Primero la instalación del programa desde:
https://antigravity.google/download

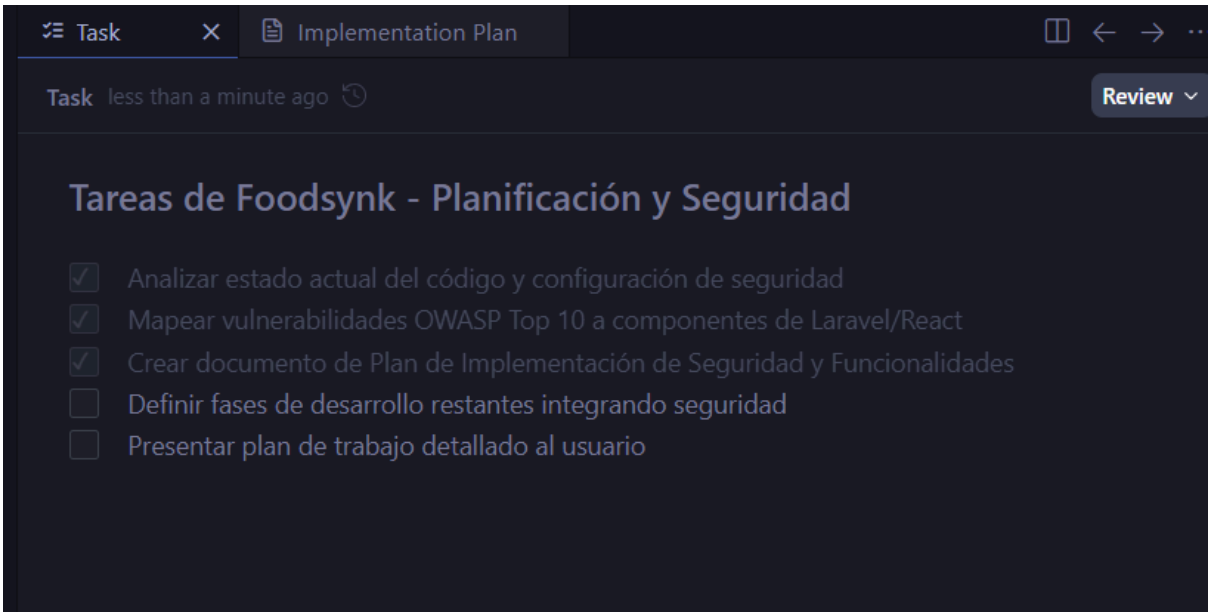
Luego abrimos la carpeta de mi proyecto en mi caso como está creado con Laragon sería:
C:\laragon\www\Foodsynk

Para el pront, bastante largo le pasé las primeras páginas de mi proyecto con nombre, definición, el stack tecnológico, objetivos… También le pasé la información de mi diario de proyecto, para que supiera como estoy enfocando el trabajo.

Por último le dije:
Con todo esto, puedes ver como es el proyecto y como lo estoy enfocando más o menos...
Hazme un plan de trabajo con mejoras de seguridad según OWASP TOP TEN

si tienes cualquier duda del proyecto o de como lo he planteado no dudes en preguntarme
Contestame en español

Esto a generado:



Qué determinó, el siguiente documento:

Plan de Trabajo y Mejoras de Seguridad (OWASP Top 10) - Foodsynk

Este documento detalla el plan de desarrollo para las siguientes fases del proyecto Foodsynk, integrando controles de seguridad desde el diseño para prevenir vulnerabilidades comunes (OWASP Top 10).

Estado Actual

- **Frameworks:** Laravel 10+ (API), React (Frontend).
- **Autenticación:** Tokens Bearer (Sanctum).
- **Funcionalidad:** Login/Registro/Logout, CRUD de Recetas (Orion).
- **Seguridad Base:** Validación de entrada básica, Policies simples para Recetas.

