Tetr3d

Joc de Tetris en 3D amb sincronització al núvol

Institut Puig Castellar
Cicle formatu: DAM2
Grau: CFGS
Projecte intermodular

Autor: Pau Abril Iranzo
Tutor: Luis Elía Talón
Curs acadèmic: 2025–2026
Data de lliurament: 17/5/2026

Resum del projecte

Tetr3d és un joc de puzzle inspirat en el Tetris clàssic, portat a tres dimensions amb tetracubs (peces formades per quatre cubs) que cauen per una graella de 10×10×20.

El projecte comprèn tres components interconnectats. En primer lloc, el client del joc, desenvolupat amb Godot i GDScript, amb suport multiplataforma (Linux i Windows). En segon lloc, un back-end construït amb FastAPI i Python, que ofereix autenticació JWT i persistència sobre DocumentDB i Valkey. En tercer lloc, una aplicació web feta amb Astro i Vue, que conté la pàgina d'inici, documentació, enllaços a les descàrregues, gestió de comptes, rànquings públics i panells d'administració. Els tres clients consumeixen una mateixa especificació OpenAPI 3.0, generant SDKs per a fer les crides al back-end. El sistema es desplega amb Docker Compose i, opcionalment, darrere d'un proxy invers Caddy.

El resultat és un sistema complet client-servidor: un client del joc, un back-end que gestiona autenticació, sincronització de configuració i puntuacions, i un client web públic amb rànquings i panells d'administració

Paraules clau

Tetris 3D, multiplataforma, sincronització al núvol, rànquings, allotjament local

---

Abstract

Tetr3d is a puzzle game inspired by classic Tetris, brought into three dimensions with tetracubes (four-cube pieces) that fall through a 10×10×20 grid.

The project spans three interconnected components. First, the game client, built with Godot and GDScript, with cross-platform support for Linux and Windows. Second, a backend built with FastAPI and Python, providing JWT authentication and persistence over DocumentDB and Valkey. Third, a website built with Astro and Vue, hosting the landing page, documentation, links to the downloads, account management, public leaderboards and administration panels. All three

clients consume a single OpenAPI 3.0 specification, with autogenerated SDKs to make calls to the backend. The whole system is deployed with Docker Compose and optionally behind a Caddy reverse proxy.

The result is a complete client-server system: a game client, a backend that handles authentication, configuration sync and scores, and a public website with leaderboards and an administration panels.

Keywords

3D Tetris, cross-platform, cloud sync, leaderboards, self-hostable

Índex

1. Presentació del projecte

1.1 Introducció

Tetr3d és un joc puzzle en tres dimensions inspirat en el Tetris clàssic. En lloc de tetròminos plans, fa servir tetracubs, peces formades per quatre cubs en diferents configuracions, que cauen per una graella tridimensional de 10×10×20 cel·les.

El projecte també inclou un backend per a l'autenticació i la sincronització de la configuració entre dispositius, i una aplicació web que permet visualitzar els rànquings públics i fer operacions d'administració. Els tres components estan acoblats únicament per una especificació OpenAPI 3.0, a partir de la qual es generen automàticament SDKs per al joc (GDScript) i la web (TypeScript).

1.2 Context

El projecte sorgeix de dos interessos personals i d'una observació tècnica.

D'una banda, el desenvolupament de videojocs és una disciplina que no havia treballat abans i volia provar-la amb Godot, un motor totalment FOSS. D'una altra, volia experimentar amb DocumentDB com a alternativa FOSS als motors relacionals tradicionals i a productes comercials com MongoDB.

L'observació tècnica que connecta tots dos interessos és que el Tetris en 3D està poc explorat: les implementacions existents tendeixen a quedar-se a mitges, ja sigui sense completar el conjunt de mecàniques esperades del Tetris clàssic o bé sense oferir cap funcionalitat distribuïda (comptes d'usuari, sincronització entre dispositius, rànquings públics).

1.3 Justificació

El problema que es vol resoldre és precisament aquest buit: no he trobat cap implementació de Tetris en 3D que tingui simultàniament totes les mecàniques esperades del Tetris clàssic (peces construïdes amb 4 cubs, generador aleatori sense repeticions...) i una arquitectura distribuïda real (autenticació, sincronització i rànquings).

Tetr3d apunta a ser aquesta peça. Un joc 3D fidel al Tetris clàssic i, alhora, un sistema complet client-servidor amb les funcionalitats que els jugadors esperen d'un producte modern (perfil, configuració sincronitzada, rànquings públics).

