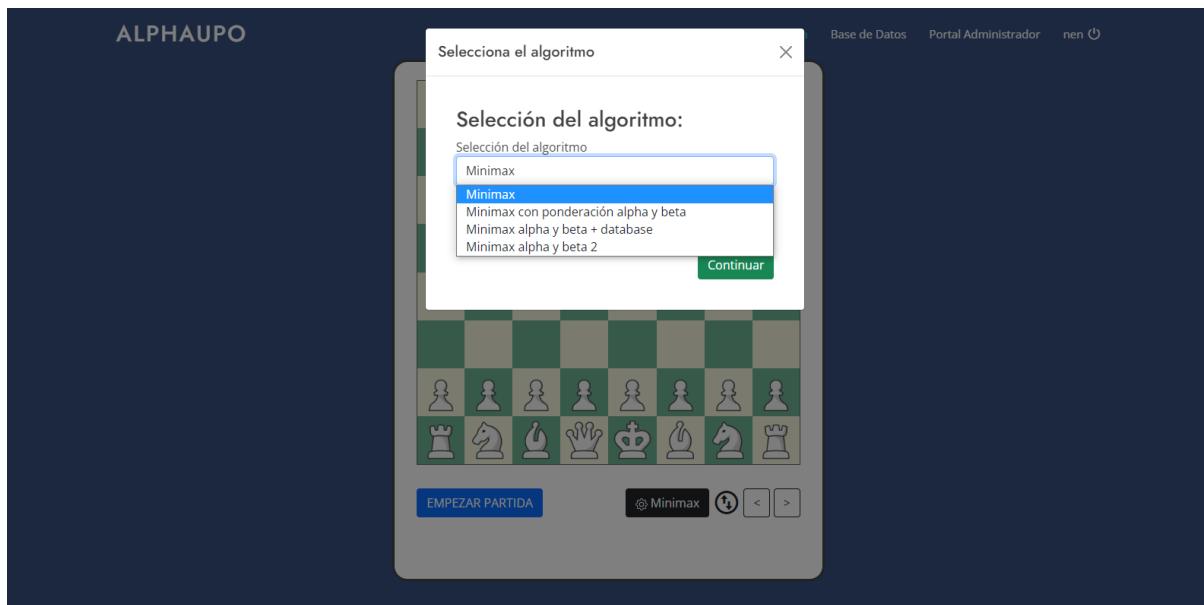


Figure 202: Menacho, E.

3.6 Base de Datos

The screenshot shows the ALPHAUPO chess application's Database page. The title "BASE DE DATOS" is centered at the top. Below it is a search bar with two input fields and a magnifying glass icon. A table lists eight chess games from the Gibraltar Masters 2019 tournament. The columns are: Id, Black, White, Event, Date, Round, Result, Site, and a blue "Analizar" (Analyze) button. The games are:

ID	Black	White	Event	Date	Round	Result	Site	Analizar
3095	Adams,Mi	Cuenca Jimenez,J	Gibraltar Masters 2019	2019-01-23	2	1/2-1/2	Caleta ENG	<button>Analizar</button>
3096	Gukesh,D	Adams,Mi	Gibraltar Masters 2019	2019-01-24	3	1-0	Caleta ENG	<button>Analizar</button>
3097	Adams,Mi	Lagno,Kateryna	Gibraltar Masters 2019	2019-01-25	4	0-1	Caleta ENG	<button>Analizar</button>
3098	Papp,Ga	Adams,Mi	Gibraltar Masters 2019	2019-01-26	5	1/2-1/2	Caleta ENG	<button>Analizar</button>
3099	Adams,Mi	Edouard,R	Gibraltar Masters 2019	2019-01-27	7	1/2-1/2	Caleta ENG	<button>Analizar</button>
3100	Lalith,B	Adams,Mi	Gibraltar Masters 2019	2019-01-28	8	1/2-1/2	Caleta ENG	<button>Analizar</button>
3101	Adams,Mi	Deac,Bogdan-Daniel	Gibraltar Masters 2019	2019-01-29	8	1/2-1/2	Caleta ENG	<button>Analizar</button>
3102	Svane,R	Adams,Mi	Gibraltar Masters 2019	2019-01-30	9	1-0	Caleta ENG	<button>Analizar</button>

