Proyecto Final: Módulo Predictor de Tiempo de Espera de Buses *"CuandoLlega"*

Fecha	27 de Julio de 2022
Paralelo	2
Asignatura	ELO329 - Diseño y Programación Orientados a Objetos
Integrantes	Benjamín Lillo
	Catalina Zelaya
	Gustavo Matamala
	Diego Alfaro
Profesor	Werner Creixell

ELO329 - Diseño y Programación Orientados a Objetos

Descripción del problema

En el momento de usar transporte público, uno se encuentra con la dificultad de saber, además de que micros pasan en cierto paradero, también a que horas pasan en estas. Como consecuencia, viajes cuidadosamente planeados pueden llegar a ser frustrados por largas esperas e incertidumbre en cuanto a las líneas que puede tomar en cierto paradero.

Análisis del problema

Una solución para esto es escribir una aplicación que procese información de tal manera que pueda indicar cuanto falta para que una micro específica llegue a cierto paradero indicado por el usuario.

La solución actual es mucho más compleja de lo que se ve en este informe, por lo que solo nos enfocaremos en lo que esté a nuestro alcance. La manera en que se puede proceder es dividiendo el proceso en tres partes: Envío de información GPS desde cada bus → usar un módulo que, a partir de la información de cada recorrido, entregue un valor de porcentaje con respecto al recorrido total de la micro → usar otro módulo que recibe este porcentaje y posee un método que lo asocia con el porcentaje de recorrido en que se encuentra un paradero determinado → mostrar este dato en aplicación. Nosotros nos enfocaremos en la tercera parte.

Definición de requerimientos

Los requerimientos para la resolución del problema que sera abordado en el proyecto, son los siguientes:

el primer requerimiento, seria observar detenidamente la problemática: Largas esperas e incertidumbre al momento de usar la locomoción colectiva en Valparaíso (u otras ciudades).

Para resolver esto, la solución propuesta es diseñar un modelo que permita indicar al usuario cuánto falta para que pase su micro en el paradero en que se encuentra, mediante alguna aplicación.

los requerimientos entonces para la construcción del problema serian los que se pueden observar en los siguientes diagramas:

Diagrama 1:

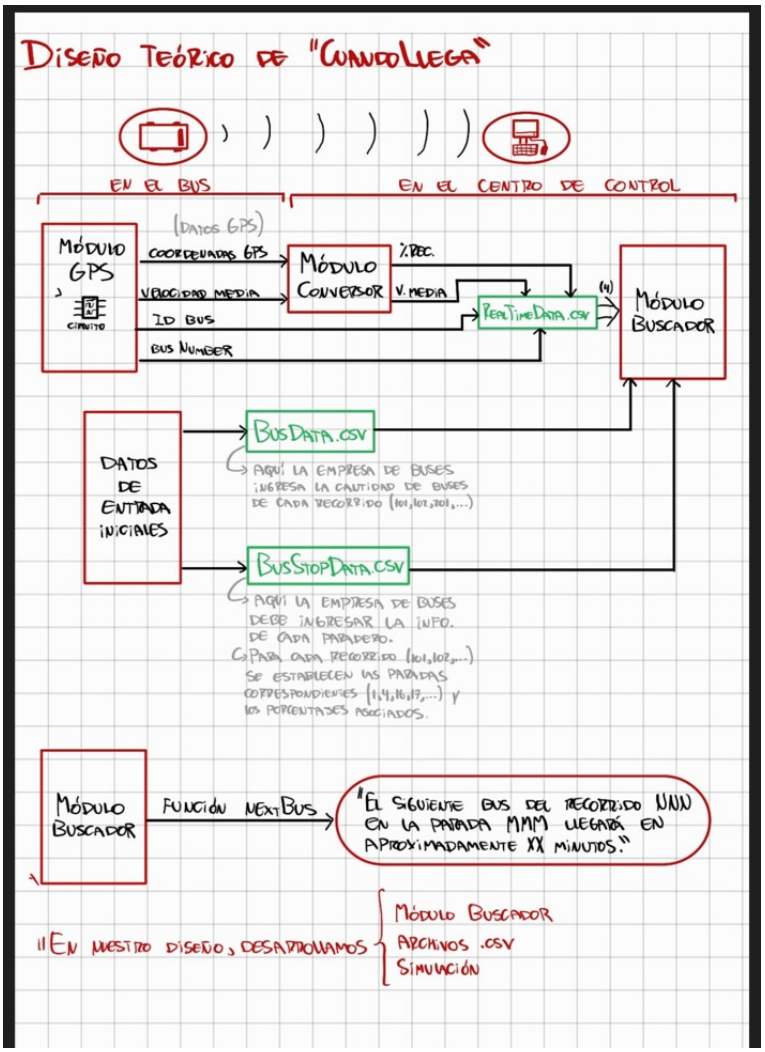

Figura 1: Diagrama de implementación real.

