Necesito que desarrolles un proyecto completo en el que se obtenga una aplicación que resuelva un problema real en este caso una aplicación en el software PSeInt para un estacionamiento privado. La aplicación debe incluir los siguientes elementos:

**Requisitos Funcionales**Definir una lista de requisitos funcionales que describan las características y funcionalidades  
básicas de tu aplicación. Esto puede incluir la manipulación de datos, la interacción del  
usuario y la lógica de programación.

**Diseño de Interfaz de Usuario**Crea un diseño de la interfaz de usuario de la aplicación utilizando diagramas de flujo y  
pseudocódigo en Pseint para representar la lógica y el flujo de la aplicación.

**Desarrollo de Código**Implementa la aplicación utilizando Pseint. Utiliza buenas prácticas de programación.  
GitHub  
Crea un repositorio en Github para tu proyecto. Realiza un seguimiento de las versiones del  
código a medida que avanzas en el desarrollo (si corresponde) y documenta los cambios  
utilizando commits significativos y mensajes claros.  
Metodologías Ágiles  
Aplica conceptos de metodologías ágiles para gestionar tu proyecto individualmente. Dividí el  
trabajo en tareas más pequeñas, establece plazos y realiza seguimiento de tu progreso de  
forma fraccionada, como si luego de hacer un prototipo en Pseint deberías planificar el sistema  
real.  
Presentación Final  
Creá un documento que presente tu aplicación y el proceso que seguiste para desarrollarla.  
Explica cómo aplicar los conceptos de pensamiento computacional, pseudocódigo, Git/Github  
y metodologías ágiles en tu proyecto. Colocá todos los links que consideres necesarios para  
que el tutor pueda acceder al repositorio y los documentos asociados. Este documento es el  
que deberás subir al campus en el espacio de trabajo final.

Default (GPT-3.5)



Necesito que desarrolles un proyecto completo en el que se obtenga una aplicación que resuelva un problema real en este caso una aplicación en el software PSeInt para un estacionamiento privado. La aplicación debe incluir los siguientes elementos: Requisitos Funcionales Definir una lista de requisitos funcionales que describan las características y funcionalidades básicas de tu aplicación. Esto puede incluir la manipulación de datos, la interacción del usuario y la lógica de programación. Diseño de Interfaz de Usuario Crea un diseño de la interfaz de usuario de la aplicación utilizando diagramas de flujo y pseudocódigo en Pseint para representar la lógica y el flujo de la aplicación. Desarrollo de Código Implementa la aplicación utilizando Pseint. Utiliza buenas prácticas de programación. GitHub Crea un repositorio en Github para tu proyecto. Realiza un seguimiento de las versiones del código a medida que avanzas en el desarrollo (si corresponde) y documenta los cambios utilizando commits significativos y mensajes claros. Metodologías Ágiles Aplica conceptos de metodologías ágiles para gestionar tu proyecto individualmente. Dividí el trabajo en tareas más pequeñas, establece plazos y realiza seguimiento de tu progreso de forma fraccionada, como si luego de hacer un prototipo en Pseint deberías planificar el sistema real. Presentación Final Creá un documento que presente tu aplicación y el proceso que seguiste para desarrollarla. Explica cómo aplicar los conceptos de pensamiento computacional, pseudocódigo, Git/Github y metodologías ágiles en tu proyecto. Colocá todos los links que consideres necesarios para que el tutor pueda acceder al repositorio y los documentos asociados. Este documento es el que deberás subir al campus en el espacio de trabajo final.

ChatGPT

**Proyecto: Aplicación de Gestión de Estacionamiento Privado en PSeInt**

En este proyecto, desarrollaremos una aplicación de software en PSeInt para gestionar un estacionamiento privado. La aplicación permitirá a los usuarios realizar operaciones como registrar entradas y salidas de vehículos, calcular tarifas de estacionamiento y generar informes. Aplicaremos conceptos de pensamiento computacional, pseudocódigo, Git/GitHub y metodologías ágiles para llevar a cabo el desarrollo de manera efectiva.

**Requisitos Funcionales:**

1. **Registro de Entrada y Salida:** Los usuarios deben poder registrar la entrada y salida de vehículos, capturando la información de la matrícula y la hora.
2. **Cálculo de Tarifas:** La aplicación debe calcular las tarifas de estacionamiento basadas en el tiempo de permanencia del vehículo en el estacionamiento.
3. **Generación de Informes:** Los usuarios deben poder generar informes que muestren la información de todas las entradas y salidas, incluyendo matrícula, hora de entrada, hora de salida y tarifa correspondiente.

