# 📘 Registro de Prompt y Ejecución de la IA

| **Campo** | **Descripción** |
| --- | --- |
| Prompt N.º | Prompt-1 |
| Autor | Ervin Caravali Ibarra |
| Prompt | READ THE ENTIRE PROJECT you have access to, including: - backend-v1 (contains backend, database, APIs, and technical documentation). - frontend-v2 (contains frontend, APIs, documentation, and configurations).  You must analyze the entire structure, code, database, APIs, documentation, and connections before generating the result. Extract key information such as: - Data entities and database structure. - API endpoints and their parameters. - Relationships between backend and frontend. - Current functionalities and main flows. - Relevant technical documentation.  User Stories (US) must be written strictly based on the information found in the backend and frontend. Do not invent functionalities outside the project analysis.  The functionality must mandatorily include: - Option during user registration to indicate if they have visual difficulties. - Backend field to store that preference. - Automatic change in the frontend to “voice mode” if the user has this preference. - Reading of questions (manual or generated) via Text-to-Speech. - Voice adjustments (voice, speed, volume). - Storage of interaction history in voice mode. - Accessible tutorial in audio format. - Administrative control to configure accessibility.  SPECIAL CONDITION: Once you thoroughly analyze both folders (backend-v1 and frontend-v2), you must clearly define in each User Story where the functionality should be implemented (backend or frontend). This must appear in the support or notes section of each US.  GENERATE: 1️⃣ Eight User Stories (US) in CONESSA format with clear acceptance criteria. 2️⃣ Formal Product Backlog structured as a table. 3️⃣ Formal Release Plan (October 7–21) divided into two sprints with calendar, deliverables, and responsible roles.  Deliver the result in a structured, numbered format without unnecessary additional information. Prioritize clarity, precision, and the use of agile techniques. |
| Función o Propósito | Este prompt tiene como finalidad principal que la IA realice un análisis exhaustivo del proyecto completo (backend y frontend),  identificando estructuras de datos, endpoints, flujos y documentación técnica, para posteriormente generar documentación ágil compuesta por: - Historias de Usuario (HU) en formato CONESSA. - Product Backlog formal con prioridades, estimaciones y responsables. - Plan de Lanzamiento (Release Plan) con calendario y entregables definidos por sprint. |
| Tiempo estimado de ejecución de la IA | 1 minuto y 35 segundos |
| Archivos afectados | - Carpeta backend-v1 y todo su contenido. - Carpeta frontend-v2 y todo su contenido. |
| Archivo(s) generado(s) | Un documento README.md que contiene de manera estructurada y detallada las Historias de Usuario (HU) en formato CONESSA,  el Product Backlog formal con sus prioridades y estimaciones, y el Release Plan completo que abarca del 7 al 21 de octubre. |
| **Campo** | **Descripción** |
| Prompt N.º | Prompt-2 |
| Autor | Ervin Caravali Ibarra |
| Prompt | Create a GitHub Actions workflow called **"Move HU to Done"** that is triggered when a pull request is closed. The workflow must:   1. Extract the User Story (HU) number from the branch name (format: hu-XXX-name), 2. Verify that an issue with that number exists in the repository, 3. If the PR was merged, move the HU to the **"Done"** column of the **"Product Backlog"** project using the GitHub API and CLI, 4. Handle errors if the issue does not exist or is not part of the project, attempting to add it if necessary.   Use environment variables for field and column IDs, and require a token with **write permissions** for issues and projects. |
| Función o Propósito | Automatizar el paso de una HU a "Done" en el tablero de proyecto al cerrar y mergear un PR relacionado. |
| Tiempo estimado de ejecución de la IA | 40 segundos |
| Archivos afectados | move\_hu\_done.yml |
| Archivo(s) generado(s) | Workflow YAML para automatización de gestión de HU en proyectos. |
| **Campo** | **Descripción** |
| Prompt N.º | Prompt-3 |
| Autor | Ervin Caravali Ibarra |