1.4 Objectius

Objectius generals

dissenyar i implementar un joc puzzle 3D estil Tetris amb sincronització al núvol i una aplicació web, formant un sistema complet i jugable de tres components independents.

Objectius específics

- Implementar les mecàniques principals del joc (peces, gravetat i drops, rotacions 3D, hold, peca fantasma, generador aleatori sense repeticions, DAS/ARR) en Godot amb GDScript.
- Dissenyar i construir un back-end REST amb FastAPI que ofereixi autenticació JWT amb claus asimètriques, sincronització de configuració i un sistema de rànquings.
- Modelar la persistència amb DocumentDB i cache amb Valkey, orquestrat amb Docker Compose.
- Desenvolupar un lloc web modern amb Astro i Vue, que cobreixi landing, descàrregues, autenticació, rànquings i panell d'administració.
- Definir una especificació OpenAPI única com a contracte entre clients i servidor, i generar automàticament els SDKs de client per a Godot (`openapi-generator`) i per al web (`@hey-api/openapi-ts`).

2. Estratègia i planificació

2.1 Estratègia de desenvolupament i viabilitat

L'estratègia escollida és el desenvolupament de components independents amb un back-end comú centralitzat. Cada component (joc, back-end, web) viu en el seu propi repositori i té el seu propi cicle de vida; l'única dependència entre ells és l'especificació OpenAPI del back-end, que permet la generació automàtica de SDKs als clients.

Viabilitat tècnica: tot l'stack escollit (Godot, FastAPI, Astro, Vue, DocumentDB, Valkey, Docker) és programari lliure, amb comunitats actives, i la generació automàtica de SDKs dels clients facilita el desenvolupament

Viabilitat temporal: tractant-se d'un sol desenvolupador, ha calgut prioritzar les funcionalitats essencials (joc funcional amb autenticació i sincronització) per damunt d'altres funcionalitats (moviment de la càmera, més modes de joc, traduccions).

2.2 Metodologia de treball

La gestió del treball s'ha fet amb una instància propia d'OpenProject.

La metodologia utilitzada es basa en tasques i fites organitzades en un diagrama de Gantt, que és la vista principal per a la planificació a mig i llarg termini i permet visualitzar les fases del projecte, les seves durades i les dependències entre elles. Cada tasca té una data d'inici i finalització prevista, i un estat (new / in progress / closed).

2.3 Planificació

En el desenvolupament, s'han seguit fases organitzades segons els components del sistema amb punts claus de funcionament.

Fase	**Objectiu**
Joc bàsic funcional	Implementar una versió bàsica del videojoc sense connexió amb el back-end
Desenvolupament del back-end	Crear el back-end amb autenticació, models de dades per a les peticions i respostes i persistència sobre DocumentDB
Desenvolupament de l'aplicació web	Crear l'aplicació web i integrar-la amb el back-end amb el client OpenAPI autogenerat
Integració joc amb el back-end i implementar sistema de puntuacions	Connectar el client del joc amb el backend: generació del SDK GDScript, login, sincronització de configuració entre dispositius i submissió de puntuació al final de cada partida.
Millores d'interfície	Polir la UX/UI del videojoc implementant un tema centralitzat amb estils semblants a l'aplicació web
Correcció d'errors i funcionalitats d'admin	Corregir els bugs detectats durant les proves finals i afegir el panell d'administració al web (gestió d'usuaris i codis de registre)

3. Anàlisi

3.1 Casos d'ús

Els casos d'ús principals del sistema són:

- **Jugar una partida** en algun dels modes disponibles:
  - estàndard (àrea de joc de 10×10×20)
  - petit (àrea de joc de 5×5×10)
  - pla (àrea de joc amb poca profunditat 10×5×20)
  - personalitzat (paràmetres triats per l'usuari)
- **Configurar el joc**: configurar els controls, ajustar la responsivitat del moviment i canviar les preferències d'interfície.
- **Registrar-se i iniciar sessió** des del joc o des de la web.
- **Sincronitzar la configuració** entre dispositius automàticament en iniciar sessió i en fer canvis.
- **Enviar puntuacions** al final d'una partida i **consultar rànquings públics** per mode des del web.
- **Administrar usuaris i codis de registre** *(rol administrador)* des de la web.

