GameHub Retro — Express + EJS + MongoDB (Mongoose)

Un squelette **prêt à démarrer** pour un dashboard joueur + plateforme de tournois, avec UI **rétro 8-bit** (pixel-art/CRT/scanlines), pages EJS stylisées et logique de bracket simple.

📦Arborescence

```
.
├─ package.json
├─ .env.example
├─ scripts/
│  └─ seed.js
├─ src/
│  ├─ server.js
│  ├─ config/
│  │  └─ db.js
│  ├─ middleware/
│  │  └─ auth.js
│  ├─ models/
│  │  ├─ User.js
│  │  ├─ Game.js
│  │  ├─ Tournament.js
│  │  ├─ Registration.js
│  │  └─ Match.js
│  ├─ controllers/
│  │  ├─ authController.js
│  │  ├─ gameController.js
│  │  └─ tournamentController.js
│  ├─ routes/
│  │  ├─ index.js
│  │  ├─ auth.js
│  │  ├─ games.js
│  │  └─ tournaments.js
│  └─ utils/
│     └─ bracket.js
├─ public/
│  ├─ styles.css
│  └─ main.js
└─ views/
   ├─ layout.ejs
   ├─ 404.ejs
   ├─ home.ejs
   ├─ dashboard.ejs
   ├─ partials/
```

```
    │   ├─ head.ejs
    │   ├─ header.ejs
    │   └─ footer.ejs
    ├─ auth/
    │   ├─ login.ejs
    │   └─ register.ejs
    ├─ games/
    │   └─ index.ejs
    └─ tournaments/
        ├─ index.ejs
        ├─ create.ejs
        └─ show.ejs
```

📁package.json

```
{
  "name": "gamehub-retro",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "nodemon src/server.js",
    "start": "node src/server.js",
    "seed": "node scripts/seed.js"
  },
  "dependencies": {
    "bcrypt": "^5.1.1",
    "connect-mongo": "^5.1.0",
    "cookie-parser": "^1.4.6",
    "dotenv": "^16.4.5",
    "ejs": "^3.1.10",
    "express": "^4.19.2",
    "express-ejs-layouts": "^2.5.1",
    "express-session": "^1.17.3",
    "method-override": "^3.0.0",
    "mongoose": "^8.6.0",
    "morgan": "^1.10.0"
  },
  "devDependencies": {
    "nodemon": "^3.1.10"
  }
}
```

🔐.env.example

```
PORT=3000
MONGODB_URI=mongodb://127.0.0.1:27017/gamehub
SESSION_SECRET=change-me
```

🗜️src/server.js

```
const path = require('path');
const express = require('express');
const morgan = require('morgan');
const session = require('express-session');
const MongoStore = require('connect-mongo');
const cookieParser = require('cookie-parser');
const methodOverride = require('method-override');
require('dotenv').config();

const { connectDB } = require('./config/db');
const indexRoutes = require('./routes/index');
const authRoutes = require('./routes/auth');
const gameRoutes = require('./routes/games');
const tournamentRoutes = require('./routes/tournaments');

const app = express();
connectDB();

const expressLayouts = require('express-ejs-layouts');

app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, '..', 'views'));
app.use(expressLayouts);
app.set('layout', 'layout');

app.use(express.urlencoded({ extended: true }));
app.use(express.json());
app.use(cookieParser());
app.use(methodOverride('_method'));
app.use(morgan('dev'));
app.use('/public', express.static(path.join(__dirname, '..', 'public')));

app.use(
  session({
    secret: process.env.SESSION_SECRET,
    resave: false,
    saveUninitialized: false,
    cookie: { maxAge: 1000 * 60 * 60 * 24 * 7 },
    store: MongoStore.create({ mongoUrl: process.env.MONGODB_URI })
  })
);

// expose user in all views
app.use((req, res, next) => {
  res.locals.currentUserId = req.session.userId || null;
  next();
});

app.use('/', indexRoutes);
app.use('/auth', authRoutes);
app.use('/games', gameRoutes);
app.use('/tournaments', tournamentRoutes);

app.use((req, res) => {
  res.status(404).render('404', { title: '404' });
});

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => console.log(`▶ GameHub Retro running at http://
localhost:${PORT}`));
```

🧰src/config/db.js

```
const mongoose = require('mongoose');

async function connectDB() {
  const uri = process.env.MONGODB_URI;
  if (!uri) throw new Error('MONGODB_URI missing');
  mongoose.set('strictQuery', true);
  await mongoose.connect(uri);
  console.log('✅ MongoDB connected');
}

module.exports = { connectDB };
```

