Documento de Diseño Técnico: Ctrl + Alt + Quest

1.0 Introducción y Visión General del Sistema

1.1. Análisis del Propósito y Alcance

Este documento de diseño técnico establece la guía arquitectónica y las especificaciones fundamentales para el desarrollo de la aplicación **"Ctrl + Alt + Quest"** . El sistema se concibe como una aplicación de escritorio diseñada para gamificar la productividad del usuario, convirtiendo el uso cotidiano del computador en una experiencia motivadora y estructurada. A través de mecánicas de juego como misiones, niveles y recompensas, la aplicación busca mejorar la concentración y fomentar hábitos positivos. Este documento servirá como la guía técnica principal para el equipo de desarrollo, detallando la arquitectura del sistema, el modelo de datos, los componentes de software y los flujos lógicos que darán vida a la visión del proyecto.

1.2. Objetivos del Proyecto

El sistema está diseñado para cumplir con los siguientes objetivos estratégicos:

- **Motivar la productividad y la creación de hábitos positivos:** Transformar tareas rutinarias, tanto digitales como de la vida real, en una experiencia de juego adictiva y divertida, similar a un RPG.
- **Reducir distracciones mediante mecánicas de juego:** Utilizar eventos dinámicos y un sistema de recompensas para mantener al usuario enfocado en sus metas y minimizar la procrastinación.
- **Proporcionar un sistema de monitoreo local y privado:** Registrar la actividad del usuario de forma segura en su propio equipo, sin enviar datos sensibles a servidores externos, garantizando la total privacidad.
- **Fomentar la salud del usuario con pausas activas:** Integrar mecánicas, como las batallas contra jefes (boss fights), que incentiven descansos y movimiento físico para prevenir el agotamiento.

1.3. Público Objetivo

La aplicación está dirigida a usuarios que pasan una cantidad significativa de tiempo frente a un computador y buscan mejorar su concentración y motivación. Los perfiles principales identificados son:

- **Estudiantes:** Que necesitan estructurar sus horas de estudio y evitar distracciones de redes sociales o videos.
- **Profesionales de oficina:** Que buscan optimizar su flujo de trabajo y mantener la constancia en sus tareas diarias.
- **Programadores y Desarrolladores:** Quienes pueden beneficiarse de misiones específicas ligadas a su entorno de desarrollo (IDEs) y métricas de productividad.
- **Escritores y Creadores de Contenido:** Que necesitan mantener rachas de escritura y cumplir con metas de producción de palabras.

1.4. Transición a la Arquitectura

Para lograr estos objetivos de manera efectiva, se ha diseñado una arquitectura de software robusta y específica, orientada a la privacidad y el rendimiento. La siguiente sección detalla esta arquitectura y la pila tecnológica seleccionada.

2.0 Arquitectura General del Sistema

2.1. Contexto de la Arquitectura

La elección de la arquitectura es una decisión estratégica fundamental для el éxito de Ctrl + Alt + Quest. Se ha optado por una **arquitectura de aplicación de escritorio con un enfoque *offline-first*** . Esta decisión responde directamente a los requisitos no funcionales clave del proyecto: la privacidad del usuario (**RNF03**) y la capacidad de operar sin una conexión a internet (**RNF01**). Al procesar y almacenar todos los datos localmente, el sistema garantiza que la información de actividad del usuario nunca abandone su máquina, construyendo una base de confianza y asegurando una funcionalidad completa en cualquier entorno.

