Universidad Tecnológica Nacional

# Facultad Regional Avellaneda

### Recuperatorio Laboratorio de Programación 1

### Profesor: Scarafilo, German

## División: 1C

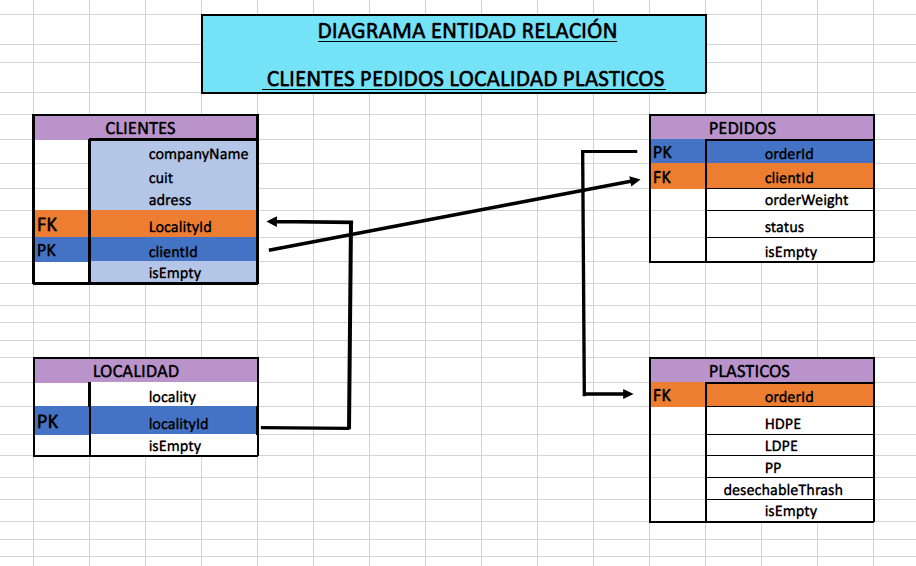
### Alumno: Lucchetta, Giovanni

### Turno: Mañana

1. [**github.com/GioLucc/Recuperatorio-Parcial-Laboratorio-1**](https://github.com/GioLucc/Recuperatorio-Parcial-Laboratorio-1)
2. [**https://drive.google.com/file/d/1dzkznJBRTff7K-b6x5rUUv\_PpgX\_zP7U/view?usp=sharing**](https://drive.google.com/file/d/1dzkznJBRTff7K-b6x5rUUv_PpgX_zP7U/view?usp=sharing)

**3**

1. ✔️
2. La estructura agregada fue Plastics que da referencia a los tipos de plásticos que se pueden reciclar después de crear un pedido e ir a procesarlo, esta se relaciona explícitamente con la estructura RecolectionOrders o Pedidos, Plastics obtiene la PK de id de orden de la estructura RecolectionOrders para procesar los tipos de plásticos del Id de ese pedido, y así RecolectionOrders pueda manipular todos los reciclados mediante el Id pedido.



1. Informe N°14: Ingrese una localidad para filtrar la cantidad de HDPE - LDPE - PP - Desechable Trash y así mostrar el total reciclado de esa zona.

**Prototipos núcleo**

**Clients**

/// @fn int newClient(eClients\*, sLocality\*, int, int\*, int\*)

/// @brief Pide y valida los datos para la creación de un nuevo cliente.

///

/// @param clientList

/// @param localitiesList

/// @param len

/// @param uniqueID

/// @param uniqueLocalityID

/// @return -1 si no se logro, 0 si se pudo.

**int newClient (eClients\* clientList, sLocality\* localitiesList, int len, int\* uniqueID, int\* uniqueLocalityID);**

/// @fn int addClient(eClients\*, int, char[], char[], char[], int, int)

/// @brief Alta de cliente, obtiene una posición libre y carga todos los datos de los parametros en esa posición.

///

/// @param clientList

/// @param len

/// @param companyName

/// @param cuit

/// @param address

/// @param localityId

/// @param id

/// @return devuelve 1 si se pudo crear.

**int addClient (eClients\* clientList, int len, char companyName [], char cuit[], char address [], int localityId, int id);**

/// @fn int modifyClient(eClients\*, sLocality\*, int, int\*)

/// @brief Pide ingresar el id de el cliente que quiere modificar y permite cambia dirección y localidad.

///

/// @param clientList

/// @param localitiesList

/// @param len

/// @param uniqueLocalityID

/// @return devuelve 1 si pudo hacerlo satisfactoriamente.

**int modifyClient (eClients\* clientList, sLocality\* localitiesList, int len, int\* uniqueLocalityID);**

/// @fn int idAsk(eClients\*, sLocality\*, int)

/// @brief Pide un id mostrando la lista de clientes existentes.

///

/// @param clientList

/// @param localitiesList

/// @param len

/// @return La posición para usar.

**int idAsk(eClients\* clientList, sLocality\* localitiesList, int len);**

/// @fn int idValidation(eClients\*, sLocality\*, int, int)

/// @brief Se fija si la posición recibida por parametros existe.

///

/// @param clientList

/// @param localitiesList

/// @param len

/// @param enteredId

/// @return -1 si no existe, la posición si existe.

**int idValidation (eClients\* clientList, sLocality\* localitiesList, int len, int enteredId);**

/// @fn int withdrawalClient(eClients\*, sLocality\*, int)

/// @brief Da de baja lógica a un cliente.

///

/// @param clientList

/// @param localitiesList

/// @param len

/// @return devuelve -1 si no se pudo dar de baja, 0 si se pudo dar de baja.