🔒src/middleware/auth.js

```
function ensureAuth(req, res, next) {
  if (req.session && req.session.userId) return next();
  return res.redirect('/auth/login');
}

module.exports = { ensureAuth };
```

🗂️Models (Mongoose)

src/models/User.js

```
const mongoose = require('mongoose');
const bcrypt = require('bcrypt');

const userSchema = new mongoose.Schema(
  {
    name: { type: String, required: true },
    email: { type: String, required: true, unique: true, lowercase: true },
    passwordHash: { type: String, required: true },
    avatarUrl: String,
    favorites: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Game' }]
  },
  { timestamps: true }
);

userSchema.methods.verifyPassword = function (pwd) {
  return bcrypt.compare(pwd, this.passwordHash);
};

module.exports = mongoose.model('User', userSchema);
```

src/models/Game.js

```
const mongoose = require('mongoose');

const gameSchema = new mongoose.Schema(
  {
    name: { type: String, required: true },
    slug: { type: String, required: true, unique: true },
    cover: String,
    genres: [String],
    rawgId: Number
  },
  { timestamps: true }
);

module.exports = mongoose.model('Game', gameSchema);
```

src/models/Tournament.js

```
const mongoose = require('mongoose');

const tournamentSchema = new mongoose.Schema(
  {
    title: { type: String, required: true },
    game: { type: mongoose.Schema.Types.ObjectId, ref: 'Game', required:
true },
    startsAt: { type: Date, required: true },
    maxPlayers: { type: Number, default: 8 },
    status: { type: String, enum: ['draft', 'open', 'running', 'finished'],
default: 'open' },
    createdBy: { type: mongoose.Schema.Types.ObjectId, ref: 'User',
required: true }
  },
  { timestamps: true }
);

module.exports = mongoose.model('Tournament', tournamentSchema);
```

src/models/Registration.js

```
const mongoose = require('mongoose');

const registrationSchema = new mongoose.Schema(
  {
    user: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required:
true },
    tournament: { type: mongoose.Schema.Types.ObjectId, ref: 'Tournament',
required: true },
    seed: Number
  },
  { timestamps: true }
);

registrationSchema.index({ user: 1, tournament: 1 }, { unique: true });

module.exports = mongoose.model('Registration', registrationSchema);
```

src/models/Match.js

```
const mongoose = require('mongoose');

const matchSchema = new mongoose.Schema(
  {
    tournament: { type: mongoose.Schema.Types.ObjectId, ref: 'Tournament',
required: true },
    round: { type: Number, required: true },
    position: { type: Number, required: true },
    playerA: { type: mongoose.Schema.Types.ObjectId, ref: 'Registration' },
    playerB: { type: mongoose.Schema.Types.ObjectId, ref: 'Registration' },
    scoreA: Number,
    scoreB: Number,
    winner: { type: mongoose.Schema.Types.ObjectId, ref: 'Registration' },
    nextMatch: { type: mongoose.Schema.Types.ObjectId, ref: 'Match' }
  },
  { timestamps: true }
);

matchSchema.index({ tournament: 1, round: 1, position: 1 }, { unique:
true });

module.exports = mongoose.model('Match', matchSchema);
```

💎Controllers

src/controllers/authController.js

```
const bcrypt = require('bcrypt');
const User = require('../models/User');

module.exports.showLogin = (req, res) => res.render('auth/login', { title:
'Login' });
module.exports.showRegister = (req, res) => res.render('auth/register', {
title: 'Register' });

module.exports.register = async (req, res) => {
  try {
    const { name, email, password } = req.body;
    const exists = await User.findOne({ email });
    if (exists) return res.render('auth/register', { title: 'Register',
error: 'Email déjà utilisé.' });
    const passwordHash = await bcrypt.hash(password, 10);
    const user = await User.create({ name, email, passwordHash });
    req.session.userId = user._id.toString();
    res.redirect('/dashboard');
  } catch (e) {
    console.error(e);
    res.render('auth/register', { title: 'Register', error: "Erreur
d'inscription." });
  }
};

module.exports.login = async (req, res) => {
  try {
    const { email, password } = req.body;
    const user = await User.findOne({ email });
    if (!user) return res.render('auth/login', { title: 'Login', error:
'Identifiants invalides.' });
    const ok = await user.verifyPassword(password);
    if (!ok) return res.render('auth/login', { title: 'Login', error:
'Identifiants invalides.' });
```

```
    req.session.userId = user._id.toString();
    res.redirect('/dashboard');
  } catch (e) {
    console.error(e);
    res.render('auth/login', { title: 'Login', error: 'Erreur de
connexion.' });
  }
};

module.exports.logout = (req, res) => {
  req.session.destroy(() => res.redirect('/'));
};
```