2.2. Pila Tecnológica Recomendada

La siguiente tabla detalla las tecnologías sugeridas para cada componente principal del sistema, junto con la justificación de su elección.| Componente | Tecnología Sugerida | Justificación Arquitectónica || ------ | ------ | ------ || **Interfaz Gráfica (Frontend)** | JavaFX | Permite la creación de interfaces de usuario de escritorio ricas, atractivas y personalizables. Su soporte nativo para gráficos y diagramas es ideal para visualizar el progreso del usuario (avatares, dashboards, barras de XP) y crear una experiencia RPG inmersiva. || **Lógica de Negocio (Backend Local)** | Spring Boot | Su contenedor de Inversión de Control (IoC) es ideal para gestionar el ciclo de vida de nuestros componentes modulares como ActivityTracker y MissionEngine. Además, sus capacidades de programación de tareas (@Scheduled) proporcionan una ruta de implementación directa para las verificaciones periódicas requeridas por el BackgroundService. || **Base de Datos Local** | SQLite o H2 | Son bases de datos embebidas, ligeras y basadas en archivos, perfectas para el almacenamiento local y el funcionamiento *offline* . No requieren un servidor separado, simplificando la instalación y garantizando la portabilidad de los datos del usuario. || **Seguridad** | Spring Security | Ofrece un marco de autenticación y control de acceso sólido para proteger los perfiles de usuario locales. Su integración facilita la implementación de prácticas de seguridad estándar, como el *hashing* de contraseñas. || **Monitoreo de SO** | Java Process API / JNA | Estas APIs de Java permiten interactuar con el sistema operativo para obtener información sobre los procesos en ejecución (aplicaciones abiertas). Son esenciales para el monitor de actividad no invasivo y de bajo consumo. |

2.3. Diagrama de Componentes Principales

La interacción entre componentes clave está diseñada para ser desacoplada y eficiente. La **Interfaz Gráfica (JavaFX)** se comunica con el **Backend Local (Spring Boot)** a través de una API de servicios bien definida. A su vez, el backend coordina los módulos de lógica de negocio. El Monitor de Actividad opera como un servicio independiente en segundo plano, publicando datos de actividad (entidades SessionLog) en una cola de eventos interna a la que se suscribe el Motor de Juego. Este patrón desacopla la recolección de datos del

procesamiento de la lógica del juego, garantizando el rendimiento y la modularidad del sistema. Toda la información se persiste de forma segura en la **Base de Datos Local** .

2.4. Transición al Modelo de Datos

Esta arquitectura de componentes se apoya en una estructura de datos bien definida. El siguiente apartado detallará el esquema de la base de datos que soporta las funcionalidades del sistema.

3.0 Modelo de Datos y Esquema de la Base de Datos

3.1. Introducción al Modelo de Datos

El modelo de datos es el núcleo de Ctrl + Alt + Quest, diseñado para almacenar de forma segura, privada y eficiente toda la información relacionada con el perfil del usuario, su actividad en el sistema y su progreso en el juego. Utilizando **PostgreSQL** como motor de referencia para el diseño del esquema, la estructura garantiza la integridad de los datos y la capacidad de generar estadísticas detalladas, todo dentro del entorno local del usuario.

3.2. Análisis del Esquema de la Base de Datos (CtrlAltQuestDB)

A continuación, se analizan las tablas más importantes del esquema.

Tabla: users

Esta es la tabla central del sistema, ya que almacena toda la información del perfil del jugador. Cada registro representa a un usuario único y consolida su progreso general.

- **Campos Clave:**
- username y email: Identificadores únicos para el usuario.
- password_hash: Almacena la contraseña de forma segura mediante un algoritmo de hash, un requisito fundamental de seguridad.
- level, current_xp, total_xp: Cuantifican el progreso del jugador en el sistema de niveles.
- coins: La moneda virtual utilizada para adquirir recompensas en la tienda.
- total_play_time: Un campo de auditoría que registra el tiempo total de uso de la aplicación, útil para estadísticas de compromiso.

Tabla: classes

Define las especializaciones de tipo RPG que un usuario puede elegir, como "Programador" o "Escritor", cada una con beneficios y misiones exclusivas.

- **Campos Clave:**
- name: El nombre de la clase (ej. "Escritor").
- bonus_xp_per_category: Un campo JSONB que permite definir bonificaciones de experiencia flexibles para categorías de tareas específicas (ej. +10% XP por palabras escritas).
- missions_exclusive: Almacena misiones que solo están disponibles para los usuarios de esta clase.