Como podemos observar, en el diagrama se explica el diseño teórico de nuestro programa: ”Cuando llega”, o básicamente los requerimientos de este mismo para su correcto funcionamiento La 1era parte(modulo GPS) consiste en el envió de información GPS de cada bus, en este caso cada bus tendrá un GPS integrado para poder observar su localización

La 2da parte consiste en usar un modulo que, a partir de la información de cada recorrido, sea capaz de entregar un valor de porcentaje con respecto al recorrido total de la micro, además de dedicarse a observar sus coordenadas y su distancia recorrida

La 3era parte consiste en usar otro modulo que recibe el porcentaje del recorrido anterior, y posee un método

que asocia este porcentaje con la distancia al siguiente paradero, para lo cual calcular el tiempo que falte para que el bus llegue al siguiente paradero.

Diagrama 2:

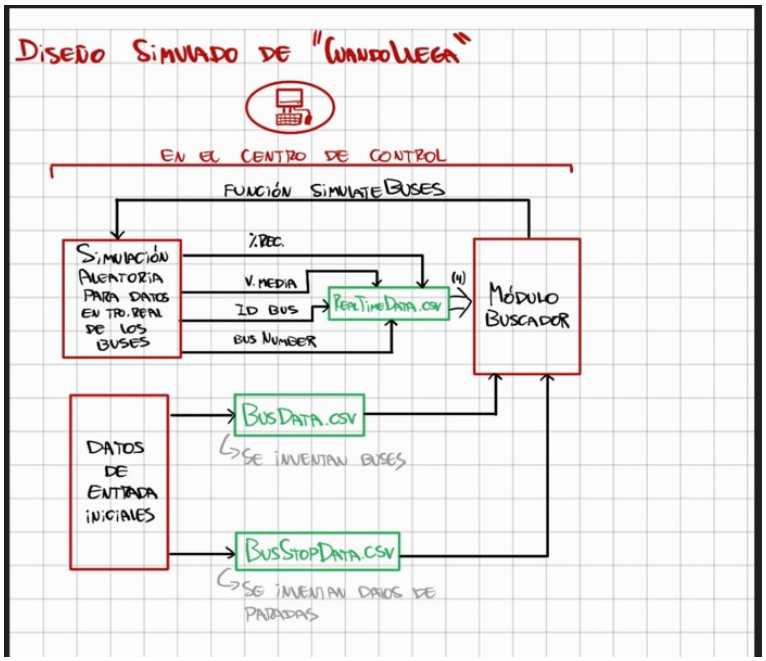

Figura 2: Diagrama de implementación simulada.

Como podemos observar, se ve un diseño simulado del programa: ”Cuando llega”, el cual esta orientado en el centro de control de información, usando las funciones que se implementaron, en este caso, simulatebuses, para simular el trayecto de un bus y su recorrido y extraer datos, usando el modulo buscador

Casos de uso

Actor	Sistema
1. El usuario abre el software del sistema de control de buses.	2. El sistema despliega el menú y espera a que una opción sea seleccionada.
3. El usuario selecciona la opción 1 (Usar datos en tiempo realprevistos por RealTime.csv).	4. El sistema despliega dos opciones, una para realizar laacción y otra para volver al menú principal, y espera a que la opción sea seleccionada.
5. El usuario selecciona la opción 1 (Consultar próximo bus en paradero).	6. El sistema despliega una pregunta para saber el bus a localizar.
7. El usuario ingresa el número del bus a localizar. (Ej. 101)	8. El sistema despliega una pregunta para saber el paradero que consulta el usuario.
9. El usuario ingresa el paradero que desea consultar. (Ej. 3)	10. El sistema despliega una pregunta para saber la dirección del bus.
11. El usuario ingresa la dirección del bus. (Ej. 1 (Norte))	11. El sistema despliega un mensaje para informar al usuario que se está consultando un bus con las características requeridas.
	12. El sistema despliega el tiempo que demorará el bus requerido al llegar al paradero indicado.