src/controllers/gameController.js

```
const Game = require('../models/Game');
const User = require('../models/User');

module.exports.list = async (req, res) => {
  const q = (req.query.q || '').trim();
  const filter = q ? { name: new RegExp(q, 'i') } : {};
  const games = await Game.find(filter).sort({ name: 1 }).lean();
  res.render('games/index', { title: 'Games', games, q });
};

module.exports.toggleFavorite = async (req, res) => {
  const gameId = req.params.id;
  const user = await User.findById(req.session.userId);
  if (!user) return res.redirect('/auth/login');
  const idx = user.favorites.findIndex((g) => g.toString() === gameId);
  if (idx >= 0) user.favorites.splice(idx, 1);
  else user.favorites.push(gameId);
  await user.save();
  res.redirect('back');
};
```

src/controllers/tournamentController.js

```
const Tournament = require('../models/Tournament');
const Registration = require('../models/Registration');
const Match = require('../models/Match');
const Game = require('../models/Game');
const { seedRoundAndLink } = require('../utils/bracket');

module.exports.index = async (req, res) => {
  const tournaments = await Tournament.find().populate('game').sort({
createdAt: -1 }).lean();
  res.render('tournaments/index', { title: 'Tournaments', tournaments });
};
```

```
module.exports.showCreate = async (req, res) => {
  const games = await Game.find().sort({ name: 1 }).lean();
  res.render('tournaments/create', { title: 'Create Tournament', games });
};

module.exports.create = async (req, res) => {
  const { title, gameId, startsAt, maxPlayers } = req.body;
  const t = await Tournament.create({
    title,
    game: gameId,
    startsAt: new Date(startsAt),
    maxPlayers: Number(maxPlayers) || 8,
    createdBy: req.session.userId
  });
  res.redirect(`/tournaments/${t._id}`);
};

module.exports.show = async (req, res) => {
  const t = await Tournament.findById(req.params.id).populate('game').lean();
  if (!t) return res.redirect('/tournaments');
  const regs = await Registration.find({ tournament:
t._id }).populate('user').lean();
  const matches = await Match.find({ tournament: t._id }).populate({ path:
'playerA playerB winner', populate: { path: 'user' } }).lean();
  res.render('tournaments/show', { title: t.title, t, regs, matches });
};

module.exports.register = async (req, res) => {
  const t = await Tournament.findById(req.params.id);
  if (!t) return res.redirect('/tournaments');
  const count = await Registration.countDocuments({ tournament: t._id });
  if (count >= t.maxPlayers) return res.redirect(`/tournaments/${t._id}`);
  try {
    await Registration.create({ tournament: t._id, user:
req.session.userId });
  } catch (e) {
    // already registered
  }
  res.redirect(`/tournaments/${t._id}`);
};

module.exports.seed = async (req, res) => {
  const tId = req.params.id;
  const regs = await Registration.find({ tournament: tId }).sort({
createdAt: 1 });
  await seedRoundAndLink(tId, regs);
  res.redirect(`/tournaments/${tId}`);
};

module.exports.report = async (req, res) => {
  const matchId = req.params.id;
  const { scoreA, scoreB } = req.body;
  const m = await Match.findById(matchId);
  if (!m) return res.redirect('back');
  m.scoreA = Number(scoreA);
  m.scoreB = Number(scoreB);
  if (m.playerA && m.playerB) {
    m.winner = m.scoreA > m.scoreB ? m.playerA : m.playerB;
  }
  await m.save();
  if (m.nextMatch && m.winner) {
    const next = await Match.findById(m.nextMatch);
    if (next) {
      // place winner in next slot
      if (!next.playerA) next.playerA = m.winner;
      else if (!next.playerB) next.playerB = m.winner;
      await next.save();
    }
  }
  res.redirect('back');
};
```