Tablas: devices y network_ips

Estas tablas son cruciales para la auditoría de seguridad y el eventual soporte multi-dispositivo. Registran desde qué equipos y redes se accede a la cuenta del usuario.

- **Propósito:** La tabla devices identifica cada PC donde se instala la aplicación mediante un device_uuid único. La tabla network_ips rastrea las direcciones IP asociadas a cada dispositivo. El sistema puede correlacionar datos de login_logs, devices y network_ips para implementar políticas de seguridad. Por ejemplo, un inicio de sesión exitoso desde un nuevo device_uuid o una dirección IP fuera del country habitual del usuario podría activar una solicitud de confirmación por correo electrónico, proporcionando seguridad proactiva.

Tablas: login_logs y password_history

Conforman el subsistema de seguridad y auditoría, registrando todos los eventos de autenticación y los cambios de credenciales.

- **Propósito:** login_logs guarda un registro de cada intento de inicio de sesión (exitoso o fallido), asociándolo a un usuario y dispositivo. password_history almacena un historial de los hashes de contraseñas anteriores para evitar su reutilización.

Tabla: activity_sessions

Esta es la tabla fundamental para el monitoreo de la actividad. Cada registro representa una sesión de uso del PC, desde que el usuario inicia la aplicación hasta que la cierra.

- **Campos Clave:**
- user_id y device_id: Vinculan la sesión a un usuario y a un dispositivo específico.
- session_start y session_end: Marcan el inicio y fin de la sesión de monitoreo.
- total_screen_time: Calcula la duración total de la sesión, sirviendo como base para muchas métricas de productividad.

Tabla: apps

Funciona como un catálogo de aplicaciones que el sistema puede detectar, permitiendo su clasificación para las misiones.

- **Propósito:** Almacena una lista de aplicaciones conocidas (nombre, ruta del ejecutable) y las clasifica con el booleano is_productive. Esto permite al motor de juego asignar misiones como "Usa una aplicación productiva durante 30 minutos".

Tabla: missions

Es el núcleo de la mecánica de juego. Contiene todas las misiones, ya sean generadas automáticamente por el sistema o creadas manualmente por el usuario.

- **Campos Clave:**
- user_id: Clave foránea que asocia cada misión a un usuario específico, reflejando la naturaleza personalizada del sistema.
- title, category, difficulty: Describen la misión y su nivel de desafío.
- xp_reward, coin_reward: Las recompensas otorgadas al completar la misión.
- conditions: Un campo JSONB de gran importancia, ya que define de manera flexible las condiciones de completitud (ej. {"app": "Excel.exe", "time_in_minutes": 30}).
- is_manual: Bandera booleana que distingue entre misiones generadas por el sistema (automáticas) y tareas creadas por el propio usuario, permitiendo una lógica de juego diferenciada.
- is_daily, is_weekly: Banderas que indican si la misión es parte de los ciclos de juego recurrentes.

3.3. Relaciones Clave (Diagrama de Entidad-Relación Conceptual)

El esquema está altamente relacionado, con la tabla users como el eje central. Un usuario (users) tiene múltiples sesiones de actividad (activity_sessions), cada una asociada a un dispositivo (devices). A su vez, un usuario tiene un conjunto de misiones (missions) asignadas. Esta estructura de relaciones permite construir una visión de 360 grados del usuario, conectando quién es, qué hace, desde dónde lo hace y qué objetivos tiene, todo ello de forma local y privada.

3.4. Transición a los Flujos de Usuario

Este modelo de datos estático cobra vida a través de la interacción del usuario con la aplicación. La siguiente sección detallará los flujos de negocio y procesos que manipulan estos datos para crear una experiencia de juego dinámica.