| Prompt | Create a GitHub Actions workflow called **"Move HU to In Progress"** that is triggered when a pull request is opened or updated. The workflow must:   1. Extract the User Story (HU) number from the branch name (format: hu-XXX-name), 2. Verify that an issue with that number exists in the repository, 3. Move the HU to the **"In Progress"** column of the **"Product Backlog"** project using the GitHub API and CLI, 4. Handle errors if the issue does not exist or is not part of the project, attempting to add it if necessary.   Use environment variables for field and column IDs, and require a token with **write permissions** for issues and projects. |
| Función o Propósito | Automatizar el paso de una HU a "In Progress" en el tablero de proyecto al abrir o actualizar un PR relacionado. |
| Tiempo estimado de ejecución de la IA | 40 segundos |
| Archivos afectados | move\_hu\_in\_progress.yml |
| Archivo(s) generado(s) | Workflow YAML para automatización de gestión de HU en proyectos. |
| **Campo** | **Descripción** |
| Prompt N.º | Prompt-4 |
| Autor | Ervin Caravali Ibarra |
| Prompt | Create a GitHub Actions workflow called **"Create Project Backlog and Sprints"** that can be executed manually. The workflow must:   1. Check if the **"Product Backlog"** project exists and create it if it doesn’t, 2. Create priority labels if they don’t exist, 3. Create milestones for backend and frontend sprints, 4. Create issues for each user story (backend and frontend) with title, description, priority, and milestone, 5. Add each issue to the project and move it to the **"To Do"** column, 6. Create branches for each issue and link them.   Use the GitHub CLI and require a token with **write permissions** for issues, projects, and pull requests. |
| Función o Propósito | Automatizar la configuración inicial del backlog, etiquetas, milestones, issues y ramas para un proyecto ágil. |
| Tiempo estimado de ejecución de la IA | 1 minuto |
| Archivos afectados | product\_backlog.yml |
| Archivo(s) generado(s) | Workflow YAML para inicialización ágil de proyectos y backlog. |
| **Campo** | **Descripción** |
| Prompt N.º | Prompt-5 : Prompt-HU8 |
| Autor | Backend Developer (Rol: Especialista en Integración de Tiempo Real) Ervin Caravali Ibarra |
| Prompt | READ THE ENTIRE CONTENT of the backend-v1 folder and the root README.md file. Analyze the backend architecture, WebSocket implementation, and all technical documentation. Your task is to implement the integration of "voice mode" with the game using WebSocket, ensuring that all voice events are processed in real time without degrading the user experience. You must: 1) Identify and document all entry and exit points for voice events, 2) Modify or create controllers/services so that voice events are transmitted and received via WebSocket, 3) Ensure robust, scalable, and low-latency integration, 4) Update the backend README with technical details and usage instructions. Do NOT invent endpoints or flows outside the current project structure. All changes must be strictly based on the code and documentation found in backend-v1. |
|  |  |
| Función o Propósito | Integrar el modo de voz con el flujo de juego en tiempo real usando WebSocket, garantizando robustez y experiencia fluida. |
| Tiempo estimado de ejecución de la IA | 1 minuto |
| Archivos afectados | backend-v1 y README.md |
| Archivo(s) generado(s) | - hybridServer.js (lógica WebSocket y eventos de voz)<br>- voiceController.js (controlador de eventos de voz)<br>- backend-v1/services/voiceService.js (servicio de integración de voz, si aplica)<br>- README.md (documentación técnica actualizada) |
| **Campo** | **Descripción** |
| Prompt N.º | Prompt-6 = HU9 |
| Autor | Backend Developer (Rol: Especialista en Procesamiento de Voz e Integración IA) Ervin Caravali Ibarra |
| Prompt | READ THE ENTIRE CONTENT of the backend-v1 folder and the root README.md file. Analyze the data structures, endpoints, and any existing logic for voice processing. Your task is to implement the processing and validation of voice answers using AssemblyAI. You must: 1) Create or update the endpoint POST /api/voice-responses/validate to receive and validate spoken answers, 2) Integrate AssemblyAI for speech-to-text processing and use the result to validate the user's answer according to game rules, 3) Register every validated interaction in the user's voice history, 4) Document the endpoint, the AssemblyAI integration, and the validation flow in the backend README. Use only the conventions and structure found in the current project. Do NOT invent new flows or endpoints outside the existing architecture. |