🪠 Routes

src/routes/index.js

```
const express = require('express');
const router = express.Router();
const { ensureAuth } = require('../middleware/auth');
const User = require('../models/User');
const Tournament = require('../models/Tournament');
const Registration = require('../models/Registration');

router.get('/', (req, res) => res.render('home', { title: 'GameHub
Retro' }));

router.get('/dashboard', ensureAuth, async (req, res) => {
  const user = await
User.findById(req.session.userId).populate('favorites').lean();
  const regs = await Registration.find({ user: user._id }).populate({ path:
'tournament', populate: { path: 'game' } }).lean();
  const upcoming = regs.filter(r => r.tournament && r.tournament.startsAt &&
new Date(r.tournament.startsAt) > new Date());
  res.render('dashboard', { title: 'Dashboard', user, upcoming });
});

module.exports = router;
```

src/routes/auth.js

```js
const express = require('express');
const router = express.Router();
const ctrl = require('../controllers/authController');

router.get('/login', ctrl.showLogin);
router.post('/login', ctrl.login);
router.get('/register', ctrl.showRegister);
router.post('/register', ctrl.register);
router.post('/logout', ctrl.logout);

module.exports = router;
```

src/routes/games.js

```js
const express = require('express');
const router = express.Router();
const { ensureAuth } = require('../middleware/auth');
const ctrl = require('../controllers/gameController');

router.get('/', ensureAuth, ctrl.list);
router.post('/:id/favorite', ensureAuth, ctrl.toggleFavorite);

module.exports = router;
```

src/routes/tournaments.js

```js
const express = require('express');
const router = express.Router();
const { ensureAuth } = require('../middleware/auth');
const ctrl = require('../controllers/tournamentController');

router.get('/', ctrl.index);
router.get('/new', ensureAuth, ctrl.showCreate);
router.post('/', ensureAuth, ctrl.create);
router.get('/:id', ctrl.show);
router.post('/:id/register', ensureAuth, ctrl.register);
router.post('/:id/seed', ensureAuth, ctrl.seed); // simple admin-free for MVP
router.post('/matches/:id/report', ensureAuth, ctrl.report);

module.exports = router;
```

🧮src/utils/bracket.js

```
const Match = require('../models/Match');

// Pairings: (1 vs N), (2 vs N-1), ... with BYE advancing automatically
async function seedRoundAndLink(tournamentId, registrations) {
  // cleanup previous matches
  await Match.deleteMany({ tournament: tournamentId });

  const N = registrations.length;
  const seeds = registrations.map((r, i) => ({ reg: r, seed: i + 1 }));

  // compute nearest power of two >= N
  const pow2 = 1 << Math.ceil(Math.log2(Math.max(1, N)));
  const byes = pow2 - N;

  // round 1
  const r1 = [];
  for (let i = 0; i < pow2 / 2; i++) {
    const sA = i + 1;
    const sB = pow2 - i;
    const playerA = seeds.find((s) => s.seed === sA)?.reg || null;
    const playerB = seeds.find((s) => s.seed === sB)?.reg || null;
    r1.push({ round: 1, position: i + 1, playerA, playerB });
  }

  const createdR1 = await Match.insertMany(
    r1.map((m) => ({
      tournament: tournamentId,
      round: 1,
      position: m.position,
      playerA: m.playerA?._id,
      playerB: m.playerB?._id
    }))
  );

  // create subsequent rounds and link nextMatch pointers
  let prevRound = createdR1;
  let round = 2;
  let size = pow2 / 4; // matches in round 2
  while (size >= 1) {
    const created = await Match.insertMany(
      Array.from({ length: size }).map((_, i) => ({ tournament:
tournamentId, round, position: i + 1 }))
    );

    // link prev matches -> next
    for (let i = 0; i < prevRound.length; i += 2) {
      const next = created[i / 2];
      prevRound[i].nextMatch = next._id;
      prevRound[i + 1].nextMatch = next._id;
    }
    await Promise.all(prevRound.map((m) => m.save()));

    prevRound = created;
    round += 1;
    size = size / 2;
  }

  // auto-advance BYEs: if one player null, the other moves to next
  const firstRound = await Match.find({ tournament: tournamentId, round:
1 });
  for (const m of firstRound) {
    if (m.playerA && !m.playerB) {
      m.winner = m.playerA;
      await m.save();
      if (m.nextMatch) {
        const nx = await Match.findById(m.nextMatch);
        if (nx) {
          if (!nx.playerA) nx.playerA = m.winner; else if (!nx.playerB)
nx.playerB = m.winner;
          await nx.save();
        }
      }
    } else if (m.playerB && !m.playerA) {
      m.winner = m.playerB;
      await m.save();
      if (m.nextMatch) {
        const nx = await Match.findById(m.nextMatch);
        if (nx) {
          if (!nx.playerA) nx.playerA = m.winner; else if (!nx.playerB)
nx.playerB = m.winner;
          await nx.save();
        }
      }
    }
  }
}

module.exports = { seedRoundAndLink };
```