4.0 Flujos de Lógica de Negocio y Procesos del Sistema

4.1. Introducción a los Flujos de Negocio

Los flujos de negocio son la columna vertebral de la experiencia del usuario, definiendo cómo las interacciones se traducen en acciones lógicas dentro del sistema. Desglosan la funcionalidad en secuencias paso a paso, desde el primer contacto del usuario con la aplicación hasta el bucle de juego diario. Esta sección detalla los procesos clave, basándose en la narrativa visual del diagrama de flujo del sistema.

4.2. Flujo 1: Onboarding y Autenticación del Usuario

Este flujo guía al usuario durante su primer contacto con la aplicación y verifica su identidad en inicios posteriores.

1. **Inicio y Verificación (Start -> CheckAuth):** Al iniciar la aplicación, el sistema verifica si existe un perfil de usuario autenticado localmente. Si existe, accede directamente al Dashboard.
2. **Proceso de Onboarding:** Si no hay un perfil, se inicia la secuencia de Onboarding.
3. **Recolección de Datos (CollectData):** Se solicitan los datos básicos para la creación de la cuenta (usuario, email, contraseña).
4. **Definición de Objetivos (Goals):** El usuario define sus metas de productividad principales.
5. **Selección de Hábitos (SelectHabits):** El usuario elige un conjunto inicial de hábitos o tareas que desea gamificar.
6. **Configuración de Preferencias (SetPreferences):** Se configuran las preferencias de notificaciones, eventos e interrupciones.
7. **Creación del Perfil (CreateProfile):** Con la información recopilada, el sistema crea el registro del usuario en la base de datos local y lo dirige al Dashboard.

4.3. Flujo 2: El Bucle de Juego Principal (Core Loop)

Este conjunto de flujos describe las interacciones diarias del usuario con las mecánicas de gamificación.

Completar Hábitos:

Cuando un usuario marca un hábito o tarea como completado, se desencadena la siguiente secuencia de recompensas:

1. **Procesar Completación (CompleteHabit -> ProcessCompletion):** El sistema registra la finalización de la tarea.
2. **Actualizar Racha (UpdateStreak):** Se actualiza la racha de constancia para ese hábito, un motivador clave.
3. **Calcular y Otorgar XP (CalculateXP -> AwardXP):** Se calcula la experiencia (XP) ganada y se añade al perfil del usuario.
4. **Verificar Subida de Nivel (CheckLevelUp):** El sistema comprueba si la XP acumulada es suficiente para subir de nivel. Si es así, se activa el evento LevelUp.
5. **Mostrar Recompensa de Nivel (ShowLevelUpReward):** En caso de subir de nivel, se notifica al usuario con una recompensa visual.
6. **Desbloquear Logros (CheckAchievement -> UnlockAchievement):** Se verifica si la acción cumple los criterios para desbloquear algún logro. Si es así, se muestra la notificación correspondiente (ShowAchievementNotif).
7. **Actualizar Interfaz (UpdateUI):** La interfaz gráfica se actualiza en tiempo real para reflejar el nuevo nivel, la XP y cualquier logro obtenido.

Gestión de Misiones:

El usuario interactúa con misiones diarias, semanales o especiales para obtener recompensas mayores.

1. **Seleccionar Misión (Daily , Weekly , Special):** El usuario navega por las listas de misiones disponibles.
2. **Reclamar Recompensa (ClaimReward):** Una vez que el sistema detecta que una misión ha sido completada, el usuario puede reclamar su recompensa.
3. **Otorgar Recompensas (AwardCoins , AwardBonusXP):** El sistema añade las monedas y la XP de bonificación al perfil del usuario.

Interacción con la Tienda:

El usuario puede gastar las monedas ganadas en la tienda para adquirir ítems cosméticos o potenciadores.

