🥚 Tamagotchi Pro

Un sistema Tamagotchi moderno, completo y profesional desarrollado con tecnologías web actuales. Diseñado para dispositivos móviles y desktop con una experiencia de usuario envolvente y características avanzadas de juego.

✨ Características Principales

🎮 Sistema de Juego Profundo

- **8 Especies Únicas**: Verdania (naturaleza), Ignius (fuego), Aquarina (agua), Voltus (eléctrico), Terralux (tierra), Aerion (aire), Glacius (hielo), Umbra (sombra)
- **5 Etapas de Evolución**: Huevo → Bebé → Adolescente → Adulto → Anciano
- **Sistema de Estadísticas Completo**: Hambre, felicidad, salud, energía, limpieza, inteligencia, fuerza, agilidad
- **Personalidades Dinámicas**: Juguetón, tímido, curioso, perezoso, enérgico, calmado

🏆 Funcionalidades Avanzadas

- **Sistema de Logros**: Desbloquea logros basados en tus acciones y progreso
- **Batallas PvP**: Combate entre criaturas de diferentes usuarios
- **Rankings Globales**: Compite por los primeros puestos en experiencia, nivel y batallas
- **Evolución Automática**: Las criaturas evolucionan cuando cumplen los requisitos
- **Loop de Juego**: Sistema automático que mantiene el mundo vivo 24/7

📱 Experiencia Móvil Optimizada

- **PWA (Progressive Web App)**: Instalable en dispositivos móviles
- **Diseño Responsive**: Optimizado para pantallas táctiles y desktop
- **Glassmorphism UI**: Interface moderna con efectos de cristal
- **Offline Support**: Funcionalidad básica sin conexión a internet

🔒 Seguridad y Rendimiento

- **Autenticación JWT**: Sistema seguro de tokens con refresh automático
- **Rate Limiting**: Protección contra spam y ataques DDoS
- **Sanitización de Datos**: Prevención de ataques XSS y injection

- **Métricas en Tiempo Real**: Monitoreo completo del sistema
- **Logging Avanzado**: Seguimiento de actividad y errores

🛠️ Tecnologías Utilizadas

Backend

- **Node.js** con **TypeScript** para type safety
- **Express.js** como framework web
- **PostgreSQL** como base de datos principal
- **JWT** para autenticación segura
- **Helmet** y middleware de seguridad

Frontend

- **Vanilla JavaScript ES2022** para máximo rendimiento
- **Canvas 2D** para animaciones fluidas
- **CSS Grid/Flexbox** para layouts responsivos
- **Vite** como bundler moderno
- **Service Workers** para funcionalidad PWA

DevOps y Herramientas

- **Docker** para containerización
- **TypeScript** para desarrollo type-safe
- **ESLint** y **Prettier** para calidad de código
- **Jest** para testing unitario

📋 Requisitos del Sistema

Desarrollo

- Node.js 18+
- PostgreSQL 13+
- npm o yarn
- Git

Producción

- Servidor con Node.js
- Base de datos PostgreSQL
- Certificado SSL (recomendado)
- Dominio propio (opcional)

🚀 Instalación y Configuración

1. Clonar el Repositorio

Bash

```bash
git clone <repository-url>
cd tamagotchi-pro
```

2. Configurar Backend

Bash

```bash
cd server
npm install
cp .env.example .env
# Editar .env con tus configuraciones
npm run build
```

3. Configurar Base de Datos

Bash

```bash
# Crear base de datos PostgreSQL
createdb tamagotchi_pro

# Ejecutar migraciones
psql -d tamagotchi_pro -f migrations/001_initial_schema.sql
```

4. Configurar Frontend

Bash

```
cd ../client
npm install
npm run build
```

5. Iniciar en Desarrollo

Bash

```
# Terminal 1 - Backend
cd server
npm run dev

# Terminal 2 - Frontend
cd client
npm run dev
```

🎯 Guía de Uso

Para Jugadores

Crear tu Primera Criatura

1. Regístrate o inicia sesión en la aplicación
2. Haz clic en "Crear Nueva Criatura"
3. Elige una especie y personaliza el nombre
4. ¡Tu huevo aparecerá y comenzará a desarrollarse!

Cuidar tu Criatura

- **Alimentar**: Mantén el hambre por encima del 50%
- **Jugar**: Aumenta la felicidad y gana experiencia
- **Limpiar**: Mantén la higiene para evitar enfermedades
- **Descansar**: Las criaturas necesitan dormir para recuperar energía

Evolución y Crecimiento

- Las criaturas evolucionan automáticamente al cumplir requisitos
- Cada etapa desbloquea nuevas habilidades y apariencias
- El cuidado constante es clave para un desarrollo saludable

Batallas y Competencia

- Desafía a otros jugadores en batallas PvP
- Gana experiencia y sube en los rankings
- Desbloquea logros por tus hazañas

Para Administradores

Panel de Administración

Accede a `/api/admin/` con permisos de administrador para:

- Ver métricas del sistema en tiempo real
- Monitorear usuarios activos y estadísticas
- Revisar logs de seguridad
- Gestionar el rendimiento del servidor

Configuración Avanzada

- Ajustar rate limits en `middleware/security.ts`
- Modificar intervalos del game loop en `utils/gameLoop.ts`
- Personalizar logros en la base de datos
- Configurar notificaciones push

📊 API Documentation

Autenticación

Plain Text

```
POST /api/auth/register - Registrar nuevo usuario
POST /api/auth/login - Iniciar sesión
POST /api/auth/refresh - Renovar token
POST /api/auth/logout - Cerrar sesión
```