📖public/styles.css (thème rétro 8-bit)

```
:root{
  --bg:#0a0c10;           /* presque noir */
  --fg:#e8f0ff;           /* blanc bleuté */
  --neon:#00f7ff;         /* cyan néon */
  --magenta:#ff2fb9;      /* magenta néon */
```

```css
  --yellow:#ffd400;      /* jaune 8-bit */
  --tile:#11131a;        /* tuile sombre */
  --grid:#1a1d27;
}

/* Reset minimal */
*{box-sizing:border-box}
html,body{height:100%}
body{
  margin:0; background:var(--bg); color:var(--fg);
  font-family: 'Press Start 2P', system-ui, -apple-system, Segoe UI, Roboto,
sans-serif;
  line-height:1.5; overflow-x:hidden; position:relative;
}

/* Scanlines + vignette CRT */
body::before{
  content:""; position:fixed; inset:0; pointer-events:none; z-index:10;
  background:
    repeating-linear-gradient(0deg, rgba(255,255,255,.06), rgba(255,255,255,.
06) 1px, transparent 2px, transparent 3px),
    radial-gradient(100% 100% at 50% 50%, transparent 60%, rgba(0,0,0,.35));
  mix-blend-mode:overlay;
}

/* Grille pixelisée */
.bg-grid{
  background-image: linear-gradient(var(--grid) 1px, transparent 1px),
linear-gradient(90deg, var(--grid) 1px, transparent 1px);
  background-size: 24px 24px, 24px 24px;
}

/* Parallax couches */
.parallax{position:relative; perspective:1px; height:60vh; overflow:hidden}
.layer{position:absolute; inset:0; transform-style:preserve-3d}
.layer--far{transform: translateZ(-2px) scale(3)}
.layer--mid{transform: translateZ(-1px) scale(2)}
.layer--near{transform: translateZ(-.5px) scale(1.5)}

/* Boutons 8-bit */
.btn-8{
  display:inline-block; padding:.9rem 1.2rem; text-decoration:none;
color:#000;
  background:linear-gradient(#fff, #cfd9ff); border:4px solid #000; box-
shadow:6px 6px 0 #000;
}
.btn-8:hover{transform:translate(-2px,-2px); box-shadow:8px 8px 0 #000}

/* Cartouche style */
.cartridge{
  background:linear-gradient(180deg, #d3d7e0, #b3b8c5);
  border:4px solid #111; border-radius:10px; box-shadow:8px 8px 0 #000;
padding:12px;
}
.cartridge .label{background:linear-gradient(#ff4d9b, #b3005c); color:white;
padding:8px 10px; display:inline-block; border-radius:6px}

/* Cards */
.card{background:var(--tile); border:2px solid #000; box-shadow:4px 4px 0
#000; padding:14px}

/* Links */
.a-arcade{color:var(--neon); text-decoration:none}
.a-arcade:hover{text-shadow:0 0 8px var(--neon)}

/* Grid helpers */
.wrap{max-width:1100px; margin:0 auto; padding:24px}
.grid{display:grid; gap:16px}
.grid-3{grid-template-columns:repeat(3, minmax(0,1fr))}
@media (max-width:900px){.grid-3{grid-template-columns:1fr}}

/* Badges */
.badge{display:inline-block; padding:6px 10px; border:2px solid #000;
background:#222; color:#fff}

/* Bracket */
.bracket{display:grid; grid-auto-flow:column; grid-auto-columns:1fr; gap:
16px; align-items:start}
.round{display:flex; flex-direction:column; gap:12px}
.match{background:#0f1320; border:2px solid #000; box-shadow:4px 4px 0 #000;
padding:10px}
.match h4{margin:0 0 6px}

/* Header/nav */
.nav{
  position:sticky; top:0; z-index:20; backdrop-filter:blur(6px);
  background:rgba(10,12,16,.8); border-bottom:2px solid #000;
}
.nav a{color:var(--fg); text-decoration:none; padding:12px; display:inline-
block}
.nav a:hover{color:var(--yellow)}

/* Hero */
.hero{position:relative; display:flex; align-items:center; min-height:70vh}
.hero-title{font-size:clamp(24px,4vw,48px); text-shadow:4px 4px 0 #000}
.sub{opacity:.8}

/* Forms */
input, select{padding:10px; border:2px solid #000; background:#161a25;
color:#fff}
label{font-size:12px}
```