3.2 Requisits funcionals

Joc

- 8 tipus de peces definides en una taula estàtica.
- Moviment en els eixos X i Z, rotació al voltant dels tres eixos, gravetat, hold (intercanvi amb la peça reservada) i ghost piece (previsualització de la posició on caurà la peça).
- Sistema de generar peces aleatòries en grups de 8 sense repeticions o completament aleatori, i amb previsualització de les següents peces.
- Sistema d'entrada amb DAS (Delayed Auto-Shift) i ARR (Auto Repeat Rate) configurables.
- Modes de joc predefinits i mode de joc personalitzat.
- Inici de sessió al back-end, càrrega del perfil i sincronització de la configuració i puntuació.

Back-end

- Rutes d'autenticació (inici de sessió, registre).
- Rutes per actualitzar la configuració
- Rutes per afegir les puntuacions
- Rutes per veure els rànquings
- Rutes d’administració

Web

- Landing page, pàgina de descàrregues, inici de sessió i registre.
- Pàgines de documentació
- Rànquings públics consultables sense autenticació.
- Panells d'administració accessibles per a usuaris amb rol admin.
- Pàgines d'error

3.3 Requisits no funcionals

- **Rendiment**: el joc ha de funcionar a 60 FPS estables en clients Linux i Windows.
- **Seguretat**: Els tokens JWT es signen amb claus asimètriques, les contrasenyes es guarden amb bcrypt, comunicacions HTTPS quan es desplega amb Caddy.
- **Usabilitat**: Tema visual coherent (Catppuccin Mocha) a tots tres clients, controls de joc completament configurables.
- **Portabilitat**: Builds del joc per a Linux, Windows.
- **Adaptable**: El lloc web és responsive.
- **Mantenibilitat**: els clients d’API es generen automàticament a partir de l'especificació OpenAPI.
- **Desplegament**: arquitectura containeritzada amb Docker Compose, reproduïble en qualsevol host amb Docker. Proxy invers Caddy opcional per a HTTPS automàtic.

3.4 Anàlisi d’alternatives tecnològiques

Motor de joc

Alternativa	Punts forts	Punts febles
Unity	Ecosistema gran	Llicència restrictiva
Unreal Engine	Gràfics d’alta qualitat, ecosistema gran	Excessiu per a un joc 3D de blocs senzill
Godot	FOSS, exports natius lleugers	Comunitat més petita

Vaig escollir Godot perquè no necessitava gràfics d’alta qualitat i és de codi lliure.

Backend

Alternativa	Punts forts	Punts febles
Express	TypeScript, mateix llenguatge que la web	
FastAPI	Validació amb Pydantic, generació OpenAPI automàtica	Python té un tipatge poc estricte que pot no ser compatible amb el codi del videojoc

Vaig escollir FastAPI perquè ja era familiar amb Python i el framework i per la generació automàtica de l'especificació OpenAPI.

Base de dades

Alternativa	Punts forts	Punts febles
PostgreSQL	Relacional, robust, transaccions	Esquemes rígids
DocumentDB	FOSS, esquemes flexibles, JSON natiu,	Base de dades nova i amb poca documentació
Redis	Documentació extensa	Llicència restrictiva
Valkey	Fork FOSS de Redis, és més ràpid que Redis	Base de dades nova

Vaig escollir DocumentDB i Valkey per provar-les, ja que són alternatives FOSS per MongoDB i Redis recents.