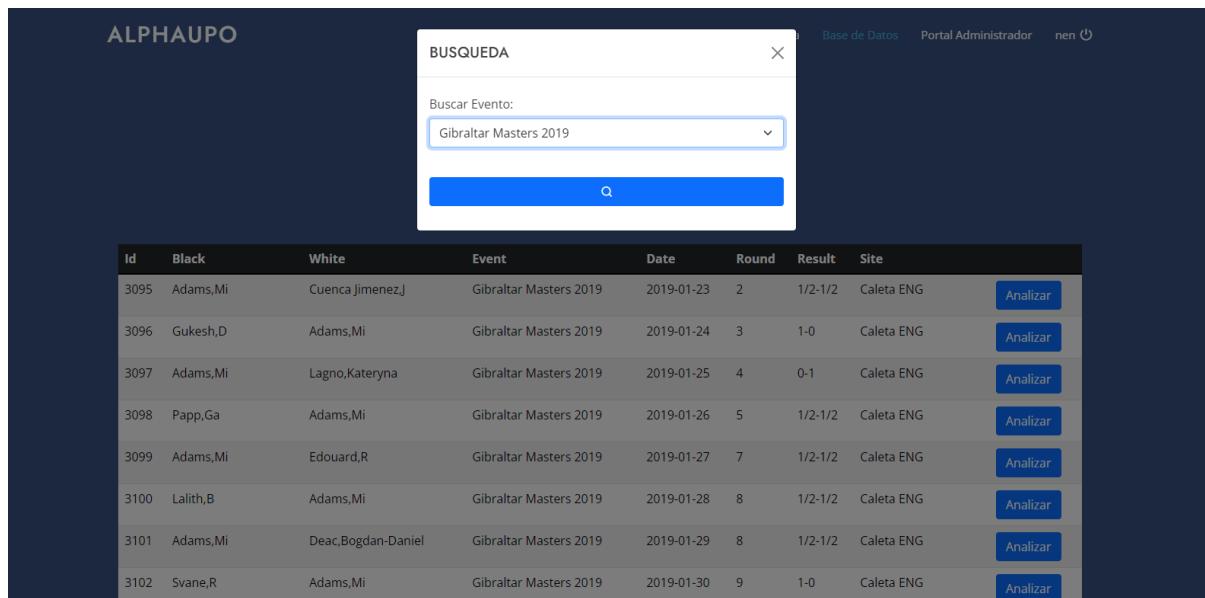
Figure 203: Menacho, E.



Figure 204: Menacho, E.



