Universidad Autónoma de Baja California

Facultad de Ciencias Químicas e Ingeniería

PROYECTO DE CARRERA

Actividad Meta 3.4
Descripción de la Solución

Docente: J. Reyes Juarez Ramirez: 17500

Alumnos:

Emmanuel Alberto Gómez Cárdenas : 01261509

Pablo Constantino Leon Romero : 01253171

Instrucciones de la actividad

1. Redactar la sección “Descripción de la solución” del reporte del proyecto. **(Ver indicaciones en la Plantilla del reporte. Además considerar los elementos expuestos en el Foro de esta actividad)**
2. Subir la redacción de la sección a Blackboard

Índice

Desarrollo de los pasos

Elementos que Conforman la Solución

ScholarSync se estructura en torno a varios componentes clave, que incluyen modelos lógicos, algoritmos y la implementación tecnológica necesaria para crear una solución efectiva y eficiente. A continuación, se describen estos elementos en detalle.

Modelos Lógicos

- **Modelo de Notificación Intrusiva:**
  - **Frecuencia de Notificaciones:** Basado en un algoritmo que ajusta la frecuencia de las notificaciones en función de la proximidad de las fechas límite y la cantidad de tareas pendientes.
  - **Detección de Aplicaciones de Ocio:** Utiliza un sistema de monitoreo que identifica cuando el usuario abre aplicaciones designadas como "Aplicaciones no permitidas" (juegos, redes sociales, etc.) y genera notificaciones que recuerdan al usuario sus tareas pendientes.

- **Integración con Google Calendar:**
  - **Sincronización de Tareas:** Implementa una API que permite sincronizar las tareas y eventos del Google Calendar del usuario con ScholarSync. Esto asegura que las notificaciones estén actualizadas y sean relevantes.
  - **Gestión de Permisos y Seguridad**: Asegura que la aplicación gestione los datos del calendario de manera segura y con el consentimiento del usuario.

Algoritmos

- **Algoritmo de Priorización de Tareas:**
  - **Evaluación de Urgencia:** Clasifica las tareas según su urgencia y el tiempo restante hasta la fecha límite.
  - **Priorización Dinámica:** Ajusta la prioridad de las tareas en tiempo real basándose en la actividad del usuario y el calendario.
- **Algoritmo de Detección de Aplicaciones:**
  - **Monitoreo en Tiempo Real:** Detecta cuando se abren aplicaciones de ocio y envía notificaciones intrusivas que interrumpen la actividad del usuario para recordarle sus tareas pendientes.
  - **Lista de Aplicaciones Configurables:** Permite a los usuarios definir qué aplicaciones consideran distractoras, ajustando así las notificaciones a sus necesidades personales.

Implementación Tecnológica

- **Desarrollo de la Aplicación:**
  - **Lenguaje y Entorno:** ScholarSync se desarrollará utilizando C# y .NET en el entorno de desarrollo Visual Studio, aprovechando sus capacidades para crear aplicaciones de escritorio robustas y eficientes.
  - **Interfaz de Usuario (UI):** Se diseñará una interfaz intuitiva y amigable que facilite la interacción del usuario con la aplicación. La UI mostrará las tareas pendientes, las notificaciones recientes y permitirá la configuración de las preferencias del usuario.
- **Integración de APIs:**
  - **Google Calendar API:** Se integrará para sincronizar las tareas del calendario con ScholarSync, garantizando que las notificaciones sean precisas y actualizadas.

Arquitectura de Software

- **Arquitectura Monolítica:**
  - Nuestra aplicación ScholarSync utiliza una arquitectura monolítica, donde todas las funcionalidades están integradas en un único sistema cohesivo.