💸public/main.js (parallax + petits effets)

```
// Parallax effect on mouse move for layers
const layers = document.querySelectorAll('.layer');
window.addEventListener('mousemove', (e) => {
  const x = (e.clientX / window.innerWidth - 0.5) * 2;
  const y = (e.clientY / window.innerHeight - 0.5) * 2;
  layers.forEach((l, i) => {
    const depth = (i + 1) * 4;
    l.style.transform = `translate3d(${x * depth}px, ${y * depth}px, 0)`;
  });
});

// Tiny blink on links
document.querySelectorAll('a').forEach((a)=>{
  a.addEventListener('mouseenter', ()=> a.style.filter = 'brightness(1.3)');
  a.addEventListener('mouseleave', ()=> a.style.filter = '');
});
```

🧱views/partials/head.ejs

```
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title><%= title || 'GameHub Retro' %></title>
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?
family=Press+Start+2P&family=VT323&display=swap" rel="stylesheet">
<!-- Utilitaires Tailwind (CDN) pour prototyper rapidement -->
<script src="https://cdn.jsdelivr.net/npm/@tailwindcss/browser@4"></script>
<link rel="stylesheet" href="/public/styles.css" />
```

🪠views/partials/header.ejs

```
<header class="nav">
  <div class="wrap flex items-center justify-between">
    <a class="text-xl hero-title" href="/">GAMEHUB<span style="color:var(--
magenta)">•RETRO</span></a>
    <nav class="flex items-center gap-2">
      <a href="/games">Games</a>
      <a href="/tournaments">Tournaments</a>
      <% if (currentUserId) { %>
        <a href="/dashboard">Dashboard</a>
        <form method="post" action="/auth/logout" class="inline"><button
class="badge" type="submit">Logout</button></form>
      <% } else { %>
        <a href="/auth/login">Login</a>
        <a href="/auth/register">Sign up</a>
      <% } %>
    </nav>
  </div>
</header>
```

🧱views/partials/footer.ejs

```
<footer class="wrap sub" style="margin-top:40px">
  <p>© <%= new Date().getFullYear() %> GameHub Retro — Built with Express +
EJS + MongoDB</p>
</footer>
<script src="/public/main.js" defer></script>
```

🧱views/layout.ejs

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <% include partials/head %>
  </head>
  <body class="bg-grid">
    <% include partials/header %>
    <main class="wrap"><%- body %></main>
    <% include partials/footer %>
  </body>
</html>
```

💸views/home.ejs

```
<section class="hero">
  <div class="parallax" aria-hidden>
    <div class="layer layer--far" style="background: radial-gradient(100%
100% at 50% 50%, #0a0c10 40%, rgba(0,247,255,.2));"></div>
    <div class="layer layer--mid" style="background-image:url('https://
upload.wikimedia.org/wikipedia/commons/5/59/Space_Invaders_alien.png');
background-repeat:repeat; opacity:.05"></div>
    <div class="layer layer--near"></div>
  </div>
  <div>
    <h1 class="hero-title">Bienvenue sur <span style="color:var(--
yellow)">GameHub Retro</span></h1>
    <p class="sub">Dashboard joueur + tournois, en mode arcade 8-bit.</p>
    <div class="mt-6 flex gap-3">
      <a class="btn-8" href="/games">▶ Start: Games</a>
      <a class="btn-8" href="/tournaments">🔵 Tournois</a>
    </div>
  </div>
</section>
```

---

views/dashboard.ejs

```
<h1 class="hero-title">Dashboard</h1>
<% if (user) { %>
<div class="grid grid-3 mt-4">
  <div class="card">
    <h3>Profil</h3>
    <p><strong><%= user.name %></strong></p>
    <p class="sub"><%= user.email %></p>
  </div>
  <div class="card">
    <h3>Favoris</h3>
    <ul>
      <% user.favorites.forEach(g => { %>
        <li><a class="a-arcade" href="#"><%= g.name %></a></li>
      <% }) %>
      <% if (!user.favorites.length) { %><li>Aucun favori pour l'instant.</
li><% } %>
    </ul>
  </div>
  <div class="card">
    <h3>Prochains matchs</h3>
    <ul>
      <% upcoming.forEach(r => { %>
        <li>Tournoi <strong><%= r.tournament.title %></strong> — <%= new
Date(r.tournament.startsAt).toLocaleString('fr-FR') %></li>
      <% }) %>
      <% if (!upcoming.length) { %><li>Aucun match planifié.</li><% } %>
    </ul>
  </div>
</div>
<% } %>
```

🔑views/auth/login.ejs

