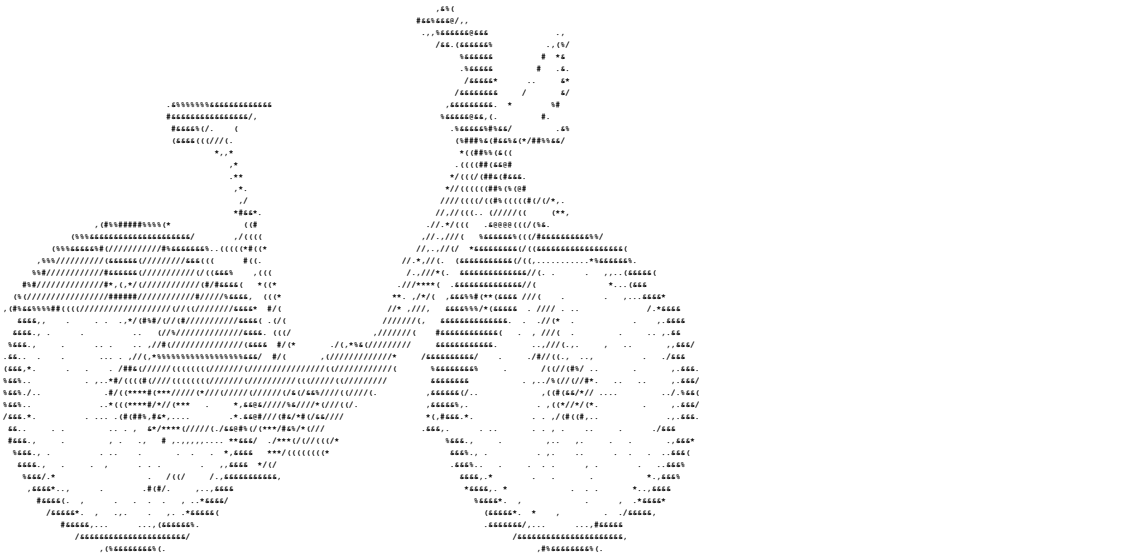
Universidad de Buenos Aires – Facultad de Ingeniería
75.40 Algoritmos y Programación I
Cátedra Pablo Guarna

Trabajo Práctico Grupal

BAEcoBici v2.0



1º Cuatrimestre 2019 - 2do Trabajo Práctico

Objetivo

El objetivo del trabajo práctico es ampliar las funcionalidades desarrolladas en la primera versión, en donde se implementaron distintas funciones para realizar simulaciones del sistema de BAEcoBici de la ciudad de Buenos Aires. La atención estará puesta en la utilización de archivos y la persistencia. Para esto, se detallan a continuación las distintas entidades y archivos con los que cuentan.

Entidades y archivos

El problema cuenta con los siguientes archivos y entidades:

- **Estación de bicicletas**
  - Identificador: numérico empezando en 1
  - Dirección: string
  - Latitud y longitud: números con decimales, pueden ser negativos
  - Capacidad: entre 15 a 30 anclajes
  - Bicicletas: arreglo de bicicletas ancladas, los anclajes comienzan en 1

Archivo de estaciones: contiene todas las estaciones necesarias para llevar adelante las simulaciones.

- **Bicicletas**
  - Identificador: numérico empezando por el número 1000
  - Estado: En condiciones, Necesita reparación
  - Ubicación: Anclada en estación, en circulación.

- **Usuario**
  - DNI: numérico entre 7 y 8 dígitos
  - PIN: numérico de 4 dígitos, en blanco si el usuario está bloqueado
  - Nombre y apellido: string
  - Celular: string de dígitos y puede permitir los siguientes caracteres: (,), + y -. Validar un mínimo de 8 dígitos numéricos.

Archivos de usuarios: contienen usuarios para llevar adelante las simulaciones. Se encuentran ordenados por DNI en forma creciente. Son 4 archivos en total que tienen que combinarse en un único archivo maestro antes de ser procesado (ver el punto 2 de los requerimientos agregados).

- **Viajes en curso**
  - DNI del usuario
  - Identificador de la bicicleta
  - Estación de salida
  - Hora de salida (string en formato hhmmss)

No hay un archivo de viajes en curso.

- **Viajes finalizados**: análogo a la anterior, agregando:

- Estación de llegada
- Hora de llegada

Archivo de viajes: contiene todos los viajes realizados.

