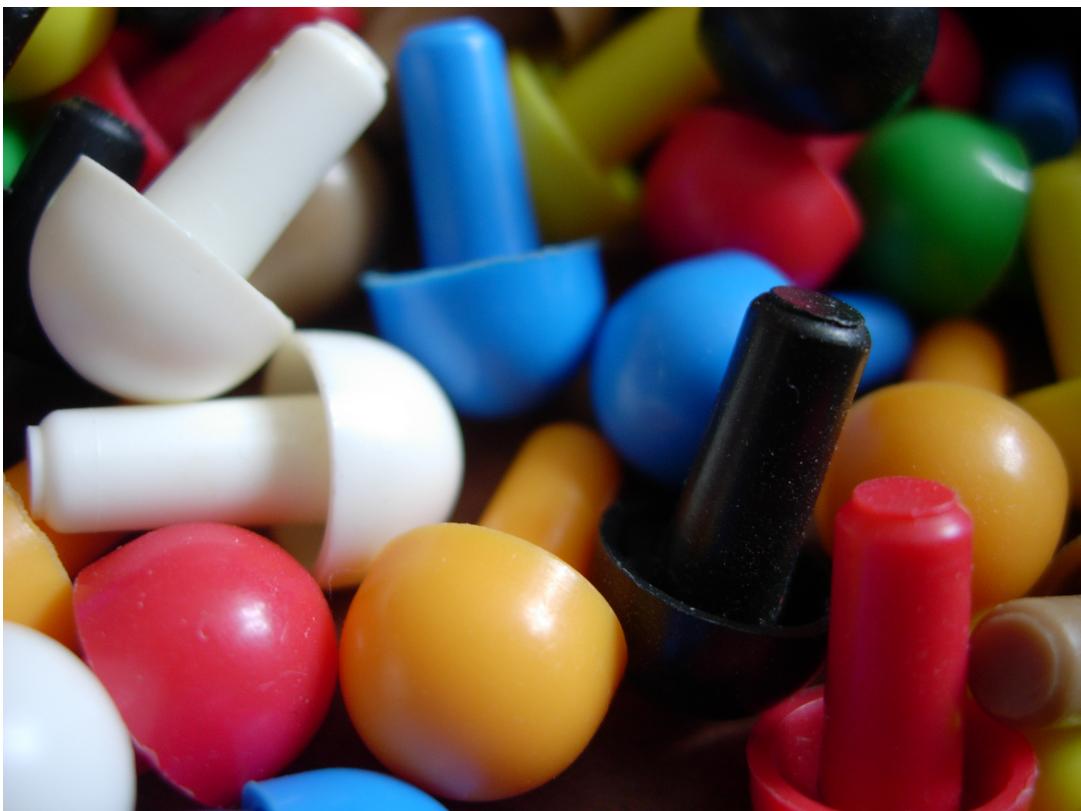


# MASTERMIND

---

## PROJECTE PROP



Versió 2.0

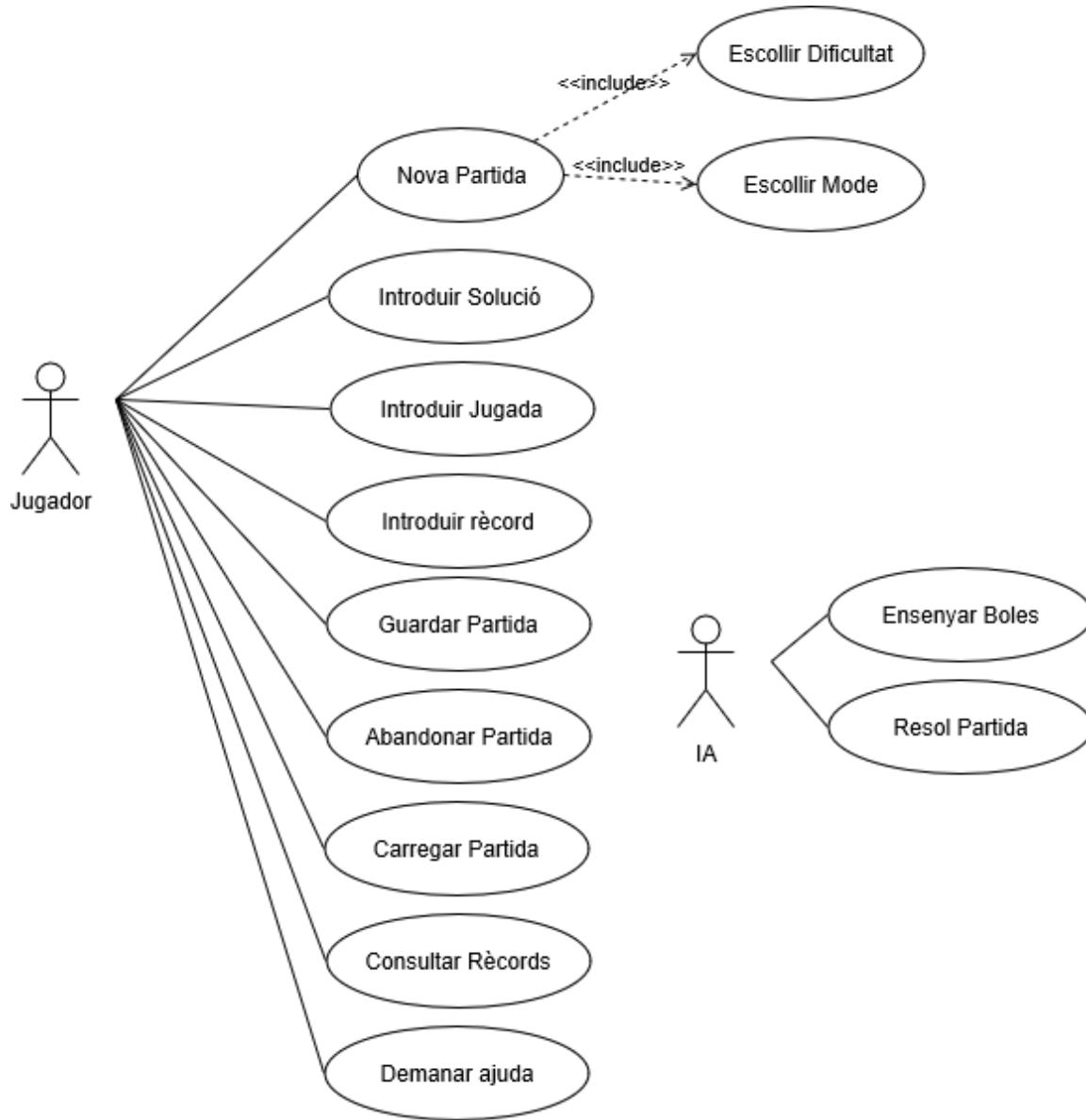
**Grup 24**

**Albert Carreira Muñoz** - albert.carreira  
**Ismael Haddad Ez-zarryaty** - ismael.haddad.ez-zarryaty  
**Jordi Nieto Maldonado** - jordi.nieto.maldonado

# Index:

<b>Index:</b>	<b>2</b>
<b>1. Diagrama de casos d'ús</b>	<b>3</b>
Descripció de cada cas	3
Nova partida	3
Include: Escollir Mode	3
Include: Escollir Dificultat	4
Introduir Solució	4
Introduir Jugada	4
Extend: Introduir record	4
Abandonar Partida	4
Extend: Guardar Partida	5
Carregar Partida	5
Consultar Rècords	5
Demanar Ajuda	5
Resol Partida	5
<b>2. Diagrama estàtic complet del model conceptual de dades</b>	<b>6</b>
Relació de les classes implementades per cada membre	7
<b>3. Descripció d'estructures de dades i algorismes</b>	<b>8</b>
Classe Pair	8
Classe Codi (Abstracta)	8
Classes CodiFacil i CodiDificil	8
Classe Partida	8
Algorisme Five Guess	9
Classe ControladorPartida	9
Classe Jugada	9
Classe Ranking	9
Classe ControladorRanking	9
<b>4. Futures implementacions:</b>	<b>10</b>

# 1. Diagrama de casos d'ús



## Descripció de cada cas

### Nova partida

Quan l'usuari escull començar una nova partida, el sistema li pregunta el mode i la dificultat en qu   la vol jugar. Acte seguit el sistema l'introduceix a la partida perqu   pugui jugar-la.

Include: Escollir Mode

L'usuari escull jugar com a *Codemaker* o com a *Codebreaker*.