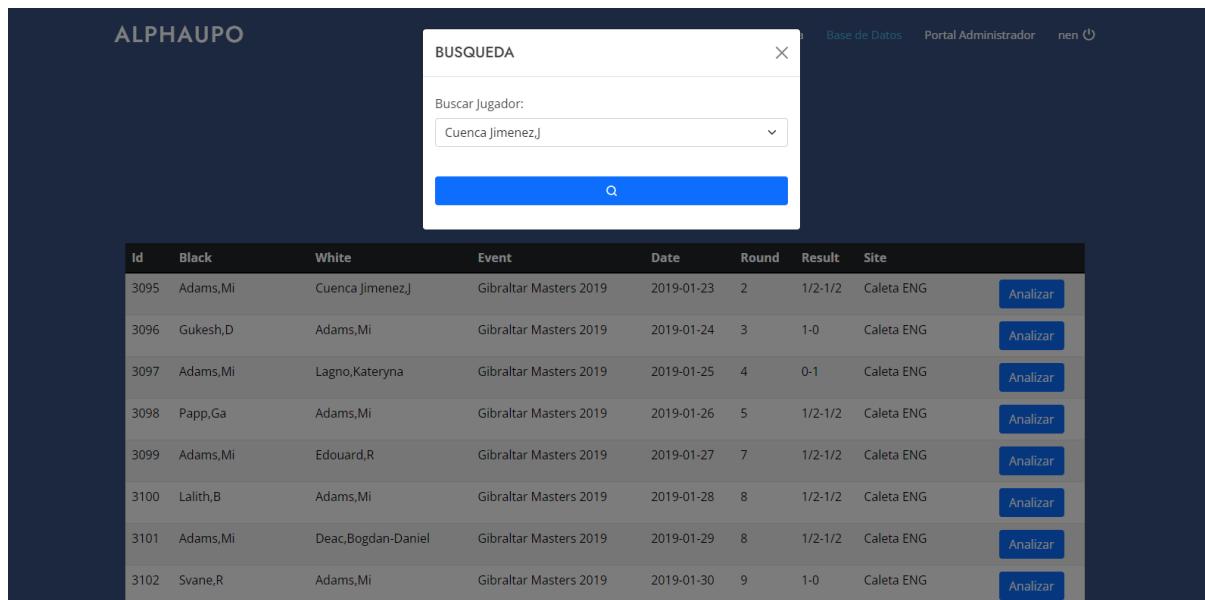
Figure 205: Menacho, E.



The screenshot shows a dark-themed web application. At the top left is the logo 'ALPHAUPO'. On the right side, there are links for 'Base de Datos', 'Portal Administrador', and a user icon. A search bar at the top center contains the text 'BUSQUEDA' and a dropdown menu set to 'Buscar Evento:' with the option 'Gibraltar Masters 2019' selected. Below the search bar is a large blue button with a white magnifying glass icon. The main area features a table with the following columns: Id, Black, White, Event, Date, Round, Result, Site, and a blue 'Analizar' button. The table contains 8 rows of data from the Gibraltar Masters 2019 event.

Id	Black	White	Event	Date	Round	Result	Site	
3095	Adams,Mi	Cuenca Jimenez,J	Gibraltar Masters 2019	2019-01-23	2	1/2-1/2	Caleta ENG	Analizar
3096	Gukesh,D	Adams,Mi	Gibraltar Masters 2019	2019-01-24	3	1-0	Caleta ENG	Analizar
3097	Adams,Mi	Lagno,Kateryna	Gibraltar Masters 2019	2019-01-25	4	0-1	Caleta ENG	Analizar
3098	Papp,Ga	Adams,Mi	Gibraltar Masters 2019	2019-01-26	5	1/2-1/2	Caleta ENG	Analizar
3099	Adams,Mi	Edouard,R	Gibraltar Masters 2019	2019-01-27	7	1/2-1/2	Caleta ENG	Analizar
3100	Lalith,B	Adams,Mi	Gibraltar Masters 2019	2019-01-28	8	1/2-1/2	Caleta ENG	Analizar
3101	Adams,Mi	Deac,Bogdan-Daniel	Gibraltar Masters 2019	2019-01-29	8	1/2-1/2	Caleta ENG	Analizar
3102	Svane,R	Adams,Mi	Gibraltar Masters 2019	2019-01-30	9	1-0	Caleta ENG	Analizar

Figure 206: Menacho, E.