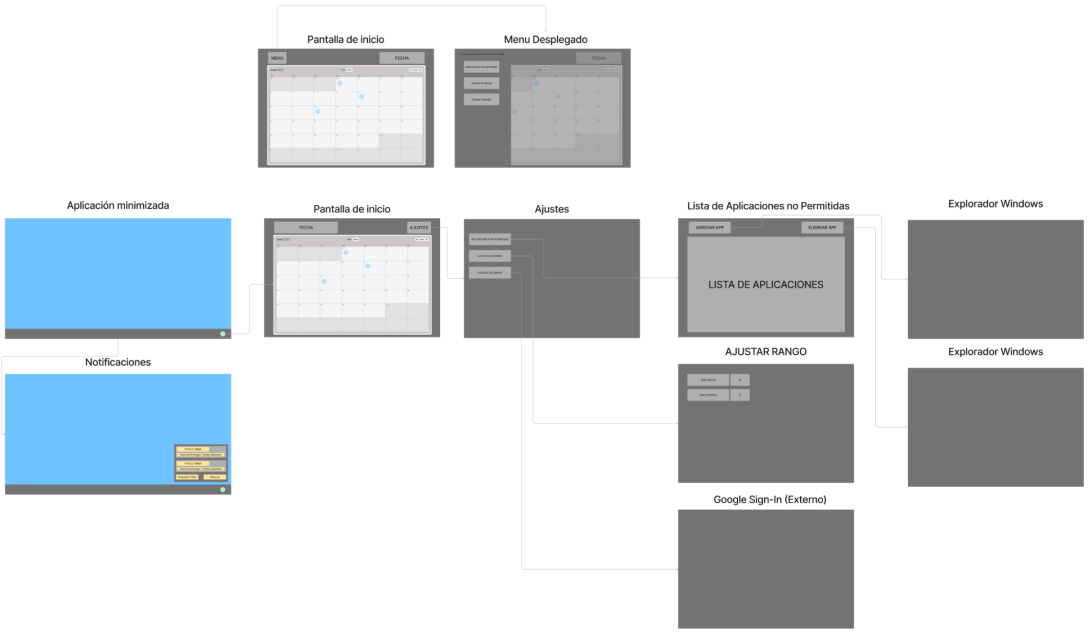
    Esto significa que la interacción con el usuario, la sincronización con Google Calendar y el almacenamiento de las configuraciones del usuario se manejan dentro de la misma aplicación. No requiere componentes separados para el cliente y el servidor, lo que simplifica la implementación y el mantenimiento, proporcionando una solución eficiente y fácil de usar.

- **Módulos de la Aplicación:**
  - **Módulo de Sincronización:** Responsable de conectarse con Google Calendar, obtener datos y mantenerlos actualizados.
  - **Módulo de Notificaciones:** Gestiona la generación y visualización de notificaciones intrusivas.
  - **Módulo de Configuración**: Permite a los usuarios personalizar sus preferencias, como el rango de fechas desde el cual serán procesadas las tareas pendientes y generadas las notificaciones.

Diagramas y Ejemplos

- **Diagrama de Estructura (Wireframe):**
  - Representación visual básica de la aplicación. Es un diseño de interfaces de usuario para mostrar la estructura y disposición de los elementos principales en una pantalla sin incluir detalles de diseño gráfico, colores o tipografías específicas

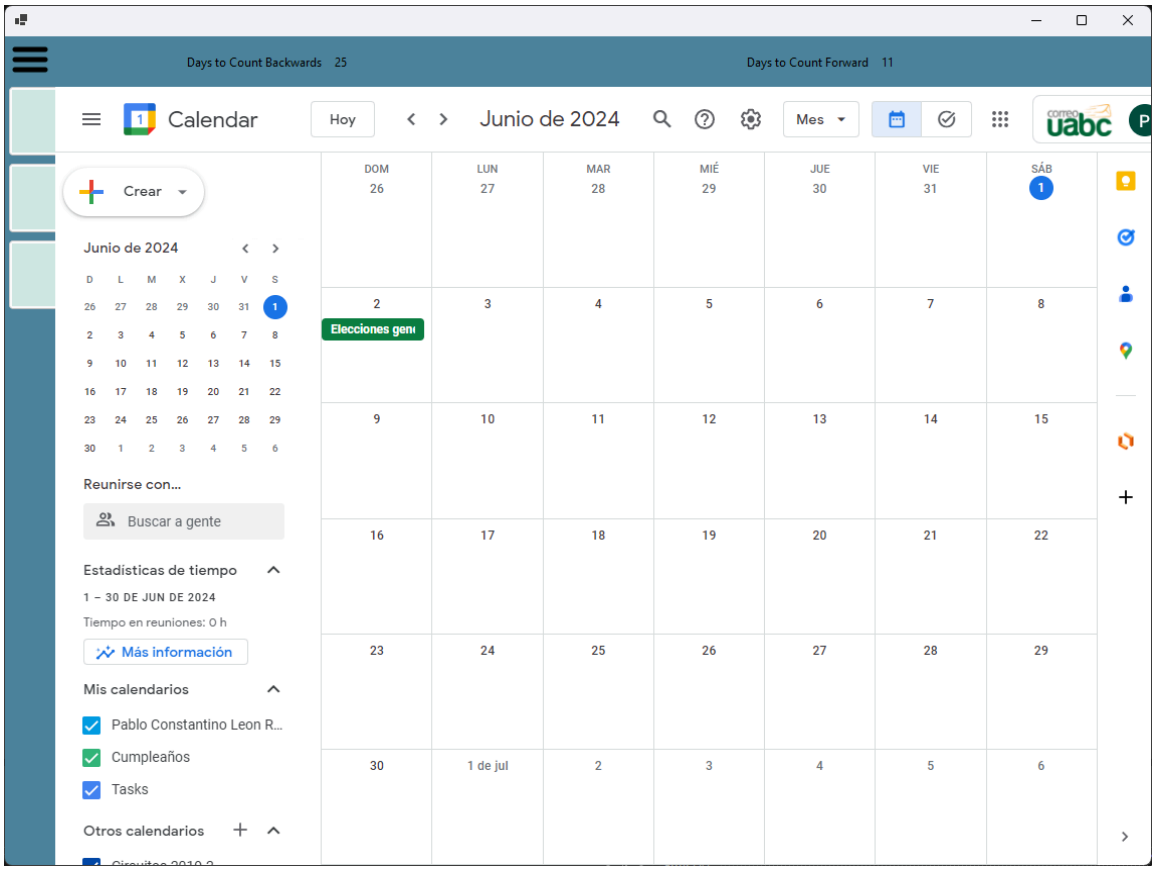
Emmanuel Alberto Gómez Cárdenas
Pablo Constantino Leon Romero