Objetivos de Seguridad (OWASP Integration)

Implementaremos medidas específicas para mitigar los riesgos del top 10 de OWASP durante el desarrollo:

- **A01: Broken Access Control:** Asegurar que los usuarios solo accedan a sus datos (Menús, Listas). Implementar roles (Admin vs User).
- **A03: Injection:** Mantener uso estricto de Eloquent ORM y validación de tipos de datos.
- **A07: Identification and Authentication Failures:** Reforzar contraseñas y rate-limiting.
- **A04: Insecure Design:** Validar lógica de negocio (ej. no comprar ingredientes negativos).

Fases de Desarrollo Propuestas

Fase 1: Hardening de Seguridad Actual (Inmediato)

Antes de avanzar, aseguraremos la base existente.

[MODIFY] Seguridad en Autenticación

- **Rate Limiting:** Configurar `Throttle` en ruta de login para evitar fuerza bruta (A07).

- **Validación de Contraseñas:** Asegurar reglas de complejidad (`Password::defaults()`) en registro (A07).
- **Mensajes de Error Genéricos:** Confirmar que el login no revela si el email existe o no (A07).

[MODIFY] Seguridad en Recetas

- **Validación Estricta:** Revisar reglas de validación en `RecipeController` (evitar XSS en campos de texto si se renderizan como HTML después, aunque React escapa por defecto) (A03).
- **Policy Check:** Verificar que `view` diferencie entre recetas públicas y privadas (si aplica en el futuro) (A01).

Fase 2: Gestión de Menús (Semana 8-9)

Implementación del núcleo de planificación.

[NEW] Backend Menús

- **Modelo y Migración:** `Menu`, `MenuRecipe` (Tabla pivote).
- **Controlador API:** `MenuController` (usando Orion o standard).
- **OWASP Focus (A01):** Crear `MenuPolicy` estricto. **CRÍTICO:** Un usuario NO debe poder ver ni editar el menú de otro ID secuencialmente (IDOR).
  - `view`: `user_id === $user->id`
  - `update`: `user_id === $user->id`
- **Validación:** Asegurar que las recetas añadidas al menú existen.

Fase 3: Lista de la Compra (Semana 10-11)

Lógica de negocio compleja y manejo de datos de usuario.

[NEW] Generación de Lista

- **Lógica:** Servicio `ShoppingListGenerator` que itera menús -> recetas -> ingredientes.
- **OWASP Focus (A04 - Insecure Design):**
  - Evitar bucles infinitos en cálculos.
  - Validar límites de arrays (ej. no procesar un menú de 1000 días para evitar DoS).
- **Endpoints:** `GET /shopping-list`, `POST /shopping-list/item` (añadir manual).

Fase 4: Roles y Administración (Semana 12+)

Introducción del rol de Administrador.

[MODIFY] Sistema de Roles

- **Base de Datos:** Añadir columna `role` o tabla `roles` a usuarios.
- **Middleware:** Crear middleware `CheckRole` o `EnsureUserIsAdmin`.
- **OWASP Focus (A01):**
    - Proteger rutas de borrado global con middleware de admin.
    - Asegurar que un usuario normal no pueda elevar privilegios (Mass Assignment en user update).

Plan de Verificación

Pruebas Automatizadas (PHPUnit)

1. **Auth Test:** Intentar login fallido 5 veces y verificar bloqueo (Throttle).
2. **Policy Test:**
    - Usuario A intenta borrar Receta de Usuario B -> `403 Forbidden`.
    - Usuario A intenta ver Menú de Usuario B -> `403 Forbidden`.
3. **Input Test:** Enviar strings muy largos o caracteres extraños a endpoints de creación.

Verificación Manual

1. **Postman:** Usar token de Usuario A para atacar endpoints de recursos ID de Usuario B.
2. **SQL Injection:** Intentar inyectar `' OR 1=1 --` en campos de búsqueda (Orion debería manejarlo, verificar).