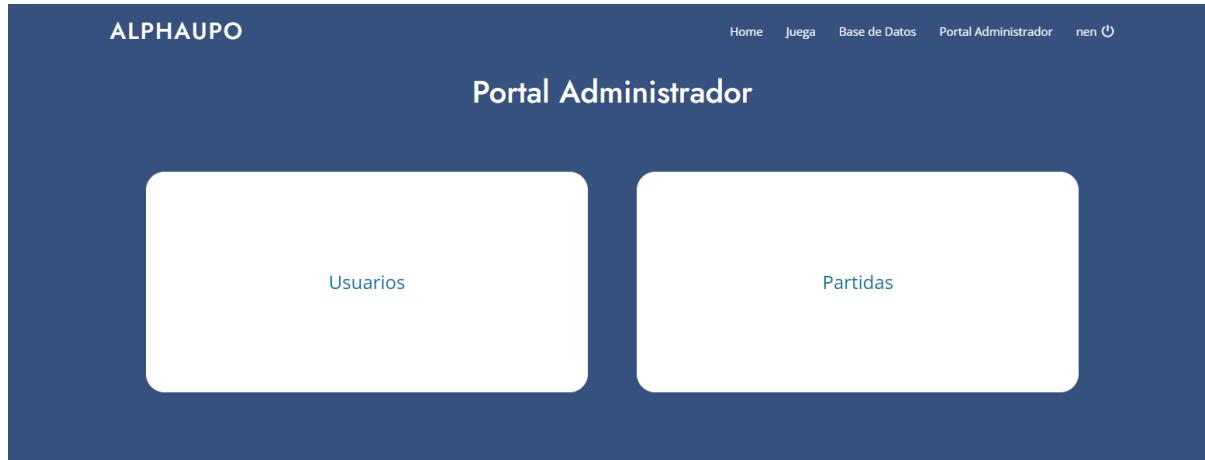


This screenshot shows the same application interface as Figure 206, but with a different search query. The search bar now displays 'Buscar Jugador:' and the dropdown menu is set to 'Cuenca Jimenez,J'. The table below shows the same 8 rows of data from the Gibraltar Masters 2019 event, with the player Cuenca Jimenez,J appearing in both the Black and White columns.

Id	Black	White	Event	Date	Round	Result	Site	
3095	Adams,Mi	Cuenca Jimenez,J	Gibraltar Masters 2019	2019-01-23	2	1/2-1/2	Caleta ENG	Analizar
3096	Gukesh,D	Adams,Mi	Gibraltar Masters 2019	2019-01-24	3	1-0	Caleta ENG	Analizar
3097	Adams,Mi	Lagno,Kateryna	Gibraltar Masters 2019	2019-01-25	4	0-1	Caleta ENG	Analizar
3098	Papp,Ga	Adams,Mi	Gibraltar Masters 2019	2019-01-26	5	1/2-1/2	Caleta ENG	Analizar
3099	Adams,Mi	Edouard,R	Gibraltar Masters 2019	2019-01-27	7	1/2-1/2	Caleta ENG	Analizar
3100	Lalith,B	Adams,Mi	Gibraltar Masters 2019	2019-01-28	8	1/2-1/2	Caleta ENG	Analizar
3101	Adams,Mi	Deac,Bogdan-Daniel	Gibraltar Masters 2019	2019-01-29	8	1/2-1/2	Caleta ENG	Analizar
3102	Svane,R	Adams,Mi	Gibraltar Masters 2019	2019-01-30	9	1-0	Caleta ENG	Analizar

Figure 207: Menacho, E.

3.7 Gestión de Usuarios y Partidas



ALPHAUPO

Ctra. de Utrera, 1, 41013
Sevilla
Email: emende@alu.upo.es

LINKS

- > Home
- > Juega
- > Base de Datos

SOCIAL

Échale un ojo a nuestros métodos de comunicación.



Figure 208: Menacho, E.

Administrar Usuarios				
Id	Name	Email	Admin	
2	user	user@itsolutionstuff.com	x	Eliminar Editar
7	admin	admin@itsolutionstuff.com	✓	Eliminar Editar
9	nen	nenitomg@gmail.com	✓	Eliminar Editar
11	Francisco	fmaralv@upo.es	✓	Eliminar Editar
12	Alicia	atrolor@upo.es	✓	Eliminar Editar

Figure 209: Menacho, E.



The screenshot shows a table titled "Administrador Partidas" (Manage Games) on the ALPHAUPO website. The table lists 8 chess games from the Gibraltar Masters 2019 tournament. Each row contains the following information: Id, Black player, White player, Event, Date, Round, Result, Site, and a red "Eliminar" (Delete) button.