- Si juga com a **Codemaker** l'usuari haurà d'introduir la solució de la partida i la IA generarà jugades per tal de trobar la combinació, utilitzant l'algorisme de Five Guess.
- Si juga com a **Codebreaker** es generarà una solució aleatòria i l'usuari haurà d'introduir jugades per tal d'intentar trobar la combinació correcta.

### Include: Escollir Dificultat

L'usuari escull el nivell de dificultat de la partida: fàcil o difícil. En ambdós casos hi hauran 8 colors disponibles, però es varia la quantitat de colors que s'utilitzen i s'afegeix una restricció de repetits en la dificultat difícil.

- Si es juga una partida **fàcil** la quantitat de colors que es podran utilitzar serà de 4. No hi podran haver repetits de cap color.
- Si es juga una partida **diffícil** la quantitat de colors que es podran utilitzar serà de 6. Hi ha la possibilitat de què es repeteixi cada color com a màxim un cop.

### Introduir Solució

Si l'usuari juga com a **Codemaker**, aleshores haurà d'introduir la combinació a endevinar per la màquina amb les restriccions de la dificultat establerta. En cas contrari la solució és generada de manera aleatòria i, per tant, l'usuari no ha d'introduir cap solució.

### Introduir Jugada

Si l'usuari juga com a **Codebreaker**, aleshores introduceix la següent combinació per tal d'intentar endevinar la solució prèviament generada de forma aleatòria. Acte seguit es calculen i mostren les boles Blanques i Negres corresponents a la comparació de la jugada amb la solució de la partida.

### Introduir record

Quan s'acaba una partida, si la puntuació de la partida actual supera algun rècord anterior de la mateixa dificultat, l'usuari haurà d'introduir el seu nom per a registrar-lo. La puntuació de la partida, per tant, és rècord en la dificultat establerta si:

- No s'han registrat la màxima quantitat de rècords que el sistema permet guardar en aquesta dificultat (10).
- S'ha arribat al màxim de rècords que el sistema permet guardar en aquesta dificultat i el jugador ha superat algun dels rècords existents al sistema (és a dir, ha arribat a la solució amb un nombre menor de jugades que un altre). S'eliminarà l'últim rècord de la taula i es situarà el nou rècord a la posició corresponent (segons les puntuacions en ordre creixent, és a dir, com menor sigui la puntuació, millor).

## Abandonar Partida

L'usuari està jugant una partida i decideix abandonar-la. El sistema li preguntarà si vol guardar la partida i finalment enviarà l'usuari a la pantalla principal.

## Guardar Partida

Un cop l'usuari ha decidit abandonar la partida pot guardar l'estat actual per a continuar en un altre moment. Si es dóna el cas, el sistema s'encarregarà de guardar l'estat de la partida correctament.

## Carregar Partida

Quan l'usuari vulgui carregar la partida des del menú principal, el sistema li mostrarà la llista de partides guardades i haurà d'escol·lir-ne la que vulgui. El sistema s'encarregarà de carregar la partida tal i com estava quan la va guardar i, d'aquesta manera, l'usuari la podrà seguir jugant.

## Consultar Rècords

Se li ofereix a l'usuari la possibilitat de consultar els rècords guardats actuals des del menú principal. L'usuari escol·lirà la dificultat i el sistema mostra la taula de rècords segons la dificultat establerta.

## Demanar Ajuda

Mentre l'usuari està jugant una partida, pot demanar que se li proporcioni una pista que l'ajudi a trobar la solució de la partida. Per cada pista demanada, a l'usuari se li sumarà 5 punts com a penalització.