```
<h1 class="hero-title">Login</h1>
<% if (typeof error !== 'undefined') { %>
  <p class="badge" style="border-color:#a00;background:#300"><%= error %></p>
<% } %>
<form method="post" class="grid" style="max-width:420px">
  <label>Email<input type="email" name="email" required /></label>
  <label>Mot de passe<input type="password" name="password" required /></
label>
  <button class="btn-8" type="submit">Se connecter</button>
</form>
```

💻views/auth/register.ejs

```
<h1 class="hero-title">Créer un compte</h1>
<% if (typeof error !== 'undefined') { %>
  <p class="badge" style="border-color:#a00;background:#300"><%= error %></p>
<% } %>
<form method="post" class="grid" style="max-width:420px">
  <label>Nom<input type="text" name="name" required /></label>
  <label>Email<input type="email" name="email" required /></label>
  <label>Mot de passe<input type="password" name="password" required
minlength="6" /></label>
  <button class="btn-8" type="submit">Créer</button>
</form>
```

🕚views/games/index.ejs

```
<h1 class="hero-title">Games — Stage Select</h1>
<form class="mt-3" method="get">
  <input name="q" value="<%= q %>" placeholder="Rechercher un jeu…" />
  <button class="btn-8" type="submit">Go</button>
</form>
<div class="grid grid-3 mt-4">
  <% games.forEach(g => { %>
    <div class="cartridge">
      <div class="label"><%= g.name %></div>
      <% if (g.cover) { %>
        <img src="<%= g.cover %>" alt="<%= g.name %> cover" style="width:
100%; border:2px solid #000"/>
      <% } %>
      <div class="mt-2 flex gap-2">
        <form method="post" action="/games/<%= g._id %>/favorite">
```

```
          <button class="badge" type="submit">★ Favori</button>
        </form>
        <span class="badge"><%= (g.genres||[]).join(', ') %></span>
      </div>
    </div>
  <% }) %>
  <% if (!games.length) { %>
    <p>Aucun jeu trouvé.</p>
  <% } %>
</div>
```

🕖views/tournaments/index.ejs

```
<div class="flex items-center justify-between">
  <h1 class="hero-title">Tournaments</h1>
  <% if (currentUserId) { %><a class="btn-8" href="/tournaments/new">+
Nouveau</a><% } %>
</div>
<div class="grid grid-3 mt-4">
  <% tournaments.forEach(t => { %>
    <div class="card">
      <h3><a class="a-arcade" href="/tournaments/<%= t._id %>"><%= t.title
%></a></h3>
      <p class="sub"><%= t.game?.name %> — <%= new
Date(t.startsAt).toLocaleString('fr-FR') %></p>
      <span class="badge"><%= t.status %></span>
    </div>
  <% }) %>
  <% if (!tournaments.length) { %>
    <p>Pas encore de tournoi.</p>
  <% } %>
</div>
```

---

views/tournaments/create.ejs

```
<h1 class="hero-title">Créer un tournoi</h1>
<form method="post" class="grid" style="max-width:520px">
  <label>Titre<input name="title" required /></label>
  <label>Jeu
    <select name="gameId" required>
      <% games.forEach(g => { %>
        <option value="<%= g._id %>"><%= g.name %></option>
      <% }) %>
    </select>
  </label>
  <label>Date/heure<input type="datetime-local" name="startsAt" required /></
label>
  <label>Max joueurs<input name="maxPlayers" type="number" min="2" step="1"
value="8" /></label>
  <button class="btn-8" type="submit">Créer</button>
</form>
```

🕖views/tournaments/show.ejs

```
<h1 class="hero-title"><%= t.title %> <span class="sub">— <%= t.game?.name
%></span></h1>
<p class="sub">Débute le <%= new Date(t.startsAt).toLocaleString('fr-FR') %>
— Status: <span class="badge"><%= t.status %></span></p>