Frontend web

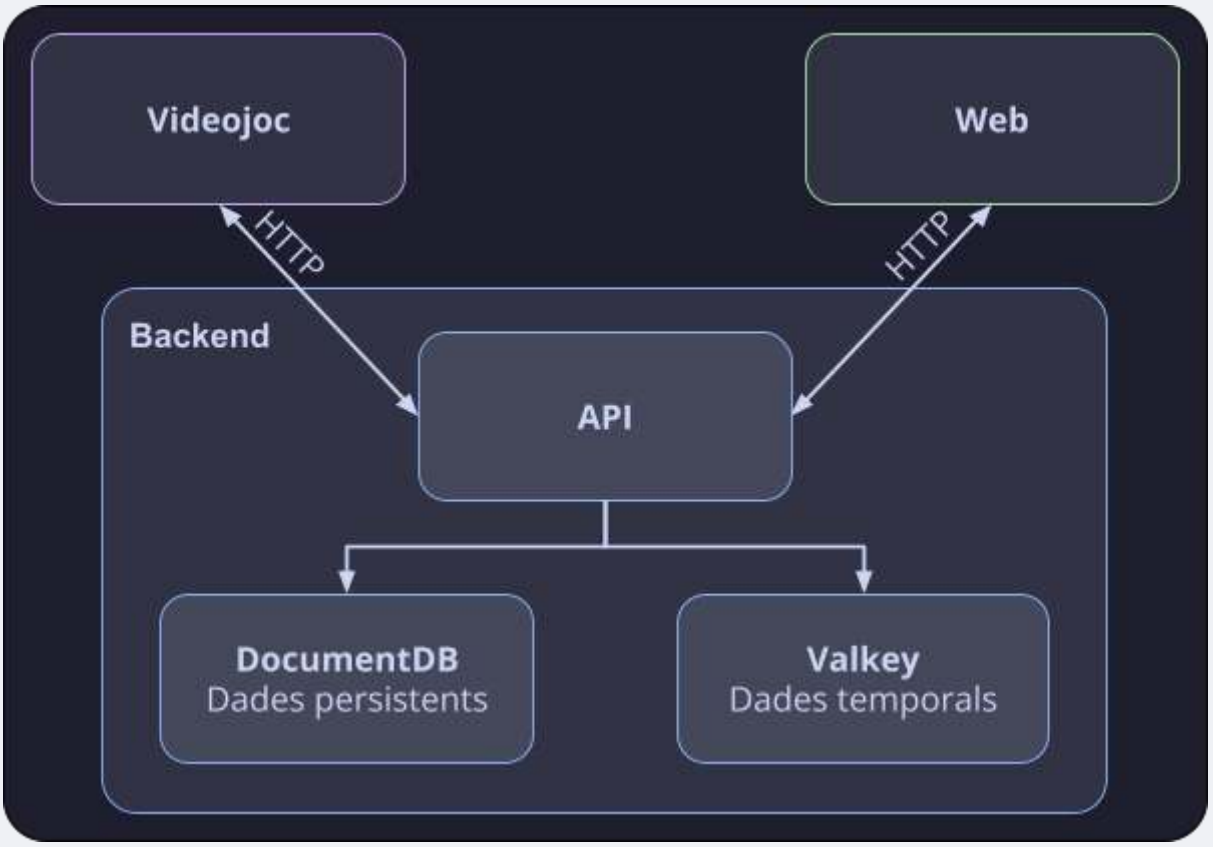
Alternativa	Punts forts	Punts febles
VitePress	Generator de webs estatics sobre Vite i Vue	Orientat a documentació. Poc flexible per a pàgines interactives
Vue	SPA, control total de l¡aplicacio	Orientat a aplicacions completament dinàmiques
Astro	Static-first amb integracions de components vue	Combinació menys habitual

Vaig escollir Astro després de provar VitePress, ja que funciona bé per a webs estàtiques i puc implementar components Vue amb les funcionalitats complexes

4. Disseny

4.1 Arquitectura del sistema

El sistema es divideix en tres components independents que es comuniquen via HTTP/REST a través d'una especificació OpenAPI compartida:



4.2 Model de dades

El projecte no utilitza una base de dades relacional, en comptes de taules, fa servir col·leccions de documents JSON.

Les col·leccions de l'aplicació són les seguents:

users

Camp	Tipus	Descripció
`_id`	`ObjectId`	Identificador únic
`username`	`string`	Nom d'usuari, únic
`password`	`string`	Hash bcrypt de la contrasenya
`email`	`string`	Adreça de correu
`name`	`string`	Nom complet
`role`	`string`	Rol (`user` o `admin`)
`avatar`	`string`	URL o referència a l'avatar
`created_at`	`datetime`	Data de registre

game_configs

Camp	Tipus	Descripció
`_id`	`ObjectId`	Identificador únic
`user_id`	`ObjectId`	Referència a `users`
`input`	`object`	Keybindings i responsivitat (DAS, ARR)
`ui`	`object`	Preferències de la interfície

games

Camp	Tipus	Descripció
`_id`	`ObjectId`	Identificador únic
`user_id`	`ObjectId`	Referència a `users`
`date`	`datetime`	Data i hora de la partida
`points`	`integer`	Puntuació obtinguda
`game_mode`	`string`	Mode (`default`, `small`, `flat` o `custom`)

4.3 Disseny d’interfície

Joc

- La interfície utilitza un tema global Godot definit a `assets/theme.tres`, basat en la paleta Catppuccin Mocha. La tipografia és Open Sans.
- Hi ha diverses variacions dels botons disponibles amb diferents mides segons les necessitats de l’ús.
- Els botons verticals dels menús principals tenen un efecte de slide horitzontal en hover/focus.
- La majoria dels menús són escenes estàtiques, i els menús de configuracions (`SettingsMenu` i `CustomGameMenu`) es creen dinàmicament des del codi en comptes de definir-se com a escenes estàtiques, ja que el seu contingut és dinàmic.
- Els sliders i toggles fan servir icones SVG personalitzades a `assets/ui/`.