Actor	Sistema
1. El usuario abre el software del sistema de control de buses.	2. El sistema despliega el menú y espera a que una opción sea seleccionada.
3. El usuario selecciona la opción 2 (Simular datos aleatorios en tiempo real para los buses).	4. El sistema ejecuta la función simulateBuses(), para generar valores aleatorios para la base de datos de RealTimeData.csv, así utilizarlos en el programa.
	5. El sistema despliega dos opciones, una para realizar la acción y otra para volver al menú principal, y espera a que la opción sea seleccionada.
6. El usuario selecciona la opción 1 (Consultar próximo bus en paradero).	7. El sistema despliega una pregunta para saber el bus a localizar.
8. El usuario ingresa el número del bus a localizar. (Ej. 101)	9. El sistema despliega una pregunta para saber el paradero que consulta el usuario.
10. El usuario ingresa el paradero que desea consultar. (Ej. 16)	11. El sistema despliega una pregunta para saber la dirección del bus.
12. El usuario ingresa la dirección del bus. (Ej. 1 (Norte))	13. El sistema despliega un mensaje para informar al usuario que se está consultando un bus con las características requeridas.
	14. El sistema despliega el tiempo que demorará el bus requerido al llegar al paradero indicado.

Actor	Sistema
1. El usuario abre el software del sistema de control de buses.	2. El sistema despliega el menú y espera a que una opción sea seleccionada.
3. El usuario selecciona la opción 3 (Mostrar información cargada en el sistema).	4. El sistema despliega por pantalla la información ordenadade las bases de datos entregadas (Ej: cantidad de paraderos y cantidad de buses).

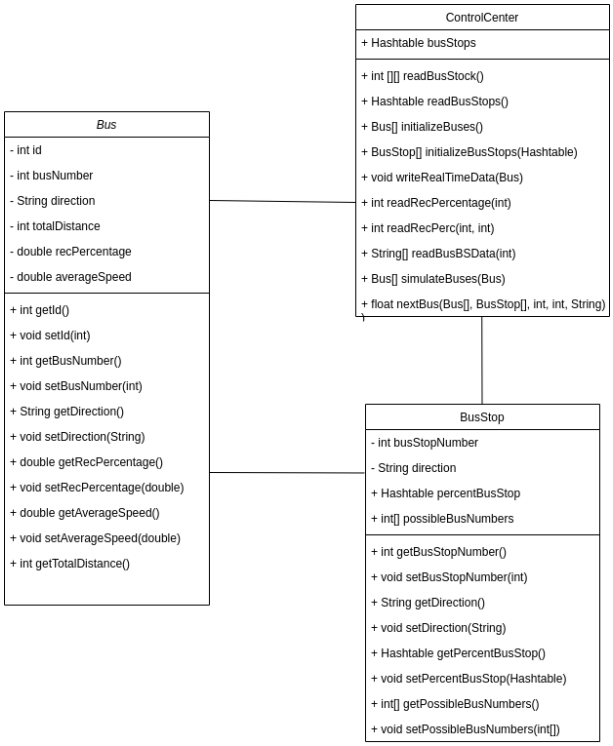

Figura 3: Diagrama UML

Diagramas UML y esquemas

Prueba y funcionamiento

Para esta sección, se explica a continuación el funcionamiento de el código de nuestro proyecto, exponiendo los resultados de las pruebas de ambos casos explicados. Se efectuará la simulación de buses de forma aleatoria.

Al iniciar el programa, se despliega un menú por consola, tal como se muestra a continuación:

```
============================================================
¡Bienvenido al sistema de control de buses!:
1. Usar datos en tiempo real provistos por RealTimeData.csv
2. Simular datos aleatorios en tiempo real para los buses
3. Mostrar información cargada en el sistema
4. Salir
============================================================
Ingrese su opción:
```

Figura 4: Menú principal.

Luego, se elige la opción 2, de tal forma de generar un archivo de datos en tiempo real simulado y totalmente aleatorio. Luego se muestra el siguiente menú:

```
Ingrese su opción:
2
============================================================
1. Consultar próximo bus en paradero
2. Volver al menú principal
============================================================
Ingrese su opción:
```

Figura 5: Siguiente menú.

Tras esto, conviene revisar los archivos proporcionados con datos fijos de buses y paraderos. Se asignan 100 buses para nuestra línea de prueba (101), tal como se muestra en el archivo CSV:

busNumber	busQuantity	totalDistance
101	100	20
102	5	15
103	5	15
104	5	15
105	5	15
106	5	15
201	5	15
202	5	15
203	5	15
204	5	15
205	5	15
206	5	15
301	5	25
302	5	15
303	5	15
304	5	15

Figura 6: Contenido de BusData.csv.

