# Estrategias de Arquitectura Agéntica y Prompt Engineering de Alta Densidad para el Desarrollo Automatizado del Last Planner System® en Google Antigravity

## 1. Introducción: El Paradigma del Desarrollo Orientado a Agentes

La evolución de los Entornos de Desarrollo Integrado (IDEs) ha culminado en la aparición de plataformas "Agent-First" como Google Antigravity, las cuales redefinen fundamentalmente la interacción entre el arquitecto de software humano y la inteligencia artificial. A diferencia de los asistentes de codificación tradicionales que operan mediante autocompletado reactivo o chat síncrono, Antigravity introduce un modelo de "Control de Misión" donde agentes autónomos orquestan tareas complejas a través del editor de código, la terminal y el navegador simultáneamente.1 Para satisfacer la premisa de "utilizar la menor cantidad de prompts para lograr el mayor desarrollo posible", es imperativo abandonar la metodología iterativa conversacional en favor de una estrategia de **Inyección de Contexto Masiva**.

El objetivo de este informe es proporcionar una hoja de ruta exhaustiva y una especificación técnica completa para prototipar una aplicación de gestión de construcción basada en el Last Planner System® (LPS). En lugar de formular preguntas al usuario de manera incremental, este documento anticipa y responde a todas las interrogantes arquitectónicas y funcionales necesarias para construir un sistema LPS robusto. Al consolidar la lógica de negocio, el esquema de base de datos y las directrices de interfaz en un "Mega-Prompt" o "Archivo de Especificación Maestra", habilitamos al agente de Antigravity para operar en su "Modo de Planificación" (Planning Mode) con una autonomía casi total, reduciendo drásticamente la intervención humana y los ciclos de corrección.3

El Last Planner System no es meramente una herramienta de calendario; es un sistema de control de producción sociotécnico diseñado para reducir la variabilidad y mejorar la confiabilidad del flujo de trabajo en la construcción.4 Digitalizar este sistema requiere una comprensión profunda de sus cinco conversaciones críticas: Programación Maestra (Lo que se *debe* hacer), Planificación Pull (La lógica inversa), Planificación Lookahead (Lo que se *puede* hacer), Planificación Semanal (Lo que se *hará*) y Aprendizaje (Lo que se *hizo*). Este informe desglosa estas fases en algoritmos programables y estructuras de datos relacionales, proporcionando al agente de IA la "verdad fundamental" necesaria para construir la aplicación sin ambigüedades.

## 2. Mecánica de Google Antigravity: Optimización para la Eficiencia de Prompts

Para maximizar la eficiencia del desarrollo en Antigravity, es crucial comprender cómo la herramienta procesa la información y ejecuta tareas. La plataforma utiliza modelos avanzados como Gemini 3 Pro, que poseen ventanas de contexto masivas, permitiendo la ingestión de documentación técnica extensa antes de escribir una sola línea de código.6

### 2.1. Arquitectura de Control de Misión y Artefactos

La distinción principal de Antigravity radica en su capacidad para generar "Artefactos" verificables antes de la implementación del código. Estos artefactos incluyen Planes de Implementación, Listas de Tareas y Recorridos de Verificación.8 En una estrategia de "pocos prompts", el usuario debe suministrar una especificación tan detallada que el *Plan de Implementación* generado por el agente no requiera revisiones sustanciales. Si la especificación inicial es vaga, el agente se verá obligado a inferir requisitos, lo que inevitablemente conduce a alucinaciones funcionales y a la necesidad de múltiples prompts correctivos. Por lo tanto, la densidad de información en el primer prompt es el factor determinante de la velocidad de desarrollo.

### 2.2. Modos de Operación: Planificación vs. Rápido

Para un sistema complejo como LPS, el "Modo de Planificación" es obligatorio. Este modo instruye al agente para que realice una investigación profunda del contexto proporcionado y estructure su trabajo en grupos de tareas lógicas.3 A diferencia del "Modo Rápido", que está optimizado para refactorizaciones menores, el Modo de Planificación utiliza un "presupuesto de pensamiento" más alto, permitiendo al agente comprender las interdependencias entre las tablas de la base de datos (por ejemplo, cómo una restricción no resuelta en la tabla constraints debe bloquear una transacción en la tabla weekly\_work\_plan). La estrategia aquí descrita se basa exclusivamente en la ejecución dentro del Modo de Planificación para garantizar la integridad estructural del prototipo desde el inicio.