| Función o Propósito | Procesar y validar respuestas por voz usando AssemblyAI, registrando la interacción y documentando el flujo en el backend. |
| Tiempo estimado de ejecución de la IA | 1 minuto |
| Archivos afectados | backend-v1 y README.md |
| Archivo(s) generado(s) | voiceResponses.js (definición del endpoint)<br>- voiceController.js (lógica de validación y procesamiento)<br>- assemblyAIService.js (integración con AssemblyAI)<br>- backend-v1/services/voiceHistoryService.js (registro de historial, si aplica)<br>- README.md (documentación técnica actualizada) |

# 

| **Campo** | **Descripción** |
| --- | --- |
| Prompt N.º | Prompt-7 = SU1 |
| Autor | Front-end Developer (Rol: Especialista en Procesamiento de Voz e Integración IA) Wilson Andrés Mosquera zapata |
| Prompt | READ CAREFULLY THE ENTIRE PROJECT YOU HAVE ACCESS TO, including:  - backend-v1 (Node.js + Express + Firebase Auth + Firestore).  - frontend-v2 (React + Vite + TailwindCSS + Firebase Auth).  - README.md (contains the project’s accessibility plan and user stories).  Objective: Implement the accessibility option in the registration form that allows a new player to indicate if they have visual difficulties, ensuring the preference is stored in the database and can be retrieved later.  You must:  1️⃣ Frontend (RegisterPage.jsx & CompleteProfilePage.jsx)  • Add a checkbox labeled “Tengo dificultades visuales” in the registration form.  • Bind its value to a local state variable visualDifficulty.  • Ensure the checkbox is accessible (ARIA labels, keyboard navigation).  • On form submission, include visualDifficulty in the request body.  2️⃣ Backend (usersController.js / Firebase schema)  • Update endpoint POST /api/users/register to accept visualDifficulty: boolean.  • Update endpoint PUT /api/users/profile to allow preference updates.  • Store the field in the Firestore users collection. Default: false.  • Add validation and proper error handling.  3️⃣ Testing & Validation  • Confirm data persistence in Firestore after registration.  • Test with users that select and don’t select the checkbox.  • Ensure backward compatibility for existing users.  Acceptance Criteria:  ✅ The registration form includes a visible and functional “Tengo dificultades visuales” checkbox.  ✅ When submitting, the field visualDifficulty is correctly sent to the backend API.  ✅ The backend stores and retrieves the visualDifficulty value from Firestore.  ✅ Default value is false if not selected.  ✅ The preference can be updated later via /api/users/profile.  ✅ Tested in at least two browsers.  Expected Output:  • Updated RegisterPage.jsx and CompleteProfilePage.jsx with the checkbox and form logic.  • Updated usersController.js and backend routes.  • Updated backend documentation (README.md or Swagger) describing the new field.  • Confirmation logs or screenshots showing the preference stored in Firestore.endpoints outside the existing architecture. |
| Función o Propósito | Permitir que los nuevos jugadores indiquen si tienen dificultades visuales desde el registro y guardar esa preferencia para habilitar accesibilidad personalizada. |
| Tiempo estimado de ejecución de la IA | 3 minutos |
| Archivos afectados | Frontend:  frontend-v2/src/pages/RegisterPage.jsx - Added accessibility checkbox  frontend-v2/src/pages/CompleteProfilePage.jsx - Added preference update  frontend-v2/src/\_\_tests\_\_/RegisterPage.test.jsx - New comprehensive tests  frontend-v2/src/\_\_tests\_\_/CompleteProfilePage.test.jsx - New comprehensive tests  frontend-v2/README.md - Updated documentation  Documentation:  ACCESSIBILITY\_IMPLEMENTATION\_SUMMARY.md - Complete implementation summary |
| Archivo(s) generado(s) | - RegisterPage.jsx actualizado con checkbox de accesibilidad  - CompleteProfilePage.jsx con actualización de preferencia  - usersController.js con manejo de visualDifficulty  - README.md actualizado con documentación del campo |

# 