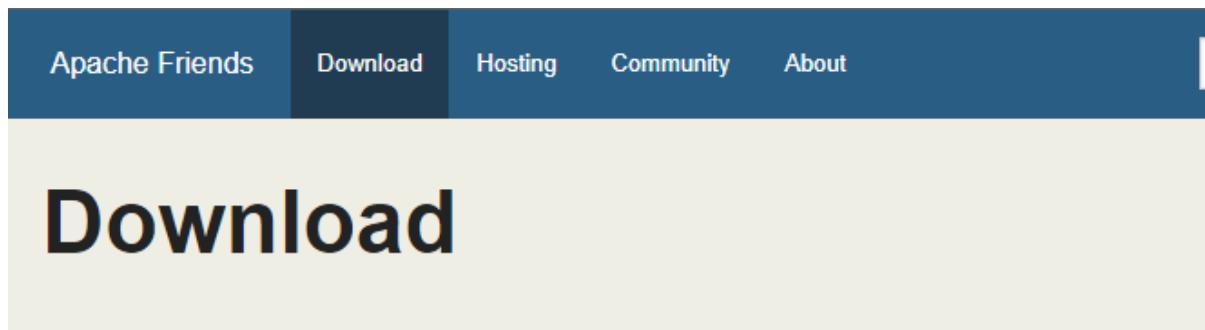
Id	Black	White	Event	Date	Round	Result	Site	
3095	Adams,Mi	Cuenca Jimenez,J	Gibraltar Masters 2019	2019-01-23	2	1/2-1/2	Caleta ENG	Eliminar
3096	Gukesh,D	Adams,Mi	Gibraltar Masters 2019	2019-01-24	3	1-0	Caleta ENG	Eliminar
3097	Adams,Mi	Lagno,Kateryna	Gibraltar Masters 2019	2019-01-25	4	0-1	Caleta ENG	Eliminar
3098	Papp,Ga	Adams,Mi	Gibraltar Masters 2019	2019-01-26	5	1/2-1/2	Caleta ENG	Eliminar
3099	Adams,Mi	Edouard,R	Gibraltar Masters 2019	2019-01-27	7	1/2-1/2	Caleta ENG	Eliminar
3100	Lalith,B	Adams,Mi	Gibraltar Masters 2019	2019-01-28	8	1/2-1/2	Caleta ENG	Eliminar
3101	Adams,Mi	Deac,Bogdan-Daniel	Gibraltar Masters 2019	2019-01-29	8	1/2-1/2	Caleta ENG	Eliminar
3102	Svane,R	Adams,Mi	Gibraltar Masters 2019	2019-01-30	9	1-0	Caleta ENG	Eliminar

Figure 210: Menacho, E.

4. USAR PÁGINA EN LOCAL

4.1 Descargar XAMPP

- <https://www.apachefriends.org/download.html>



XAMPP is an easy to install Apache distribution containing MariaDB, PHP, and Perl. Just download and start the installer. It's that easy.

The screenshot shows the "XAMPP for Windows" download page. At the top, there is a logo of the Windows operating system and the text "XAMPP for Windows 8.0.28, 8.1.17 & 8.2.4". Below this, there is a table listing three versions of XAMPP for Windows:

Version	Checksum	Size
8.0.28 / PHP 8.0.28	What's Included? md5 sha1	Download (64 bit) 144 Mb
8.1.17 / PHP 8.1.17	What's Included? md5 sha1	Download (64 bit) 148 Mb
8.2.4 / PHP 8.2.4	What's Included? md5 sha1	Download (64 bit) 149 Mb

Below the table, there are links for "Requirements" and "More Downloads ». At the bottom, a note states: "Windows XP or 2003 are not supported. You can download a compatible version of XAMPP for these platforms [here](#)". A blue oval highlights the "Download (64 bit)" link for the 8.1.17 version.

Figure 211: Menacho, E.