Especificaciones

Se requiere:

1) Generar una función que lea los archivos y genere todas las entidades necesarias para llevar adelante las simulaciones. Para esto, debe levantarse el archivo de bicicletas, y asignar a cada estación una cantidad de bicicletas que sea mayor a su capacidad y menor a 30. Puede hacerse en forma aleatoria hasta distribuir todas las bicicletas o bien pueden ir llenando según un criterio acordado por el grupo.
2) Generar una función que permita dar de alta a un usuario, ídem a la v1.0.
3) Permitir que un usuario ingrese al sistema, ídem a la v1.0.
4) Permitir desbloquear a un usuario, ídem a la v1.0.
5) Permitir que un usuario retire una bicicleta, ídem a la v1.0.
6) Permitir que un usuario devuelva una bicicleta, ídem a la v1.0.
7) Permitir generar una cantidad de entre 1 y 100 simulaciones aleatorias, ídem v1.0. Para el punto anterior y para este se pide que se agregue al informe del viaje un dato más: la distancia recorrida en km. Para esto, deben utilizar las coordenadas reales provistas en el archivo de estaciones. Para sacar la distancia entre dos estaciones deben utilizar la fórmula de Haversine. Pueden utilizar una librería que las provea como puede ser geopy o bien puede programar su propia fórmula. Recomendamos la primera de las opciones.
8) Generar los mismos informes que en la versión v1.0.

Adicionales

1) Al salir del programa, se deben persistir todos los movimientos nuevos que se hayan dado durante la ejecución. Es decir:
    a) Se deben guardar en el archivo de usuarios aquellos usuarios dados de alta en el sistema. Estos deben ser guardados en el archivo maestro de usuarios (ver el siguiente punto).
    b) Se deben guardar en el archivo de viajes los viajes FINALIZADOS durante la ejecución.
    c) En el caso de que haya algún viaje en curso y se quiera salir del programa, los viajes en curso deben ser persistidos en un archivo en formato binario, y apenas se inicia nuevamente el programa, se deben completar las acciones para finalizarlos (es decir que el usuario que sacó la bicicleta no será el que realice la devolución, sino que el sistema mismo completará la devolución en una estación elegida aleatoriamente). La forma y los datos persistidos deben ser acordados a criterio del grupo. Recordar que esta PROHIBIDO persistir un diccionario completo.
2) El archivo maestro de usuarios tienen que generarse a partir de los 4 archivos parciales haciendo un merge. Cada archivo de usuarios viene ordenado por el DNI

del usuario en forma ascendente, es decir de menor a mayor. El maestro tiene que quedar ordenado bajo el mismo criterio. Este archivo debe ser generado únicamente en caso de no existir el archivo maestro de usuarios (es decir, una vez que se hizo el merge, no debería volver realizarse a menos que se borre el archivo maestro). El proceso de importación de datos debe intentar generarlo a partir de los 4 subarchivos para luego procesarlo, es decir que debe ser generado cuando se llame a la función del primer punto de las especificaciones.

3) Robo de bicicletas: Dentro del menú de ingreso al sistema, deben agregar una opción para robar una bicicleta. El flujo del robo consta de las siguientes etapas y consideraciones:
    a) Loguearse con identificador y PIN, realizando las validaciones correspondientes.
    b) Preguntar al usuario el ID de la bicicleta desea robar.
    c) La bicicleta que se quiere robar debe estar siendo utilizada por otro usuario. Esto quiere decir que para realizar un robo, la bicicleta debe haber sido retirada por un usuario previamente, pero no debe haber sido devuelta. En caso de intentar utilizar una bicicleta anclada a una estación, se bloquea al usuario que intentó realizar el robo.
    d) Si se efectúa el robo, la bicicleta debe pasar a estar en propiedad del usuario asaltante. Esto quiere decir que quien realice la devolución debe ser el usuario que realizó el robo. En caso de que quien intente realizar la devolución sea el usuario asaltado, deben imprimir por pantalla el mismo mensaje de error que imprimían en caso de que un usuario que no está usando una bicicleta intente realizar una devolución.
    e) Consideración: La duración del viaje no varía en caso de que ocurra un asalto.
    f) Consideración: Únicamente puede haber un robo por viaje. En caso de que un ladrón quiera robar a otro ladrón, deben imprimir un mensaje de error.
4) Reporte de robos: A los reportes que había generado en el TP1, hay que agregar uno más. Este reporte es sumamente simple: deben listar los viajes en los que sucedió un robo en un formato a discutir a criterio del grupo. Deben tener en cuenta las distintas ejecuciones del programa, por lo que se debe llevar un archivo en un formato a decidir por el grupo donde se vayan guardando los viajes para los cuáles hubo un robo.

Forma de entrega

El trabajo práctico debe ser entregado vía campus el 27/06/2019 hasta las 23:55h. Entregas fuera de tiempo no podrán ser subidas. El archivo debe estar comprimido en un zip que lleve el formato “${nombreGrupo} - TP2” (deben reemplazar ${nombreGrupo} por el nombre de su grupo). Dentro del zip debe haber un archivo TXT donde se debe incluir nombre, apellido y padrón de los miembros del equipo. También se deben realizar todas las

aclaraciones que a su criterio sean necesarias, explicando la funcionalidad del código cuando sea relevante. Validar con cada ayudante asignado si requerirán además una versión impresa del código fuente.

Se vuelve a remarcar que no deben realizarse copias. Una detección de copias entre dos grupos tiene como consecuencia la expulsión inmediata de los integrantes de ambos grupos de

la cátedra y la posibilidad de apertura de un sumario. Lo mismo puede suceder con un integrante que no haya trabajado durante el desarrollo del trabajo práctico.