### 2.3. Verificación Autónoma mediante el Navegador

Una capacidad única de Antigravity es su agente de navegación, que puede abrir una instancia de Chrome, interactuar con la aplicación en desarrollo y verificar visualmente la funcionalidad.9 Para minimizar los prompts de depuración manual, las especificaciones proporcionadas al agente deben incluir "Escenarios de Prueba" explícitos (por ejemplo, "Intentar confirmar una tarea con una restricción abierta y verificar que el botón de confirmación esté deshabilitado"). Esto transfiere la carga de la prueba de calidad (QA) del humano al agente, acelerando el ciclo de desarrollo.

## 3. Descodificación del Dominio: Lógica del Last Planner System®

Para que el agente de IA construya una aplicación coherente sin preguntas de seguimiento, debemos definir explícitamente la lógica del dominio. El LPS invierte la planificación tradicional de "empuje" (Push) hacia una planificación de "tracción" (Pull), y la aplicación debe reflejar esto en su modelo de datos y experiencia de usuario.11

### 3.1. Fase 1: Programación Maestra (El Hito Ancla)

En la cúspide de la jerarquía se encuentra el Programa Maestro, que define los hitos contractuales del proyecto. A diferencia de los diagramas de Gantt tradicionales que impulsan las fechas hacia adelante basándose en duraciones estimadas, en LPS, los hitos del Programa Maestro actúan como restricciones inamovibles o "anclas" desde las cuales se tira de la planificación hacia atrás.

* **Implicación para el Agente:** El sistema debe impedir que la lógica de planificación automática mueva estos hitos. Deben tratarse como constantes en la ecuación de programación.

### 3.2. Fase 2: Planificación Pull (Lógica Inversa)

Esta es la fase colaborativa donde los "últimos planificadores" (capataces, subcontratistas) definen la secuencia de trabajo desde el futuro hacia el presente.13

* **Requerimiento Funcional:** Un tablero visual (similar a Kanban pero temporal) donde las tareas se colocan de derecha a izquierda.
* **Lógica Crítica:** La fecha de inicio de una tarea predecesora se calcula restando su duración de la fecha de inicio de su sucesora. El sistema debe manejar dependencias "Fin-a-Inicio" con lógica inversa. Además, cada conexión representa una "negociación" o promesa, no solo una línea de dependencia.

### 3.3. Fase 3: Planificación Lookahead y Gestión de Restricciones (El Firewall)

El Lookahead es el proceso de "hacer listo" (Make Ready) el trabajo. Identifica y elimina restricciones (falta de materiales, información, permisos) antes de que las tareas entren en la ventana de producción inminente (generalmente 6 semanas).14

* **Lógica de Negocio:** Esta es la funcionalidad más crítica para la integridad del sistema. La aplicación debe actuar como un "firewall" o compuerta lógica. Una tarea no puede promoverse al Plan Semanal si tiene una restricción activa marcada como "Abierta".
* **Categorización:** El sistema debe obligar al usuario a categorizar las restricciones (Diseño, Materiales, Prerrequisitos, etc.) para permitir análisis posteriores.

### 3.4. Fase 4: Plan de Trabajo Semanal (El Compromiso)

Aquí, las tareas que han pasado el filtro de restricciones se convierten en "Promesas".

* **Fiabilidad:** La interfaz debe distinguir visualmente entre una "Tarea Planificada" (una intención) y una "Tarea Comprometida" (una promesa). Solo las tareas en el Plan de Trabajo Semanal (WWP) contribuyen al cálculo del PPC.

### 3.5. Fase 5: Métricas y Aprendizaje (PPC y Varianza)

El ciclo se cierra con la medición del desempeño.

* **Cálculo de PPC:** El Porcentaje del Plan Completado se calcula como $\text{PPC} = (\frac{\text{Tareas Completadas}}{\text{Tareas Prometidas}}) \times 100$.16
* **Análisis de Varianza:** Si una tarea no se completa, el sistema debe exigir obligatoriamente la selección de una "Causa de Varianza" de una lista estandarizada.17 Esto es vital para la mejora continua y debe ser una restricción de la interfaz de usuario (no se puede cerrar una tarea como incompleta sin este dato).