4.2 Instalar XAMPP

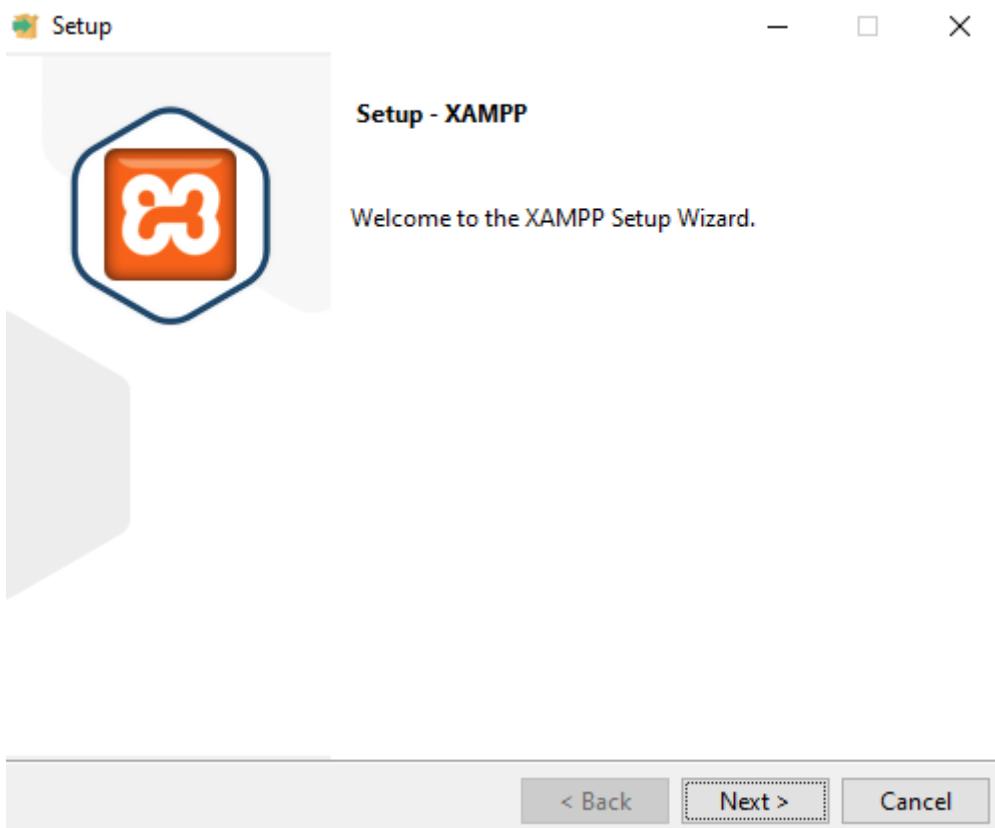


Figure 212: Menacho, E.

Instalamos XAMPP y en las variables de entorno añadimos a la variable PATH la ruta de la carpeta de nuestra instalación de php en XAMPP, por defecto suele ser la carpeta C:\xampp\php