Web

- Fa servir TailwindCSS com a sistema d'estil.
- Es manté coherència visual amb el client del joc mitjançant la mateixa paleta de colors i tipografia.
- Utilitza el sistema de components shadcn/vue personalitzats
- Les illes Vue interactives es munten sobre pàgines Astro estàtiques per minimitzar el JavaScript enviat al navegador fent que les pàgines carreguin més ràpidament.

5. Desenvolupament

5.1 Estructura del projecte

El projecte està dividit en 3 repositoris diferents:

tetr3d (joc)

- **project.godot** Configuració de Godot
- **export_presets.cfg** Configuracio per a compilar el joc
- **scripts/** Lògica Principal
  - **field_controller.gd** Bucle principal del joc
  - **api_client_init.gd** Wrapper del client OpenAPI generat
  - **gen/** Client OpenAPI autogenerat
- **scenes/** Escenes (.tscn)
- **assets/** Tema, fonts, imatges
- **codegen.sh** Genera el client OpenAPI

tetr3d-server (back-end)

- **api/v1/**
  - **main.py** Punt d'entrada de l'aplicacio
  - **pyproject.toml** Configuració i dependències de Python
  - **docker/** Configuracions de docker per al servei
  - **app/** Codi font
    - **__init__.py** Configuració FastAPI i connexions a BBDD
    - **models/** Models de dades per a l'aplicació
    - **routes/** Endpoints agrupats per recurs
    - **services/** Lògica principal dels endpoints
    - **dependencies/** Middlewares d'autenticació
- **docker-compose.yml** Configuració per a docker amb les BBDD
- **.env** Configuració de l'aplicació
- **Caddyfile.example** Exemple de configuració de proxy

tetr3d-web (aplicació web)

- **astro.config.mjs** Configuració d'Astro i Vite
- **package.json** Dependències de l'aplicació
- **src/**
  - **pages/** Pàgines en Astro o Markdown
    - **_app.ts** Configuració de Vue
  - **components/** Components per a les pàgines
    - **ui/** Components bàsics
  - **layouts/** Layouts per a renderitzar les pàgines
  - **stores/auth.ts** Store de Pinia per a gestionar l'autenticació
  - **lib/**
    - **gen/** Codi generat per a les crides d'api
  - **styles/** Estils i configuracio de Tailwind CSS
- **codegen.sh** Genera el client OpenAPI

5.2 Implementació de funcionalitats

Joc

La lógica es basa en un controlador i tres clases principals

- **field_controller.gd**: bucle principal que connecta les peces, el camp de joc, els paràmetres de la partida, l'entrada de l'usuari i la detecció de pèrdua.
- **Piece**: Representa una peça concreta definida en la classe `PieceData`. S'encarrega de renderitzar-la sobre un `GridMap`, moure-la, rotar-la i calcular les col·lisions.
- **PlayField**: controla els límits del camp de joc, les cel·les ocupades i les visualitzacions de les parets.
- **PieceGenerator**: Generador aleatori amb previsualització de les següents peces.

Back-end

- L’api s'organitza per recursos, en subpaquets sota **app/routes/**.
- La validació d'entrada i sortida es fa amb Pydantic.
- La signatura dels tokens JWT utilitza claus Ed25519 emmagatzemades en variables d'entorn.
- La connexió a DocumentDB es fa via la llibreria pymongo i a Valkey via el client propi de valkey.

Web

- Les pàgines Astro estàtiques contenen el contingut text/HTML.
- Les funcionalitats interactives (formularis, taules amb filtres, panells d'administració) es munten com a illes Vue
- L'estat d'autenticació viu en una store de Pinia amb persistència a localStorage.
- Les crides al back-end és fan amb el client TypeScript generat amb `@hey-api/openapi-ts`, integrat amb TanStack Query per al control de l’estat i caches.