**Diseño de Interfaz de Usuario:**

1. **Registro de Entrada y Salida:**

Inicio

Leer matrícula

Registrar hora de entrada

Registrar hora de salida

Fin

1. **Cálculo de Tarifas:**

csharpCopy code

Inicio Calcular tiempo de permanencia Si tiempo <= 1 hora Tarifa = Tarifa base Sino Tarifa = Tarifa base + (Tarifa por hora extra \* (tiempo - 1)) Fin

1. **Generación de Informes:**

pythonCopy code

Inicio Mostrar "Informe de Estacionamiento" Para cada vehículo en la lista de vehículos Mostrar información del vehículo Fin

**Desarrollo de Código:**

pseudocodeCopy code

Algoritmo EstacionamientoPrivado Definir vehiculos como Matriz de Caracteres Definir horas\_entrada como Matriz de Enteros Definir horas\_salida como Matriz de Enteros Definir tarifa\_base como Real Definir tarifa\_hora\_extra como Real Definir opcion como Entero tarifa\_base = 5.0 tarifa\_hora\_extra = 3.0 Mientras verdadero Mostrar "Menú:" Mostrar "1. Registrar entrada" Mostrar "2. Registrar salida" Mostrar "3. Generar informe" Mostrar "4. Salir" Leer opcion Segun opcion Hacer 1: RegistrarEntrada(vehiculos, horas\_entrada) 2: RegistrarSalida(vehiculos, horas\_salida) 3: GenerarInforme(vehiculos, horas\_entrada, horas\_salida, tarifa\_base, tarifa\_hora\_extra) 4: Mostrar "Saliendo del programa." Fin De Otro Modo: Mostrar "Opción no válida." Fin Segun Fin Mientras Fin Algoritmo

**GitHub:**

Crear un repositorio en GitHub para el proyecto y realizar seguimiento de las versiones a medida que se desarrolla. Realizar commits significativos y documentar los cambios de manera clara.

**Metodologías Ágiles:**

Aplicar metodologías ágiles para gestionar el proyecto de manera individual. Dividir el trabajo en tareas más pequeñas, establecer plazos y hacer un seguimiento del progreso de forma fraccionada.

1. **Sprint 1:**
   * Diseñar la interfaz de usuario en PSeInt.
   * Implementar el registro de entrada de vehículos.
2. **Sprint 2:**
   * Implementar el registro de salida de vehículos.
   * Calcular tarifas de estacionamiento.
3. **Sprint 3:**
   * Generar informes de entradas y salidas.
   * Realizar pruebas y ajustes finales.

**Presentación Final:**

Crear un documento que explique el proceso de desarrollo, desde el diseño de la interfaz hasta la implementación y las pruebas. Destacar cómo se aplicaron los conceptos de pensamiento computacional, pseudocódigo, Git/GitHub y metodologías ágiles. Proporcionar enlaces al repositorio de GitHub y cualquier otro recurso necesario.

Este proyecto se ha diseñado para abordar un problema real al proporcionar una solución funcional a la gestión de un estacionamiento privado utilizando el software PSeInt.

El código que proporcionaste es una mejora del algoritmo para gestionar el estacionamiento. Esta sección del código se encarga de registrar la salida de un vehículo, calcular la tarifa y eliminar la entrada del vehículo del arreglo.

Aquí hay una explicación detallada de lo que hace esta parte del código:

1. El bucle **Para i <- 1 Hasta autosEstacionados Hacer** itera a través de todos los autos estacionados para buscar la matrícula del vehículo que sale.
2. Dentro del bucle, verifica si la matrícula del vehículo en la posición **i** coincide con la matrícula del vehículo que sale (**matriculaSalida**).
3. Si encuentra una coincidencia (**Si matriculas[i] = matriculaSalida Entonces**), marca **Encontrado** como Verdadero y continúa con el proceso de salida.
4. Se le pide al usuario que ingrese la hora de salida del vehículo en formato HHMM (**Escribir "Ingrese la hora de salida (en formato HHMM): "**).
5. Calcula el tiempo que el vehículo estuvo estacionado restando la hora de salida (**horaSalida**) y la hora de ingreso (**horasIngreso[i]**). Luego, divide el resultado por 100 para obtener el tiempo en horas (**tiempoEstacionado <- (horaSalida - horasIngreso[i]) / 100**).
6. Calcula la tarifa multiplicando el tiempo estacionado por el precio por hora (**costo <- tiempoEstacionado \* precioPorHora**).
7. Dentro de un bucle **Para j <- i Hasta autosEstacionados - 1 Hacer**, reorganiza el arreglo **matriculas** y **horasIngreso** para eliminar la entrada del vehículo que está saliendo.
8. Reduce el contador **autosEstacionados** en uno (**autosEstacionados <- autosEstacionados - 1**).
9. Muestra un mensaje de salida exitosa (**Escribir "Salida registrada exitosamente."**) y un espacio en blanco para mejorar la presentación (**Escribir " "**).

Este código permite registrar la salida de un vehículo y calcular la tarifa automáticamente, eliminando la entrada del vehículo del arreglo. Asegúrate de haber definido **Encontrado** y **precioPorHora** antes de este fragmento de código según tus necesidades.