Criaturas

Plain Text

```
GET /api/creatures - Obtener criaturas del usuario
POST /api/creatures - Crear nueva criatura
GET /api/creatures/:id - Obtener criatura específica
POST /api/creatures/:id/feed - Alimentar criatura
POST /api/creatures/:id/play - Jugar con criatura
POST /api/creatures/:id/clean - Limpiar criatura
```

Batallas

Plain Text

```
POST /api/battles/start - Iniciar batalla
GET /api/battles/history - Historial de batallas
```

Logros y Rankings

Plain Text

```
GET /api/achievements - Obtener logros disponibles
GET /api/achievements/user - Logros del usuario
GET /api/leaderboard/experience - Ranking por experiencia
GET /api/leaderboard/battles - Ranking por batallas
```

🔧 Configuración Avanzada

Variables de Entorno

Backend (.env)

Plain Text

```
# Base de datos
DATABASE_URL=postgresql://user:password@localhost:5432/tamagotchi_pro
DB_HOST=localhost
DB_PORT=5432
DB_NAME=tamagotchi_pro
DB_USER=your_user
DB_PASSWORD=your_password

# JWT
JWT_SECRET=your-super-secret-jwt-key
JWT_REFRESH_SECRET=your-refresh-secret-key
```

```
JWT_EXPIRES_IN=1h
JWT_REFRESH_EXPIRES_IN=7d

# Servidor
PORT=3000
NODE_ENV=development
FRONTEND_URL=http://localhost:5173

# Administración
ADMIN_USER_IDS=user-id-1,user-id-2

# Límites
UPLOAD_MAX_SIZE=5mb
RATE_LIMIT_WINDOW=15
RATE_LIMIT_MAX=100
```

Frontend (.env)

Plain Text

```
VITE_API_URL=http://localhost:3000/api
VITE_APP_NAME=Tamagotchi Pro
VITE_APP_VERSION=1.0.0
```

Personalización del Juego

Ajustar Dificultad

Modifica los valores en `utils/gameLoop.ts`:

TypeScript

```typescript
const STAT_DEGRADATION = {
  hunger: 2,      // Velocidad de hambre
  happiness: 1,   // Velocidad de felicidad
  energy: 1,      // Velocidad de energía
  cleanliness: 1  // Velocidad de limpieza
};
```

Crear Nuevas Especies

1. Agrega la especie en `shared/types.ts`
2. Actualiza la base de datos con nuevos sprites

3. Modifica `routes/creatures.ts` para incluir la nueva especie

Configurar Logros Personalizados

Edita la tabla `achievements` en PostgreSQL:

SQL

```sql
INSERT INTO achievements (name, description, icon, rarity, conditions)
VALUES ('Master Trainer', 'Crea 10 criaturas', 'trophy', 'rare',
        '{"type": "creature_created", "count": 10}');
```

🧪 Testing

Tests Unitarios

Bash

```bash
cd server
npm test
```

Tests de Integración

Bash

```bash
npm run test:integration
```

Tests E2E

Bash

```bash
cd client
npm run test:e2e
```

Performance Testing

Bash

```bash
npm run test:performance
```

📈 Monitoreo y Métricas

Métricas Disponibles

- **Usuarios Activos**: Usuarios conectados en tiempo real
- **Criaturas Totales**: Número de criaturas vivas en el sistema
- **Tiempo de Respuesta**: Latencia promedio de la API
- **Uso de Memoria**: Consumo de RAM del servidor
- **Acciones Populares**: Estadísticas de uso por funcionalidad

Health Checks

- **Endpoint**: `GET /api/admin/health`
- **Frecuencia**: Cada 30 segundos
- **Alertas**: Automáticas por email/Slack (configurar)

Logs

Los logs se almacenan en:

- **Desarrollo**: Console output
- **Producción**: Archivos rotativos en `/logs/`
- **Errores**: Tracking automático con stack traces

🚀 Despliegue en Producción

Usando Docker

Bash

```bash
# Construir imágenes
docker-compose build

# Iniciar servicios
docker-compose up -d

# Ver logs
docker-compose logs -f
```

Despliegue Manual

Bash

```bash
# Preparar backend
cd server
npm run build
npm start

# Preparar frontend
cd client
npm run build
# Servir dist/ con nginx o servidor web
```

Configuración de Nginx

Plain Text

```
server {
    listen 80;
    server_name your-domain.com;

    location / {
        root /path/to/client/dist;
        try_files $uri $uri/ /index.html;
    }

    location /api/ {
        proxy_pass http://localhost:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

🤝 Contribución

Guías de Desarrollo

1. Fork el repositorio
2. Crea una rama para tu feature: `git checkout -b feature/nueva-funcionalidad`
3. Commit tus cambios: `git commit -m 'Agregar nueva funcionalidad'`
4. Push a la rama: `git push origin feature/nueva-funcionalidad`
5. Abre un Pull Request

Estándares de Código

- Usar TypeScript para type safety
- Seguir convenciones de ESLint
- Escribir tests para nuevas funcionalidades
- Documentar APIs y funciones complejas
- Mantener commits atómicos y descriptivos

Reportar Bugs

Usa el template de issues para reportar bugs:

- Descripción clara del problema
- Pasos para reproducir
- Comportamiento esperado vs actual
- Screenshots si aplica
- Información del entorno

📄 Licencia

Este proyecto está bajo la Licencia MIT. Ver el archivo `LICENSE` para más detalles.

🙏 Agradecimientos

- Inspirado en el Tamagotchi original de Bandai
- Comunidad de desarrolladores de JavaScript/TypeScript
- Contribuidores y testers del proyecto

📞 Soporte

- **Documentación**: Ver carpeta `/docs/`
- **Issues**: GitHub Issues
- **Discusiones**: GitHub Discussions
- **Email**: support@tamagotchi-pro.com

¡Disfruta cuidando tus criaturas virtuales! 🎮✨