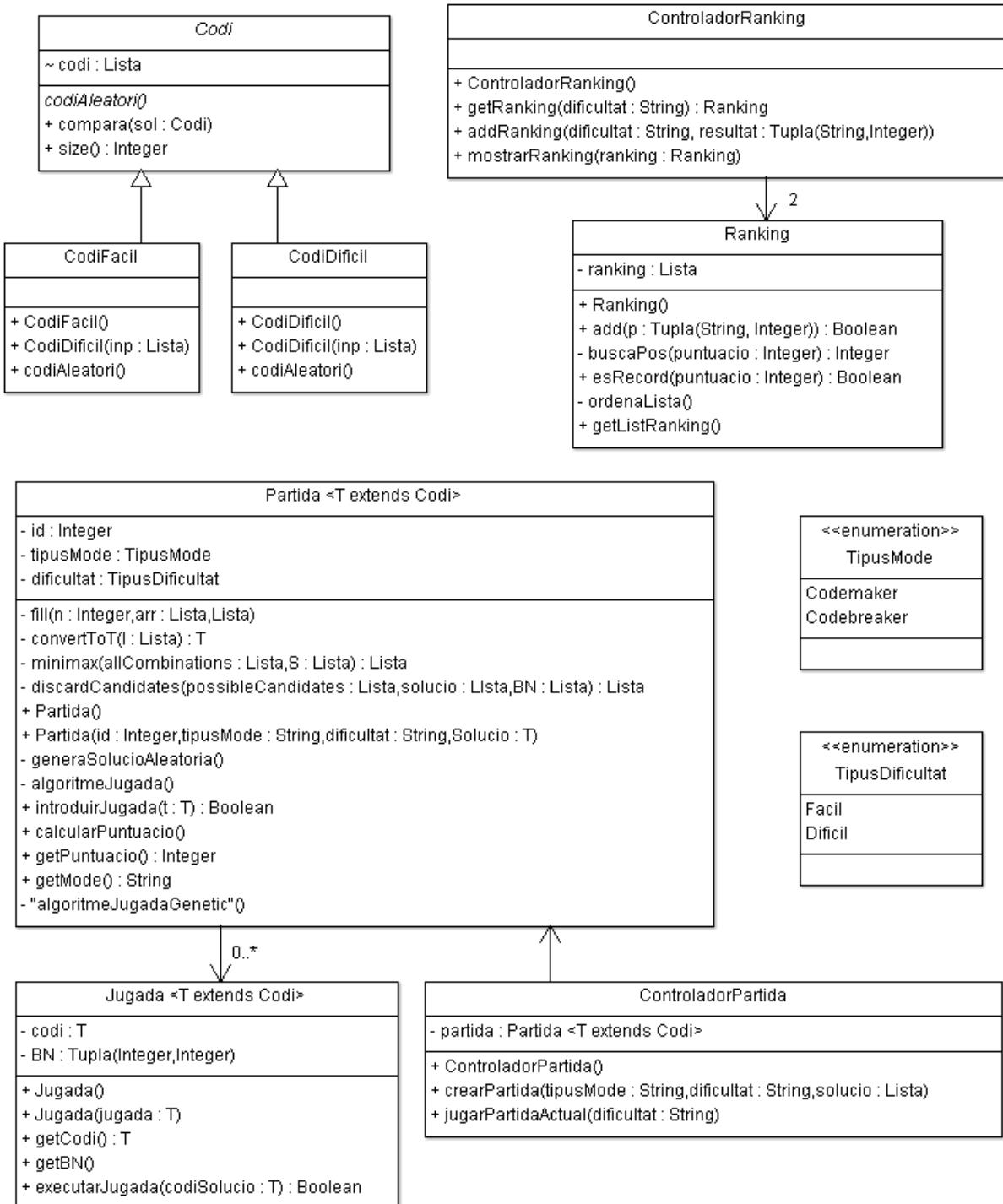
## Ensenya boles

A cada jugada introduïda per l'usuari, la màquina compararà els codis de la jugada i de la solució i ensenyarà la quantitat de boles blanques i negres corresponents.

## Resol Partida

Quan s'està jugant com a **Codemaker**, després de què el jugador hagi introduït la solució a resoldre la màquina aplicarà l'algorisme del Five Guess (1977, *Donald Knuth*) per a arribar a la susdita solució.

## 2. Diagrama estàtic complet del model conceptual de dades



## Relació de les classes implementades per cada membre

	Albert	
Classes	Partida	ControladorRanking
Tests	Junit	Driver

	Jordi				
Classes	Pair	Codi	CodiFacil	CodiDificil	Ranking
Tests	-----	Driver	Driver	Driver	Driver

	Ismael		
Classes	Jugada	ControladorPartida	
Tests	Driver	Driver	

### 3. Descripció d'estructures de dades i algorismes

#### Classe *Pair*

Hem implementat una classe d'ajuda genèrica *Pair* que, de forma similar a la que hi ha en C++, ens permet guardar parells d'objectes i treballar amb les dues parts (*left* i *right*). Necessitavem guardar el resultat de les jugades en una estructura de dades per a poder treballar fàcilment amb ells, i no ens convencia l'opció d'utilitzar una llista de només dos elements. Així doncs, i donat que Java no implementa aquesta classe, vam decidir crear-ne una classe genèrica que simulés el comportament per defecte i ens ajudés a complir el nostre objectiu.