4.3 Levantar el servidor apache y mysql

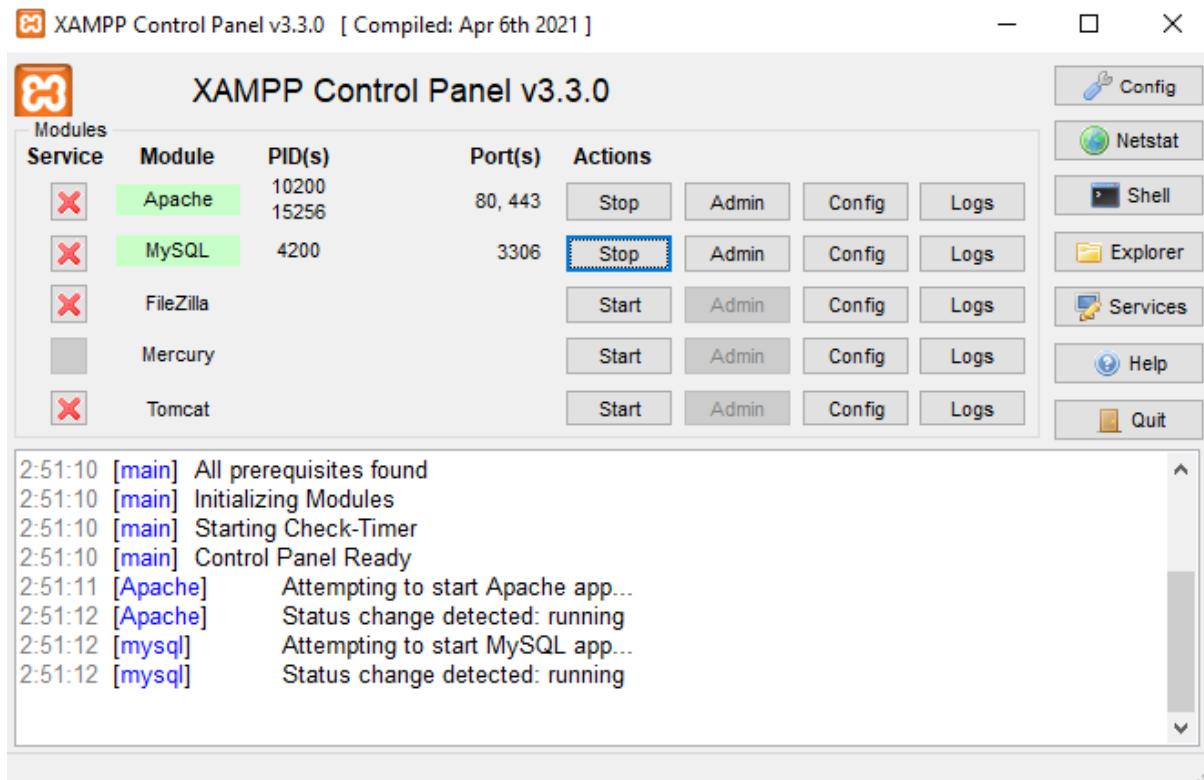


Figure 213: Menacho, E.