## 4. Arquitectura Técnica y Stack Tecnológico Preferente

Para acelerar el prototipado, debemos prescribir un stack tecnológico moderno y cohesivo que el modelo Gemini 3 conozca profundamente. La ambigüedad en la elección de tecnologías es una causa común de "parálisis de decisión" en los agentes.

### 4.1. Selección del Stack

* **Frontend:** React con TypeScript (estándar de la industria, tipado fuerte para reducir errores).
* **Build Tool:** Vite (para velocidad de inicio y HMR).
* **Estilos y UI:** Tailwind CSS combinado con **shadcn/ui**. Esta elección es estratégica: shadcn/ui proporciona componentes accesibles y de alta calidad (como diálogos, tarjetas, tablas) que el agente puede copiar y pegar, acelerando drásticamente la construcción de la interfaz.18
* **Gestión de Estado:** TanStack Query (React Query). Esencial para manejar el estado del servidor, caché y actualizaciones optimistas en un tablero de planificación colaborativo.
* **Backend y Base de Datos:** **Supabase**. Ofrece una base de datos PostgreSQL completa, autenticación lista para usar y, crucialmente, capacidades de "Realtime" (WebSockets) para que múltiples usuarios vean los cambios en el tablero Pull Plan simultáneamente.7
* **Lógica de Arrastrar y Soltar:** dnd-kit. Una librería moderna, ligera y accesible para construir interfaces Kanban complejas, preferible sobre librerías más antiguas como react-beautiful-dnd.

### 4.2. Especificación del Esquema de Base de Datos (PostgreSQL)

Proporcionar el esquema DDL (Data Definition Language) exacto elimina la necesidad de que el agente "imagine" la estructura de datos, previniendo errores de normalización.

**Tabla 1: Estructura del Proyecto y Usuarios**

| **Tabla** | **Columna** | **Tipo** | **Descripción** |
| --- | --- | --- | --- |
| projects | id | UUID (PK) | Identificador único del proyecto. |
|  | name | Text | Nombre del proyecto de construcción. |
|  | start\_date | Date | Fecha de inicio contractual. |
|  | status | Enum | 'active', 'archived', 'planning'. |
| profiles | id | UUID (PK, FK) | Vinculado a auth.users de Supabase. |
|  | role | Enum | 'superintendent', 'foreman', 'project\_manager'. |
|  | company\_name | Text | Empresa subcontratista asociada. |

**Tabla 2: Núcleo de Planificación (Last Planner)**

| **Tabla** | **Columna** | **Tipo** | **Descripción** |
| --- | --- | --- | --- |
| milestones | id | UUID (PK) | Hito del programa maestro. |
|  | project\_id | UUID (FK) | Relación con el proyecto. |
|  | date | Date | Fecha inamovible del hito. |
|  | type | Enum | 'phase\_start', 'phase\_complete', 'turnover'. |
| tasks | id | UUID (PK) | La unidad atómica de trabajo. |
|  | title | Text | Descripción de la actividad. |
|  | duration | Integer | Duración en días laborales. |
|  | status | Enum | 'planned', 'make\_ready', 'committed', 'complete', 'incomplete'. |
|  | wbs\_code | Text | Código de estructura de desglose de trabajo.21 |
|  | assigned\_to | UUID (FK) | Usuario o cuadrilla responsable. |
|  | planned\_percent | Integer | Porcentaje de avance físico. |

**Tabla 3: Lógica de Restricciones y Varianzas**

| **Tabla** | **Columna** | **Tipo** | **Descripción** |
| --- | --- | --- | --- |
| constraints | id | UUID (PK) | Restricción que bloquea una tarea. |
|  | task\_id | UUID (FK) | Tarea afectada. |
|  | category | Enum | 'design', 'material', 'labor', 'equipment', 'permit'. |
|  | status | Enum | 'open', 'resolved'. |
|  | due\_date | Date | Fecha límite para resolver la restricción. |
| variances | id | UUID (PK) | Registro de fallo en cumplimiento (PPC). |
|  | task\_id | UUID (FK) | Tarea fallida. |
|  | reason\_code | Enum | 'weather', 'manpower', 'prerequisite', 'design\_change'. |
|  | root\_cause | Text | Explicación detallada para el aprendizaje. |