<div class="grid grid-3 mt-3">
  <div class="card">
    <h3>Participants</h3>
    <ul>
      <% regs.forEach(r => { %>
        <li>🈁 <%= r.user?.name %></li>
      <% }) %>
      <% if (!regs.length) { %><li>Personne (encore) — rejoins !</li><% } %>
    </ul>
    <% if (currentUserId) { %>
      <form method="post" action="/tournaments/<%= t._id %>/register"
class="mt-2">
        <button class="btn-8" type="submit">S'inscrire</button>
      </form>
    <% } %>
    <% if (currentUserId) { %>
      <form method="post" action="/tournaments/<%= t._id %>/seed"
class="mt-2">
        <button class="badge" type="submit">Générer le bracket</button>
      </form>
    <% } %>
  </div>

  <div class="card" style="grid-column:span 2">
    <h3>Bracket</h3>
    <% if (!matches.length) { %>
      <p>Aucun match pour l'instant. Générez le bracket une fois les joueurs
inscrits.</p>
    <% } else { %>
      <div class="bracket">
        <% const maxRound = Math.max(...matches.map(m => m.round)); %>
        <% for (let r = 1; r <= maxRound; r++) { %>
          <div class="round">
            <h4>Round <%= r %></h4>
      <% matches.filter(m => m.round === r).sort((a,b)=>a.position-
b.position).forEach(m => { %>
       <div class="match">
        <h4>Match <%= m.position %></h4>
        <div>🅰️ <%= m.playerA?.user?.name || '— BYE —' %> (<%=
m.scoreA ?? '-' %>)</div>
        <div>🅱️ <%= m.playerB?.user?.name || '— BYE —' %> (<%=
m.scoreB ?? '-' %>)</div>
        <% if (currentUserId && m.playerA && m.playerB) { %>
         <form method="post" action="/tournaments/matches/<%= m._id
%>/report" class="mt-2 flex gap-2">
          <input type="number" name="scoreA" min="0"
placeholder="Score A" required />
          <input type="number" name="scoreB" min="0"
placeholder="Score B" required />
          <button class="badge" type="submit">Reporter</button>
         </form>
        <% } %>
        <% if (m.winner) { %>
         <div class="badge mt-2">Gagnant: <%= m.winner?.user?.name
%></div>
        <% } %>
       </div>
      <% }) %>
     </div>
    <% } %>
   </div>
  <% } %>
 </div>
</div>
```

🗄️ views/404.ejs

```
<h1 class="hero-title">404</h1>
<p>Perdu dans le niveau ? Cette page n'existe pas.</p>
```

🗒️ scripts/seed.js

```
require('dotenv').config();
const { connectDB } = require('../src/config/db');
const User = require('../src/models/User');
const Game = require('../src/models/Game');
const Tournament = require('../src/models/Tournament');
const bcrypt = require('bcrypt');
```

```
(async () => {
  await connectDB();
  console.log('Seeding…');
  await Promise.all([User.deleteMany({}), Game.deleteMany({}),
Tournament.deleteMany({})]);

  const passwordHash = await bcrypt.hash('admin123', 10);
  const user = await User.create({ name: 'Admin', email:
'admin@example.com', passwordHash });

  const games = await Game.insertMany([
    { name: 'Super Pixel Bros', slug: 'super-pixel-bros', genres:
['Platformer'], cover: 'https://upload.wikimedia.org/wikipedia/commons/3/31/
Pixel-heart_2.png' },
    { name: 'Neon Racer 2049', slug: 'neon-racer-2049', genres: ['Racing'] },
    { name: 'Dungeon Quest', slug: 'dungeon-quest', genres: ['RPG'] }
  ]);

  await Tournament.create({
    title: 'Open — Super Pixel Bros',
    game: games[0]._id,
    startsAt: new Date(Date.now() + 864e5),
    maxPlayers: 8,
    createdBy: user._id
  });

  console.log('Done. Login: admin@example.com / admin123');
  process.exit(0);
})();
```

📝Notes

- **Sécurité & Rôles** : pour MVP, tout utilisateur connecté peut générer le bracket et reporter des scores. À raffiner (rôle `admin`/`owner`).
- **RAWG/Steam** : à intégrer plus tard via un service séparé et une page `/games/search` (proxy côté serveur + cache Mongo).
- **Styles** : Tailwind via CDN pour utilitaires rapides + CSS custom pour l'ADN rétro (scanlines, cartouches, boutons 8-bit, parallax).
- **Accessibilité** : textes contrastés, focus visibles (à améliorer selon audits). ```