Propuesta Completa del Proyecto

Juego tipo Agar.io (blobs que comen comida y compiten) desarrollado con HTML5 (canvas), Node.js + Express + Socket.io y MongoDB.
Incluye modelado de datos (JSON), mapa del sitio, wireframes textuales, eventos clave y notas técnicas.

1. Modelado de Datos (JSON)

Colecciones principales: users, servers, matches, skins.

Ejemplo users:

{
"username": "sebasbose",
"email": "sebas@example.com",
"passwordHash": "...",
"avatar": {"color": "#3AA7FF", "skin": "stripes-01"},
"stats": {"bestMass": 4520, "totalKills": 213},
"leaderboard": [{"score": 4520, "date": "2025-08-01"}]
}

Ejemplo servers:

{
"name": "Server Sebas - Casual",
"hostUserId": "...",
"visibility": "public",
"map": {"width": 4000, "height": 4000, "maxFood": 200}
}

2. API y Eventos Socket.io

REST Endpoints: /api/auth/login, /api/servers, /api/users/:id/profile, etc.
Eventos Socket.io: AUTHENTICATE, JOIN_SERVER, PLAYER_INPUT, SERVER_TICK, PLAYER_DEAD, MATCH_END, CHAT_MESSAGE.
Servidor es authority: clientes solo envían inputs, servidor calcula lógica.

3. Mapa del Sitio

- / (Home / Bienvenida)
- /auth/login, /auth/register, /auth/recover
- /lobby/servers
- /servers/create
- /servers/:id (detalle server)
- /game/:serverId (canvas juego)
- /profile/:username
- /me/settings, /me/servers
- /leaderboard/global
- /help

4. Wireframes (Descripción)

A) Home/Lobby: listado de servidores, botón crear server.
B) Crear Server: formulario (nombre, visibilidad, mapa, max players).
C) Server Detail: lista de jugadores, botón Join, chat opcional.
D) Game Screen: canvas central, HUD con masa, rank, leaderboard.
E) Profile: avatar, estadísticas, leaderboard personal, skins.

5. Flujo del Juego

1. Usuario entra a lobby y elige un server.
2. Cliente autentica con token vía socket.
3. JOIN_SERVER -> servidor valida y spawnea jugador.
4. Cliente envía PLAYER_INPUT, servidor procesa y emite SERVER_TICK.
5. Cliente renderiza en canvas con interpolación.
6. Servidor guarda resúmenes en matches y actualiza leaderboard.

6. Notas Técnicas

- Socket.io para tiempo real (servidor authoritative).
- Guardar solo resúmenes en MongoDB.
- Validar inputs y seguridad con JWT.
- Optimizar canvas con capas y rAF.

7. Breve Explicación del Juego

El juego es un estilo Agar.io donde cada jugador controla un blob que recolecta comida y compite con otros para aumentar de tamaño. Los usuarios pueden crear servidores o unirse a otros, personalizar su blob y registrar estadísticas en su perfil.

Se implementará con HTML5 (canvas) para el frontend, Node.js + Express + Socket.io para el backend y MongoDB para la persistencia. El servidor calcula las colisiones y envía actualizaciones a los clientes en tiempo real para mantener la coherencia del juego.