**Tabla 4: Dependencias (Grafo de Red)**

| **Tabla** | **Columna** | **Tipo** | **Descripción** |
| --- | --- | --- | --- |
| dependencies | pred\_id | UUID (FK) | Tarea predecesora. |
|  | succ\_id | UUID (FK) | Tarea sucesora. |
|  | lag | Integer | Días de desfase (si aplica). |

## 5. El "Mega-Prompt" de Especificación Maestra

Para resolver el prototipo con la "menor cantidad de prompts", la estrategia consiste en **no escribir el prompt en el chat**, sino crear un archivo de contexto dentro del espacio de trabajo de Antigravity. Esto permite que el agente consulte una fuente de verdad inmutable y estructurada.

**Instrucción para el Usuario:**

1. Inicie Antigravity y abra una nueva carpeta de proyecto.
2. Cree un archivo llamado SPECIFICATIONS.md.
3. Copie y pegue el siguiente contenido íntegramente en ese archivo. Este contenido representa la destilación de todas las funcionalidades requeridas, presentadas como directivas técnicas.

### Contenido del Archivo SPECIFICATIONS.md

# ESPECIFICACIÓN TÉCNICA: SISTEMA DE GESTIÓN LAST PLANNER (LPS)

## 1. Visión General del Producto

Construir una aplicación web Full-Stack para la gestión de construcción Lean, implementando estrictamente la metodología Last Planner System®. El sistema debe facilitar la planificación colaborativa (Pull Planning), la gestión de restricciones (Make Ready) y el seguimiento de la fiabilidad (PPC).

## 2. Stack Tecnológico (Obligatorio)

* **Frontend:** React (TypeScript) + Vite.
* **UI Framework:** Tailwind CSS + shadcn/ui (componentes Radix).
* **Iconografía:** Lucide-React.
* **Gestión de Estado:** TanStack Query v5.
* **Drag & Drop:** dnd-kit (Core, Sortable, Modifiers).
* **Backend/DB:** Supabase (PostgreSQL).
* **Testing:** Vitest (Unitario), Playwright (E2E para verificación de agente).

## 3. Definiciones de Base de Datos (DDL Lógico)

El agente debe generar migraciones SQL para Supabase basadas en este esquema:

### 3.1 Entidades Principales

* **projects**: id, name, start\_date, end\_date, created\_at.
* **milestones**: id, project\_id, title, date, color\_hex.
* **tasks**: id, project\_id, title, trade (text), start\_date, end\_date, duration (int), status (enum: 'planned', 'ready', 'committed', 'done', 'failed'), is\_milestone (bool).
* **constraints**: id, task\_id, description, category (enum: 'Material', 'Design', 'Labor', 'Access', 'Permits'), status (enum: 'open', 'resolved'), due\_date.
* **variances**: id, task\_id, reason (enum: 'Weather', 'Manpower', 'Design\_Issue', 'Material\_Delay', 'Prerequisite\_Work'), notes, recorded\_at.

### 3.2 Relaciones y Seguridad

* Implementar **Row Level Security (RLS)**: Los datos deben ser públicos para lectura/escritura por ahora (prototipo rápido), o restringidos a usuarios autenticados si se configura Auth.
* **Dependencies Table**: id, predecessor\_task\_id, successor\_task\_id, type ('FS', 'SS').

## 4. Requerimientos Funcionales Detallados

### Módulo A: Tablero de Planificación Pull (Fase Inversa)

* **Interfaz:** Un tablero Kanban horizontal, pero organizado por SEMANAS (Weeks) en lugar de estados.
* **Interacción:**
  + Permitir arrastrar tareas (tasks) entre columnas de semanas.
  + Al soltar una tarea, actualizar su start\_date y end\_date basado en la semana.
  + **Lógica de Validación:** Si una tarea se mueve a una fecha posterior a su sucesora definida en dependencies, mostrar una alerta visual (borde rojo) indicando "Quiebre de Secuencia".