1. **Seleccionar Ítem (SelectItem):** El usuario elige un objeto de la tienda.
2. **Validar Saldo (CheckBalance):** El sistema verifica si el usuario tiene suficientes monedas.
3. **Confirmar Compra (ConfirmPurchase):** Si el saldo es suficiente, se descuentan las monedas y se añade el ítem al inventario del usuario.
4. **Aplicar Ítem (ApplyItem):** El ítem se aplica al perfil del usuario (ej. un nuevo avatar o un potenciador de XP).

4.4. Flujo 3: Procesos en Segundo Plano (Background Service)

El BackgroundService es un componente crítico que se ejecuta de forma continua mientras la aplicación está abierta. Realiza verificaciones periódicas para mantener el estado del juego actualizado.

- **Verificar Recordatorios (CheckReminders):** Envía notificaciones push del sistema operativo para recordar al usuario las tareas pendientes.
- **Verificar Rachas (CheckStreaks):** Monitoriza la constancia del usuario. Si una racha está en riesgo, envía una advertencia (WarnStreak); si se rompe, la resetea (ResetStreak).

- **Verificar Reset Diario (CheckDaily):** A la medianoche, este proceso se activa para resetear el progreso de las tareas diarias (ResetDaily). Inmediatamente después, invoca al MissionEngine para generar un nuevo conjunto de misiones diarias para el usuario (GenerateNewMissions).

4.5. Transición a los Módulos

Estos flujos de negocio no son monolíticos; son implementados por un conjunto de módulos de software especializados y cohesivos. La siguiente sección describirá en detalle cada uno de estos módulos y sus responsabilidades.

5.0 Desglose de Módulos y Componentes Funcionales

5.1. Introducción a los Módulos

Los módulos de software son los bloques de construcción que encapsulan la lógica de negocio del sistema. Cada módulo es una unidad de software cohesiva, responsable de un conjunto específico de funcionalidades vinculadas directamente a los requisitos del sistema. Este diseño modular promueve la mantenibilidad, la escalabilidad y una clara separación de responsabilidades.

5.2. Análisis de Módulos Clave

Módulo 1: Monitor de Actividad (ActivityTracker)

- **Responsabilidades:** Este módulo es el colector de datos primario. Opera discretamente en segundo plano para registrar métricas de uso como aplicaciones activas y tiempo de pantalla. Su implementación se basará en una estrategia de sondeo de bajo impacto para minimizar el consumo de recursos.
- **Requisitos Asociados:** Implementa el requisito funcional **RF01** y satisface la historia de usuario **HU-001** .

Módulo 2: Motor de Juego (MissionEngine)

- **Responsabilidades:** Funciona como un motor de reglas que traduce los datos del ActivityTracker en progreso. Evalúa continuamente el campo conditions (JSON) de las misiones activas contra el flujo de datos de actividad. Al encontrar una coincidencia, emite un evento de "misión completada" que es procesado por el RewardSystem. También es responsable de generar misiones diarias y semanales.
- **Requisitos Asociados:** Cumple con el requisito **RF02** y la historia de usuario **HU-002** .

Módulo 3: Sistema de Recompensas (RewardSystem)

- **Responsabilidades:** Gestiona la economía interna del juego. Su lógica calcula y otorga recompensas (XP, monedas, ítems, logros) al recibir eventos de finalización de tareas desde el MissionEngine o la UI. También encapsula la lógica de subida de nivel del usuario.
- **Requisitos Asociados:** Satisface el requisito **RF03** y la historia de usuario **HU-006** (basada en **RF04**).

Módulo 4: Gestor de Eventos (HealthGuardian & BossFights)

- **Responsabilidades:** Introduce elementos dinámicos en la experiencia. Este módulo actúa como un programador de eventos que dispara "boss fights" y recordatorios de salud basándose en reglas configurables (ej. tiempo de pantalla continuo). Gestiona el estado de estos eventos y sus recompensas asociadas.
- **Requisitos Asociados:** Implementa el requisito **RF04** y la historia de usuario **HU-004** (basada en **RF05**).