El archivo de paraderos contiene lo siguiente:

BusNumber	BusStops	BusStopPercs
101	1,2,3,5,6	200,400,600,800,900
102	1,2,3,4,5,6,8,1	200,400,600,800,850,880,900,950
103	1,2,3,5,6	200,400,600,800,900
104	1,2,3,5,6	200,400,600,800,900
105	1,2,3,5,6	200,400,600,800,900
106	1,2,3,5,6	200,400,600,800,900
201	1,2,3,5,6	200,400,600,800,900
202	1,2,3,5,6	200,400,600,800,900
203	1,2,3,5,6	200,400,600,800,900
204	1,2,3,5,6	200,400,600,800,900
205	1,2,3,5,6	200,400,600,800,900
206	1,2,3,5,6	200,400,600,800,900
301	1,2,3,5,6	200,400,600,800,900
302	1,2,3,5,6	200,400,600,800,900
303	1,2,3,5,6	200,400,600,800,900
304	1,2,3,5,6	200,400,600,800,900

Figura 7: Contenido de BusStopsData.csv.

La prueba que se realiza es para el bus de la línea 101, en el paradero de ID 5 y dirección Sur. Revisando la tabla de datos, se observa que el paradero 5 se encuentra en un 80.0 por ciento del recorrido total, y por consiguiente se espera que el bus más cercano esté a menos de dicho porcentaje del recorrido.

```
Ingrese su opción:
1
Ingrese el número de bus:
101
Ingrese el número de paradero:
5
Ingrese la dirección (1: Norte, 2: Sur):
2
Consultando próximo bus de la línea 101 en el paradero 5 en la dirección South...
El próximo bus llegará en 4 minutos más.
============================================================
1. Consultar próximo bus en paradero
2. Volver al menú principal
============================================================
```

Figura 8: Ejecución del programa para los valores especificados.

Se observa que el bus próximo llegará en 4 minutos. Dicho cálculo es realizado a partir de la distancia total de la línea, el porcentaje en tiempo real del bus más próximo y la velocidad promedio de este. Aquí se observan los datos en tiempo real del bus más próximo encontrado anteriormente:

	BusID	BusNumbe	BusDirectior	RecPercentage	AverageSpeed
16	24	101	North	814	35
17	61	101	South	805	39
18	26	101	North	789	21
19	92	101	North	785	22
20	2	101	North	769	36
21	65	101	South	766	28
22	37	101	South	759	25
23	34	101	North	758	36

Figura 9: Datos en tiempo real.

Link a GitHub

A continuación se adjunta el lenguaje de acceso libre del proyecto: CuandoLlega.

Bugs y problemas de diseño

El diseño del código fue realmente complejo debido a la flexibilidad que se quiso dar como objetivo y enfoque principal del programa. Manejar archivos de datos, manipularlos, iterar en ellos para extraer información y convertirla a instancias de objetos, es un dolor de cabeza. La aparición de bugs durante el proceso fue constante, pero durante el desarrollo se atacaron los problemas uno por uno, basándonos en diagramas y esquemas para poder entender la raíz de estos.

Uso de la IA GitHub Copilot

El código en su totalidad fue diseñado con la ayuda del plugin lanzado este año llamado GitHub Copilot, lo cual sin lugar a dudas fue una ayuda y una forma de trabajar totalmente nueva, y quizás la del futuro, ya que esta IA mediante procesos de *machine learning*, logra adaptarse al contexto de la creación, tal como se muestra aquí:

```java
public Bus[] simulateBuses(Bus[] buses){

    // Iterates every bus and sets all attributes to 0
    for (int i = 0; i < buses.length; i++) {
        String simulatedDirection; // Sets to random value between "North" and "South"
        int random = (int) (Math.random() * 2);
        if (random == 0)
            simulatedDirection = "North";
        else
            simulatedDirection = "South";

        int simulatedRecPercentage; // Sets to random value between 0 and 100
        random = (int) (Math.random() * 1000);
        simulatedRecPercentage = random;

        int simulatedAverageSpeed; // Sets to random value between 20 and 45
        random = (int) (Math.random() * 26) + 20;
        simulatedAverageSpeed = random;

        buses[i].setId(i + 1);
        buses[i].setDirection(simulatedDirection);
        buses[i].setRecPercentage(simulatedRecPercentage);
        buses[i].setAverageSpeed(simulatedAverageSpeed);
    }
    return buses;
```

Figura 10: Uso de la IA GitHub Copilot. Aquí el programador sólo escribe los comentarios, para que la IA autocomplete todo el código a partir de su modelo de aprendizaje.