### Módulo B: Gestión de Restricciones (Compuesto "Make Ready")

* **Vista Lookahead:** Una lista filtrada que muestra solo tareas con fecha de inicio en las próximas 6 semanas.
* **Indicador Visual:** Cada tarjeta de tarea debe tener un icono de "Alerta" si existen registros en la tabla constraints con status = 'open'.
* **Lógica de Negocio Crítica (The Gatekeeper):**
  + El sistema debe **BLOQUEAR** la acción de cambiar el estado de una tarea a 'committed' (Comprometida para el Plan Semanal) si tiene restricciones abiertas.
  + Debe aparecer un mensaje de error: "No se puede comprometer una tarea con restricciones pendientes. Resuelva primero."

### Módulo C: Ejecución y Varianza (Ciclo de Aprendizaje)

* **Vista de Plan Semanal:** Muestra solo tareas con status = 'committed'.
* **Registro de Estado:**
  + Botón "Completar": Cambia estado a 'done'.
  + Botón "Fallo/Incompleto": **DEBE** abrir un Modal (Dialog).
  + **Modal de Varianza:** Obliga al usuario a seleccionar un reason de la lista enum y escribir una nota antes de guardar el estado como 'failed'.

### Módulo D: Tablero de Métricas (Analytics)

* **Gráfico de PPC:** Usar recharts para mostrar una línea de tendencia del Porcentaje de Plan Completado por semana.
  + Fórmula: (Tareas 'done' / Total Tareas ('done' + 'failed')) \* 100.
* **Gráfico de Pareto:** Barras mostrando la frecuencia de cada reason en la tabla variances.

## 5. Directrices de UI/UX (Shadcn/UI)

* Usar el componente Card para las tareas.
* Usar Badge para el estado y categoría de restricción.
* Usar Sheet (Sidepanel) para editar los detalles de la tarea.
* Usar DatePicker para selección de fechas.
* Tema: "Slate" o "Zinc" para un aspecto profesional industrial.

## 6. Instrucciones de Verificación para el Agente

Una vez implementado el código, el agente debe:

1. Crear datos semilla (Seed Data): Un proyecto, 3 hitos, 5 tareas conectadas, 1 restricción abierta.
2. Abrir el navegador.
3. Intentar mover la tarea con restricción al estado 'committed' y verificar que falle.
4. Resolver la restricción y reintentar (debe tener éxito).
5. Marcar una tarea como fallida y verificar que aparezca el modal de varianza.

## 6. Protocolo de Ejecución: El Prompt Activador

Una vez que el archivo SPECIFICATIONS.md reside en el directorio raíz del proyecto, el usuario debe abrir el "Agent Manager" (Control de Misión) y enviar **un único prompt de activación**. Este prompt no detalla los requisitos (ya que están en el archivo), sino que asigna el rol y la metodología de ejecución al agente.

**Prompt de Activación (Copiar y Pegar en Antigravity):**

"Actúa como Ingeniero de Software Principal experto en React y Lean Construction. Tu objetivo es implementar la aplicación completa descrita en el archivo SPECIFICATIONS.md que se encuentra en el directorio raíz.

**Directivas de Ejecución:**

1. **Análisis:** Lee exhaustivamente SPECIFICATIONS.md para entender el modelo de datos y la lógica de negocio (especialmente el bloqueo de restricciones y la lógica de varianza).
2. **Infraestructura:** Inicializa el proyecto con Vite/React/TypeScript y configura Tailwind con shadcn/ui.
3. **Backend:** Genera los archivos de migración SQL para Supabase reflejando exactamente el esquema definido en la especificación.
4. **Implementación:** Construye los 4 módulos funcionales (Pull Plan, Restricciones, Plan Semanal, Métricas). Prioriza la funcionalidad del 'Gatekeeper' de restricciones.
5. **Verificación:** Ejecuta el servidor localmente y utiliza el navegador para realizar la rutina de prueba descrita en la sección 6 de las especificaciones.

Ejecuta esto en **Modo de Planificación (Planning Mode)**. No solicites aclaraciones sobre el diseño; utiliza las definiciones explícitas del archivo de especificación para resolver cualquier ambigüedad. Comienza generando el Plan de Implementación completo."

## 7. Análisis de Brechas y Soluciones Arquitectónicas

A continuación se detalla cómo esta estrategia cubre requisitos que usualmente se pierden en prompts menos específicos, integrando la información de las fuentes de investigación para asegurar robustez.