#### Classe *Codi (Abstracta)*

Classe fonamental per a poder fer jugades de diferents dificultats i comparar-les. Implementa una sèrie de constants i una llista on es guardaran els valors dels colors jugats (de llargària diferent segons la subclasse instanciada per dificultat), a més de dos mètodes abstractes que s'especificaran a cada subclasse. A més, en aquesta classe també s'implementen dos mètodes d'IO (*input-output*) per defecte, un mètode de comparació de dos codis i un *getter* que retorna la llargària del codi (a partir de la qual podrem saber de manera fàcil en quina dificultat estem jugant).

#### Classes *CodiFacil* i *CodiDificil*

Ambdues són subclasses de *Codi*. Cada una assigna a la seva llista de colors una llargària diferent (4 per fàcil i 6 per difícil), a més de reimplementar el *input* per a adequar-ho. Els colors vénen representats per nombres sencers. També, donat que amb *CodiFacil* no pot haver-hi repetits però en *CodiDificil* n'hi pot haver un de cada color (són restriccions diferents), cadascuna implementa la seva pròpia funció per a calcular una combinació aleatòria.

#### Classe *Partida*

Aquesta classe implementa tota la funcionalitat necessària per a poder jugar una partida. Cal remarcar que és una classe genèrica que s'implementarà de forma diferent segons si volem fer jugades amb codis fàcils o difícils. Guarda els atributs indispensables per a poder jugar (identificació de la partida, per a poder carregar-la més endavant en cas que es vulgui guardar; dificultat, puntuació i el mode en què juguem) i té mètodes que permeten generar un codi aleatori i introduir una jugada. La funcionalitat més important, però, és la implementació de una IA, un algorisme capaç de solucionar el joc per a qualsevol solució (mode *CodeMaker*).

## Algorisme Five Guess

Algorisme implementat a la classe *Partida* que permet arribar a la solució proposada en com a molt 5 conjectures. Per a poder portar a terme aquest algorisme s'han codificat diversos mètodes: *fill*, que omple una llista amb totes les combinacions possibles segons el nombre de colors i la quantitat que n'utilitzem; *convertToT*, que donada una llista d'enters ens retorna una instància de Codi corresponent per a poder comparar diferents codis; *minimax*, que escolleix les millors combinacions per a continuar amb l'algorisme i arribar a la solució i *algorismeJugada*, que conté les inicialitzacions necessàries i el cos del bucle principal.

*Sempre es comença amb la jugada 1122 perquè està demostrat que es minimitza el nombre de conjectures necessàries per a arribar a la solució.*

## Classe ControladorPartida

Classe més exterior de la capa de domini que es comunica amb la classe Partida. Ens permet manipular la partida en sí, tot creant una de nova, guardant-la o carregant-la, o jugar-la mentre s'afegeixen nous codis i es controla el resultat.

## Classe Jugada

És la responsable de guardar la combinació introduïda pel jugador (CodiFacil / CodiDificil) i les boles Blanques i Negres resultants de comparar-ho amb la solució de la partida.

## Classe Ranking

Ens permet crear una llista amb el nom del jugador i la seva puntuació en la partida. Està ordenada per puntuacions en ordre creixent ja que, com la puntuació es el nº de jugades fetes a la partida, com menys se n'hagin fet millor serà el rècord. Els nostres rànquings tenen la restricció que només guarden els 10 millors registres.

Per implementar-la hem aprofitat que havíem creat la classe genèrica Pair per guardar-nos les dades en una llista de parells.

## Classe ControladorRanking

Classe intermitja entre la capa de domini i la classe Ranking. Permet administrar correctament la creació d'un rànquing per a cada dificultat i assegurar la correcta introducció i lectura de les dades en cadascun d'aquests.

## 4. Futures implementacions:

De cara a la entrega final tenim algunes possibles implementacions i millores en ment:

- Implementar un segon algorisme per a la IA (Genetics), en el mode de CM, per tal de poder escollir amb quin algorisme es vol que es resolgui la solució.
- Millorar l'eficiència de l'algorisme actual del Five Guess ja que, tot i que s'arriba a la solució, en el cas difícil el temps d'execució és extremadament gran.
- Implementar la funcionalitat de donar ajut a l'usuari
- Implementar una possible dificultat intermèdia.
- Implementar gestió d'errors i excepcions.