4.4 Accedemos al panel de phpmyadmin y importamos la base de datos

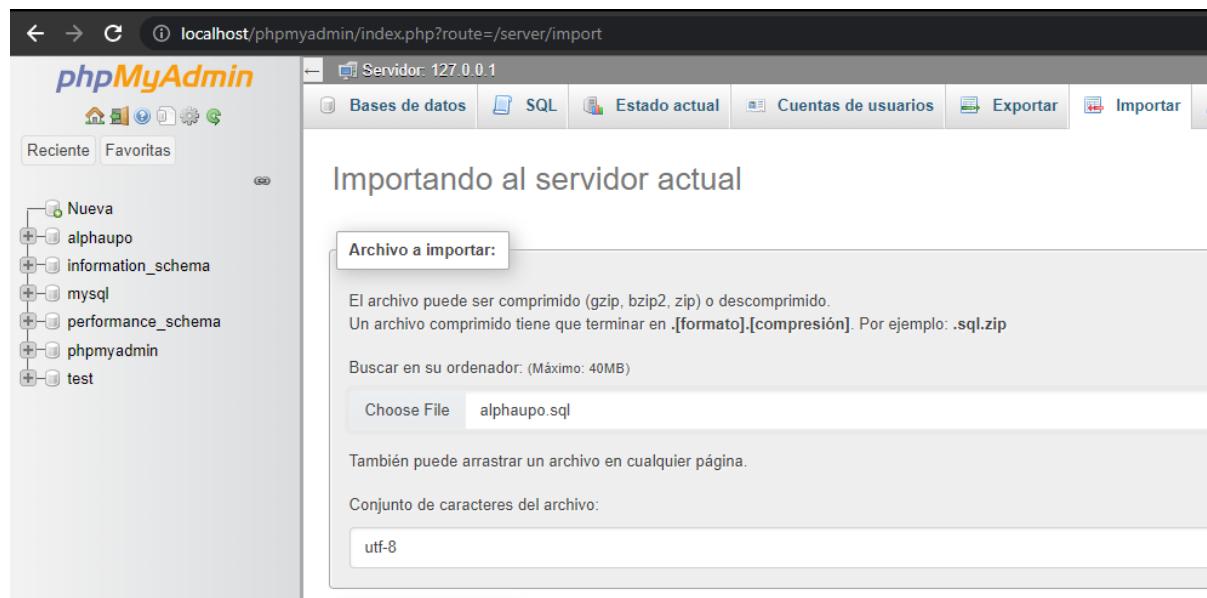


Figure 214: Menacho, E.

4.5 Levantamos la aplicación

Accedemos a la raíz del proyecto y ejecutamos el siguiente comando:

```
C:\Windows\System32\cmd.exe - php artisan serve
Microsoft Windows [Versión 10.0.19044.2965]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\nenit\Documents\Laravel\Alphaupo>php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
[Tue May 16 02:56:22 2023] PHP 8.1.6 Development Server (http://127.0.0.1:8000) started
```

Figure 215: Menacho, E.

Ahora si accedemos a <http://127.0.0.1:8000> encontraremos la página web levantada.

BIBLIOGRAFÍA

1. BIBLIOGRAFÍA

- Chess Programming Wiki. (n.d.). Minimax. Retrieved February 8, 2023, from
<https://www.chessprogramming.org/Minimax>
- The_real_greco. (n.d.). The Original Chess Engine: Alan Turing's Turochamp. Chess.com. Retrieved February 8, 2023, from
https://www.chess.com/blog/the_real_greco/the-original-chess-engine-alan-turings-turochamp
- Wikipedia contributors. (n.d.). Minimax. In Wikipedia. Retrieved February 8, 2023, from <https://en.wikipedia.org/wiki/Minimax>
- IBM. (n.d.). Deep Blue. IBM History. Retrieved February 8, 2023, from
<https://www.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/>
- Xataka. (n.d.). Deep Blue: el ordenador con una sola misión, ganar al humano. Xataka. Retrieved February 8, 2023, from <https://www.xataka.com/otros/deep-blue-el-ordenador-con-una-sola-mision-ganar-al-humano>
- Wikipedia contributors. (n.d.). Red neuronal convolucional. In Wikipedia. Retrieved February 8, 2023, from https://es.wikipedia.org/wiki/Red_neuronal_convolucional
- MathWorks. (n.d.). Convolutional Neural Network (CNN). Retrieved February 8, 2023, from <https://es.mathworks.com/discovery/convolutional-neural-network-matlab.html>
- TensorFlow. (n.d.). Convolutional Neural Networks (CNNs). Retrieved February 8, 2023, from <https://www.tensorflow.org/tutorials/images/cnn>
- Sancho Caparrini, F. (n.d.). Aprendizaje Automático: Redes Neuronales [Machine Learning: Neural Networks]. Retrieved February 8, 2023, from
<http://www.cs.us.es/~fsancho/?e=165>

Laravel. (n.d.). Laravel Documentation. Retrieved February 8, 2023, from
<https://laravel.com/docs/8.x/readme>

Quiero dedicar este proyecto a todas las personas que han puesto una mano para que esto salga adelante, en especial va dedicado a mi familia, mi padre, por estar siempre ahí para que no pierda el enfoque y por aguantar mis explicaciones, a mi madre, por estar siempre para sacarme una sonrisa cuando mas lo necesitaba, a mi hermano, por darme “collejas” cuando mas me lo merecía, a Paco mi tutor, por guiarme en este camino tan complejo, a Alicia, por enseñarme las bases de los sistemas complejos, a Pedro Almagro, por darme otra perspectiva de pensamiento, a mis amigos, por escucharme y discutir conmigo sobre ajedrez e informática.

Esto no hubiera sido posible sin ninguno de vosotros. Gracias por todo.