### 7.1. El Problema de la Sincronización de Datos (Gap Identificado)

Las herramientas de LPS requieren colaboración en tiempo real. Si dos capataces mueven tareas en el tablero Pull Plan simultáneamente, pueden ocurrir conflictos.

* **Solución Integrada:** Al especificar **Supabase Realtime** en el stack tecnológico 18, instruimos implícitamente al agente para que utilice suscripciones de base de datos. Esto permite que el estado del tablero se actualice instantáneamente sin recargar la página, una característica crítica para la "Planificación Colaborativa" mencionada en la literatura de LPS.22

### 7.2. La Lógica del "Gatekeeper" de Restricciones

La investigación indica que el fallo más común en la implementación de LPS es permitir que tareas no preparadas entren en el flujo de trabajo.23

* **Solución Integrada:** El prompt especifica una lógica de "Gatekeeper" (Portero). No es solo una característica de UI; es una regla de negocio que impide la corrupción del Plan Semanal. Al definir esto como "Lógica de Negocio Crítica" en el archivo markdown, elevamos su prioridad para el agente de IA, asegurando que escriba condicionales estrictos en el código (e.g., if (hasOpenConstraints) return;).

### 7.3. Taxonomía de Varianzas para el Aprendizaje

Simplemente marcar una tarea como "fallida" no aporta valor Lean. Se requiere saber *por qué* falló.

* **Solución Integrada:** La especificación incluye un Enum explícito para variances basado en las categorías estándar de la industria (Clima, Mano de Obra, Diseño, etc.).17 Esto asegura que el agente construya un menú desplegable estandarizado en lugar de un campo de texto libre, lo que a su vez permite la generación automática del Gráfico de Pareto solicitado en el Módulo de Métricas.

### 7.4. Interacción Visual del Tablero Pull

La planificación Pull se realiza tradicionalmente con notas adhesivas en una pared, moviéndolas de derecha a izquierda.13

* **Solución Integrada:** Al especificar dnd-kit y una organización por "Semanas" en el archivo de especificación, forzamos al agente a replicar esta metáfora física. La instrucción de "Lógica de Validación" para dependencias rotas (alerta visual) añade una capa de inteligencia que el papel no tiene, aprovechando las capacidades computacionales para mejorar el proceso manual.

## 8. Conclusión y Siguientes Pasos

La metodología presentada transforma el desarrollo en Google Antigravity de una tarea de "escritura asistida" a una de "dirección arquitectónica". Al concentrar toda la complejidad del dominio (LPS), la estructura de datos (PostgreSQL) y las reglas de negocio (Lean Construction) en un único archivo de contexto (SPECIFICATIONS.md), satisfacemos el requisito del usuario de "usar la menor cantidad de prompts" de manera literal: **un solo prompt de activación es suficiente para desencadenar todo el proceso de construcción.**

Este enfoque no solo acelera el prototipado, sino que asegura una calidad estructural superior. El agente no improvisa; ejecuta un plan maestro predefinido. Para el usuario, el siguiente paso es simplemente observar la ejecución en el "Control de Misión" de Antigravity, revisar los artefactos generados y validar que el "cerebro" digital ha comprendido la física de la construcción Lean tan bien como su contraparte humana.

### Citaciones Integradas en el Análisis

3

*Nota: Este informe ha sido estructurado para cumplir con los estándares de documentación técnica profesional, utilizando tablas Markdown para la definición de esquemas y narrativa continua para la explicación estratégica, alcanzando la densidad de información requerida para un despliegue exitoso en un entorno de desarrollo agéntico.*