5.3 Proves

Cap dels tres repositoris incorporen tests automàtics. Les proves realitzades han estat:

- **Joc**: Jugar partides completes en cada mode, verificació dels controls i provar els builds de Linux i Windows.
- **API**: proves dels endpoints amb Yaak, un client REST que permet importar directament l'especificació OpenAPI del servidor.
- **Web**: Navegació per totes les pàgines en navegadors Firefox i Chromium, provant la responsivitat amb les eines de desenvolupador I proves amb comptes de diferents rols.

6. Conclusions

6.1 Conclusions generals

L'objectiu principal del projecte, fer un videojoc de Tetris en 3D amb un sistema client-servidor, s'ha assolit sense alguns objectius menors.

Els reptes més significatius han estat aprendre a desenvolupar fent servir Godot, la falta de temps per al desenvolupament i la manca de documentació de DocumentDB, que ha causat problemes durant el desenvolupament i el desplegament final.

6.2 Assoliment dels objectius

- **Mecàniques del joc**: Ha faltat poder moure la càmera i poder registrar usuaris des del videojoc
- **Back-end**: Ha faltat tindre més operacions d'administrador. Per exemple per a gestionar els usuaris
- **Aplicació web**: S'han assolit tots els objectius.
- **CI/CD**: No s'han implementat

6.3 Valoració de la metodologia i la planificació

L'ús d'OpenProject amb diagrama de Gantt ha funcionat bé, però en ser un únic desenvolupador, ha resultat excessiu i a causa d'anar amb presses per la falta de temps no l'he fet servir per algunes tasques

6.4 Visió de futur

Algunes millores per al projecte serien:

- Implementar una manera de **moure la càmera**, amb el ratolí o a posicions predeterminades
- Afegir **traduccions**, tant al joc com a la web
- Afegir més opcions d'**administració**
- Millorar el **sistema de puntuació** per a fer-lo mes resistent a manipulació i recollir més informació
- **Renovar les sessions** d'autenticació automàticament
- Implementar un sistema de **multijugador**
- Afegir mes modes de joc

7. Glossari

- **Tetracub**: peça formada per quatre cubs, equivalent tridimensional dels tetròminos de Tetris. https://es.wikipedia.org/wiki/Policubo
- **Bag randomizer**: algoritme de generació de peces que reparteix totes les peces possibles dins d'una "bossa" abans de tornar a començar, garantint una distribució justa. https://tetris.wiki/Random_Generator
- **DAS (Delayed Auto-Shift)**: temps que cal mantenir premuda una direcció abans que comenci a repetir el moviment automàticament. https://tetris.wiki/DAS
- **ARR (Auto Repeat Rate)**: temps entre repeticions automàtiques del moviment un cop activat el DAS. https://tetris.wiki/DAS#Auto-repeat_rate
- **Peca fantasma**: previsualització fantasma del lloc on aterrarà la peça actual si es deixa caure. https://tetris.wiki/Ghost_piece
- **Gravetat**: ritme al qual les peces baixen automàticament una cel·la. https://tetris.wiki/Drop#Gravity
- **Hard drop**: deixar caure la peça immediatament fins al fons. https://tetris.wiki/Drop#Instant_dropping
- **Soft drop**: accelerar manualment la caiguda de la peça, sense arribar a la velocitat instantània del hard drop. https://tetris.wiki/Drop#Soft_drop
- **Hold**: acció de guardar la peça actual i intercanviar-la amb la peça reservada. https://tetris.wiki/Hold_piece
- **OpenAPI**: especificació estàndard per a descriure APIs REST de manera llegible per màquina. https://ca.wikipedia.org/wiki/Especificació_OpenAPI
- **Island (Astro)**: component interactiu que es munta sobre una pàgina estàtica en Astro.

8. Bibliografia

- https://tetris.wiki
- https://harddrop.com/wiki/Tetris_Wiki
- https://docs.godotengine.org/en/stable/
- https://fastapi.tiangolo.com/
- https://documentdb.io/docs
- https://docs.astro.build/

9. Annexos

Nota sobre l'ús de la intel·ligència artificial

Durant el desenvolupament d'aquest projecte he utilitzat de manera moderada eines d'intel·ligència artificial, concretament GitHub Copilot i Claude Code.

L'ús ha estat puntual i orientat principalment a ajudes per programar, resolució de dubtes i resolució de bugs.

Tot el codi i totes les decisions tècniques han estat revisades a mà i modificades si fes falta.

Llicència