Módulo 5: Panel de Control y Estadísticas (Dashboard)

- **Responsabilidades:** Es el principal módulo de visualización de datos. Agrega la información del progreso del usuario desde la base de datos y la presenta de forma clara y atractiva a través de gráficos, reportes y listas, permitiendo al usuario analizar sus patrones de productividad.
- **Requisitos Asociados:** Cubre el requisito funcional **RF05** y la historia de usuario **HU-005** .

5.3. Transición a Requisitos No Funcionales

Además de la funcionalidad encapsulada en estos módulos, el éxito del sistema depende del cumplimiento de importantes atributos de calidad. La siguiente sección detalla las consideraciones de implementación para satisfacer los requisitos no funcionales del proyecto.

6.0 Consideraciones de Implementación y Requisitos No Funcionales

6.1. Introducción a los Atributos de Calidad

Los requisitos no funcionales (NFRs) son tan cruciales como los funcionales para el éxito del producto. Definen los atributos de calidad del sistema, estableciendo *cómo* debe operar en términos de rendimiento, seguridad y usabilidad, en lugar de *qué* debe hacer. Garantizar su cumplimiento es fundamental para ofrecer una experiencia de usuario robusta, fiable y agradable.

6.2. Estrategias para Cumplir los NFRs

RNF01: Funcionamiento Offline

La arquitectura ha sido diseñada desde su concepción para ser *offline-first* . La estrategia de implementación se basa en el uso de una base de datos local embebida (SQLite o H2) y en contener toda la lógica de negocio dentro de la aplicación de escritorio. Esto garantiza que el 100% de las funcionalidades clave, desde el monitoreo de actividad hasta el sistema de recompensas, operen de manera fluida y completa sin necesidad de una conexión a internet.

RNF02: Rendimiento

El Monitor de Actividad empleará una estrategia de sondeo (polling) de baja sobrecarga, consultando los procesos activos del sistema operativo a un intervalo configurable (por defecto, 30 segundos). Mantendrá un mapa de estado simple en memoria de las aplicaciones activas para minimizar las comprobaciones redundantes y las operaciones de E/S de disco, asegurando que el consumo de CPU se mantenga en niveles insignificantes.

RNF03: Privacidad y Seguridad

La privacidad es un pilar no negociable de este sistema. La estrategia es clara: el monitoreo y el almacenamiento de datos son estrictamente locales. No se enviará ninguna información de actividad del usuario a servidores externos. A nivel de implementación, la seguridad de las credenciales de usuario se garantizará mediante el uso del campo password_hash en la base de datos, que almacenará las contraseñas utilizando un algoritmo de hash robusto, impidiendo su lectura directa.

RNF04: Configurabilidad y Control del Usuario

El usuario debe sentir que tiene el control total sobre su experiencia. La arquitectura permitirá una alta configurabilidad. A través del panel de configuración, el usuario podrá ajustar la frecuencia de las interrupciones, activar un **Modo Concentración** (HU-003) que suprime temporalmente todas las notificaciones y eventos emergentes, y habilitar o deshabilitar por completo ciertos tipos de eventos como las Boss Fights.

RNF05: Usabilidad e Interfaz Intuitiva

La implementación de la UI/UX debe adherirse estrictamente al estilo de *pixel art* establecido en el logo (CtrlAltQuest_logo). Esto se traduce en directivas de diseño concretas: uso de avatares basados en sprites, tipografías de estilo 8-bit para encabezados y elementos de UI, y barras de progreso que se llenan con un patrón de tramado (*dithering*). Esta coherencia visual creará una experiencia retro-RPG cohesiva e inmersiva.

6.3. Transición Final

Este documento de diseño proporciona una base arquitectónica sólida y completa. Con estas especificaciones, el equipo de desarrollo está equipado para construir una aplicación robusta, atractiva y funcional que cumpla con todos los requisitos funcionales y de calidad establecidos para Ctrl + Alt + Quest.