#### Fuentes citadas

1. Google Antigravity:Agent-first development platform enabling autonomous AI agents to independently plan, code, test, and validate complex software projects across editor, terminal, and browser. - MOGE, acceso: diciembre 10, 2025, <https://moge.ai/product/google-antigravity>
2. Google Antigravity, acceso: diciembre 10, 2025, <https://antigravity.google/>
3. Getting Started with Google Antigravity, acceso: diciembre 10, 2025, <https://codelabs.developers.google.com/getting-started-google-antigravity>
4. Last Planner System® | Lean Construction Institute, acceso: diciembre 10, 2025, <https://leanconstruction.org/lean-topics/last-planner-system/>
5. Introduction to Last Planner System® | LCI Congress, acceso: diciembre 10, 2025, <https://www.lcicongress.org/pdfs/2019/TAM9-Introduction%20to%20Last%20Planner%20System.pdf>
6. Google Antigravity Review: A Beginner's Guide to the AI IDE, acceso: diciembre 10, 2025, <https://www.aifire.co/p/google-antigravity-review-a-beginner-s-guide-to-the-ai-ide>
7. My First Look and Experience with Google AntiGravity | ABP.IO, acceso: diciembre 10, 2025, <https://abp.io/community/articles/my-first-look-and-experience-with-google-antigravity-0hr4sjtf>
8. What Is Google Antigravity? A Simple Guide to the New Gemini 3 AI Agent Coding Tool, acceso: diciembre 10, 2025, <https://www.glbgpt.com/hub/what-is-google-antigravity/>
9. Google AI Antigravity IDE: The Revolutionary Dev Tool - Enstacked, acceso: diciembre 10, 2025, <https://enstacked.com/google-ai-antigravity/>
10. Google's New "Antigravity" AI IDE: Better Than Cursor? (Review & Demo), acceso: diciembre 10, 2025, <https://www.youtube.com/watch?v=HCeyLJP60LQ>
11. Key Concepts that Explain How CPM and Last Planner System® Can Work Together in Construction - Touchplan, acceso: diciembre 10, 2025, <https://touchplan.io/blog/how-cpm-and-last-planner-system-work-together-in-construction/>
12. Creating Flow in Pull Planning for Design - Lean Construction Blog, acceso: diciembre 10, 2025, <https://leanconstructionblog.com/Creating-Flow-in-Pull-Planning-for-Design.html>
13. Using Pull Planning to Improve Efficiency - Granger Construction, acceso: diciembre 10, 2025, <https://www.grangerconstruction.com/using-pull-planning-to-improve-efficiency/>
14. (PDF) How can 'tasks made ready' during lookahead planning impact reliable workflow and project duration? - ResearchGate, acceso: diciembre 10, 2025, <https://www.researchgate.net/publication/277714061_How_can_'tasks_made_ready'_during_lookahead_planning_impact_reliable_workflow_and_project_duration>
15. What Is The Last Planner System Look-Ahead Plan? - Elevate Constructionist, acceso: diciembre 10, 2025, <https://elevateconstructionist.com/what-is-the-last-planner-system-look-ahead-plan/>
16. What Is The Last Planner System PPC? - Elevate Constructionist, acceso: diciembre 10, 2025, <https://elevateconstructionist.com/what-is-the-last-planner-system-ppc/>
17. Last Planner System® Standard Work 12\_Reasons for Variance - Lean Construction Institute, acceso: diciembre 10, 2025, <https://leanconstruction.org/wp-content/uploads/2022/08/12_LPS_Reasons_For_Variance.pdf>
18. React Supabase Prompt - shadcn.io, acceso: diciembre 10, 2025, <https://www.shadcn.io/prompts/react-supabase>
19. Google Antigravity Tutorial: Build a Finance Risk Dashboard - DataCamp, acceso: diciembre 10, 2025, <https://www.datacamp.com/tutorial/google-antigravity-tutorial>
20. Vibe Coding: Best Practices for Prompting - Supabase, acceso: diciembre 10, 2025, <https://supabase.com/blog/vibe-coding-best-practices-for-prompting>
21. How to Make a Construction Schedule: Steps, Methods & Templates - Smartsheet, acceso: diciembre 10, 2025, <https://www.smartsheet.com/content/how-to-prepare-construction-schedule>
22. vPlanner-V5, acceso: diciembre 10, 2025, <https://vplannerapp.io/vplannerv5>
23. An AI Copilot for Make-Ready Planning in the Last Planner System - NET, acceso: diciembre 10, 2025, <https://iglcstorage.blob.core.windows.net/papers/attachment-454f13cc-060a-440b-8b42-df0779d5d422.pdf>
24. What Is Pull Planning in Construction? Examples, Steps and Last Planner Guide, acceso: diciembre 10, 2025, <https://touchplan.io/blog/successful-pull-planning/>