Emmanuel Alberto Gómez Cárdenas
Pablo Constantino Leon Romero

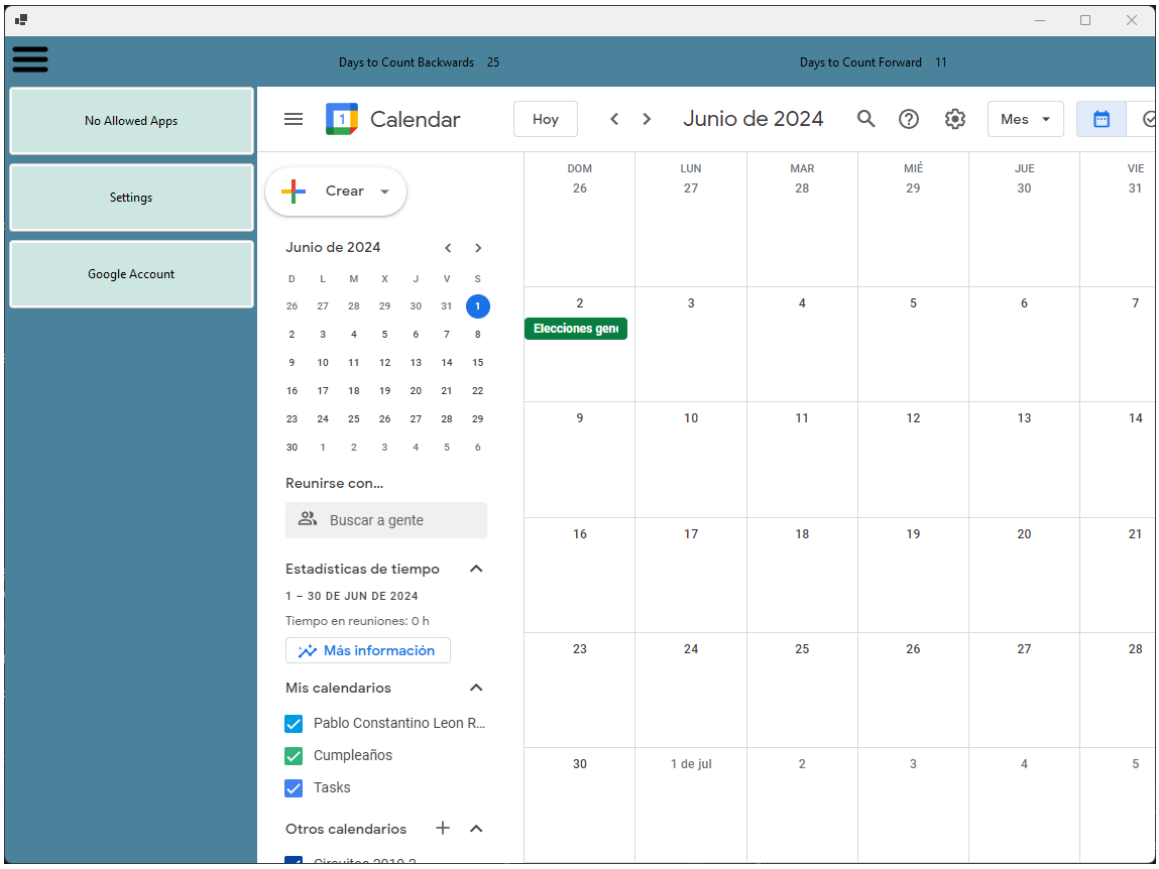
Producto Final

Form principal de la imagen

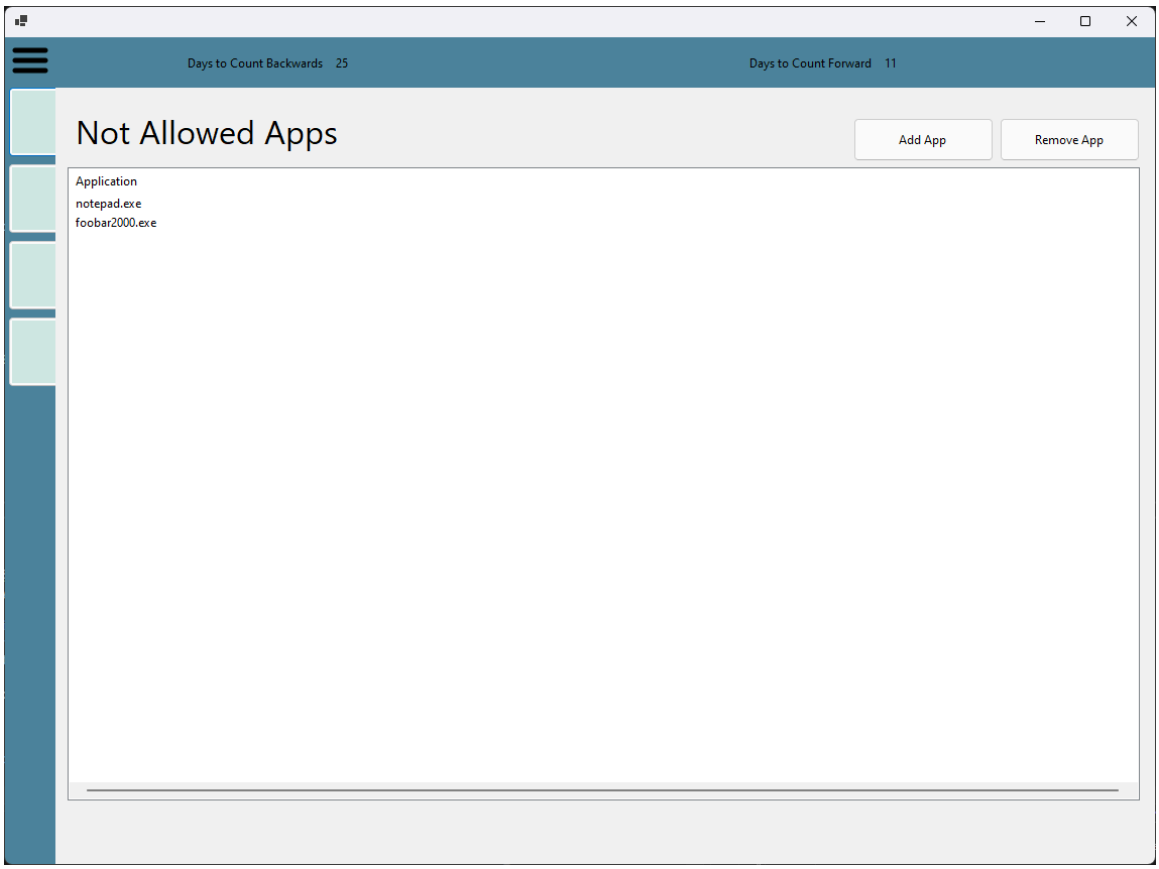


Emmanuel Alberto Gómez Cárdenas
Pablo Constantino Leon Romero

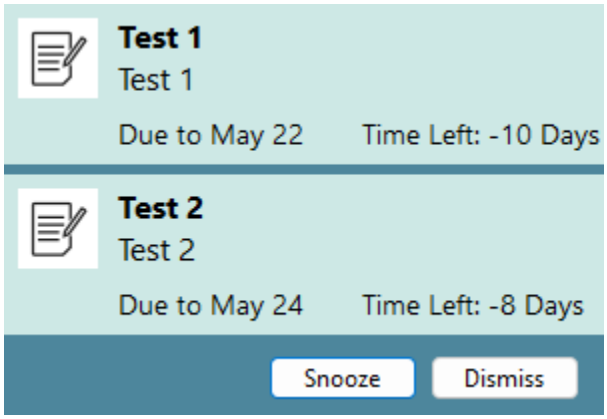
Form principal, sidebar extendido



Apartado para elegir aplicaciones a bloquear (que generan la alerta de la notificación)



Notificación



Referencias

BillWagner. *C# guide - .net managed language*. C# Guide - .NET managed language | Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/csharp/

Ghogen. *Visual studio documentation*. Microsoft Learn. https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022

Google. *Get started | API client library for .net | google for developers*. Google. https://developers.google.com/api-client-library/dotnet/get_started

Google. *Google Calendar API overview*. Google calendar api overview | google for developers. https://developers.google.com/calendar/api/guides/overview