**int withdrawalClient (eClients\* clientList, sLocality\* localitiesList, int len);**

**Prototipos núcleo**

**Orders**

/// @fn int initOrders(sOrders\*, int)

/// @brief Inicia el vector de ordenes en vacio en todas las posiciones.

///

/// @param ordersList

/// @param len

/// @return devuelve -1 si no se pudo realizar y 0 si se pudo.

**int initOrders (sOrders\* ordersList, int len);**

/// @fn int freeOrderSpot(sOrders\*, int)

/// @brief Busca un espacio libre en el vector para la creación de una nueva orden.

///

/// @param ordersList

/// @param len

/// @return devuelve -1 si no se pudo realizar y 0 si se pudo.

**int freeOrderSpot (sOrders\* ordersList, int len);**

/// @fn int newOrder(eClients\*, sOrders\*, sLocality\*, int, int\*)

/// @brief Pide todos los datos para crear una nueva orden.

///

/// @param clientList

/// @param ordersList

/// @param localitiesList

/// @param len

/// @param uniqueOrderID

/// @return devuelve -1 si no se pudo realizar y 0 si se pudo.

**int newOrder (eClients\* clientList, sOrders\* ordersList, sLocality\* localitiesList, int len, int\* uniqueOrderID);**

/// @fn int addOrder(sOrders\*, int, int, int, int, int)

/// @brief Da de alta una nueva orden relacionandola a un cliente existente.

///

/// @param ordersList

/// @param len

/// @param clientId

/// @param orderWeight

/// @param orderId

/// @param status

/// @return devuelve -1 si no se pudo realizar y 0 si se pudo.

**int addOrder (sOrders\* ordersList, int len, int clientId , int orderWeight, int orderId, int status);**

/// @fn int showOneOrder(sOrders)

/// @brief Muestra 1 orden.

///

/// @param ordersList

/// @return devuelve -1 si no se pudo realizar y 0 si se pudo.

**int showOneOrder (sOrders ordersList);**

/// @fn int showListOfOrders(sOrders\*, int)

/// @brief Muestra la lista completa de ordenes.

///

/// @param ordersList

/// @param len

/// @return devuelve -1 si no se pudo realizar y 0 si se pudo.

**int showListOfOrders (sOrders\* ordersList, int len);**

/// @fn int processingOrder(eClients\*, sOrders\*, sPlastics\*, int, int)

/// @brief Pide un id de una la lista de pedidos pendientes para procesarla.

///

/// @param clientList

/// @param ordersList

/// @param plasticsList

/// @param clientsLen

/// @param ordersLen

/// @return devuelve -1 si no se pudo realizar y 0 si se pudo.

**int processingOrder (eClients\* clientList, sOrders\* ordersList, sPlastics\* plasticsList, int clientsLen, int ordersLen);**

**Prototipos núcleo**

**Locality**

/// @fn int LOC\_init(sLocality\*, int)

/// @brief Inicia el vector de localidades en EMPTY.

///

/// @param localitiesList

/// @param localitiesLen

/// @return 0 si pudo -1 si no pudo.

**int LOC\_init (sLocality\* localitiesList, int localitiesLen);**

/// @fn int LOC\_hardcodeLocalities(sLocality\*, int, int\*)

/// @brief Harcodea localidades

///

/// @param localitiesList

/// @param len

/// @param uniqueLocalityID

/// @return0 si pudo -1 si no pudo.

**int LOC\_hardcodeLocalities (sLocality\* localitiesList, int len, int\* uniqueLocalityID);**

/// @fn int LOC\_freeSpot(sLocality\*, int)

/// @brief Busca un lugar libre en el vector para crear una nueva localidad.

///

/// @param localitiesList

/// @param localitiesLen

/// @return 0 si pudo -1 si no pudo.

**int LOC\_freeSpot (sLocality\* localitiesList, int localitiesLen);**

/// @fn int LOC\_newLocality(sLocality\*, int, int\*)

/// @brief Pide todos los datos para una nueva localidad y se los pasa LOC\_chargeLocalitiesList

///

/// @param localitiesList

/// @param len

/// @param uniqueLocalityID

/// @return 0 si pudo -1 si no pudo.

**int LOC\_newLocality(sLocality\* localitiesList, int len, int\* uniqueLocalityID);**

/// @fn int LOC\_withdrawal(sLocality\*, int)

/// @brief Da de baja logica una localidad.

///

/// @param localitiesList

/// @param localitiesLen

/// @return 0 si pudo -1 si no pudo.

**int LOC\_withdrawal (sLocality\* localitiesList, int localitiesLen);**

/// @fn int LOC\_idAsk(sLocality\*, int)

/// @brief Pide un id para dar de baja una localidad.

///

/// @param localitiesList

/// @param localitiesLen

/// @return La posicion si existe o -1 si no existe.

**int LOC\_idAsk(sLocality\* localitiesList, int localitiesLen);**

/// @fn int LOC\_idValidation(sLocality\*, int, int)

/// @brief Valida si existe un id para dar de baja una localidad.

///

/// @param localitiesList

/// @param localitiesLen

/// @param enteredId

/// @return La posicion si existe o -1 si no existe.

**int LOC\_idValidation (sLocality\* localitiesList, int localitiesLen, int enteredId);**

/// @fn int LOC\_chargeLocalitiesList(sLocality\*, int\*, int, char[])

/// @brief Alta de nueva localidad

///

/// @param localitiesList

/// @param uniqueLocalityID

/// @param localitiesLen

/// @param auxLocality

/// @return 0 si la pudo crear o -1 si no pudo crearla.

**int LOC\_chargeLocalitiesList(sLocality\* localitiesList, int\* uniqueLocalityID, int localitiesLen, char auxLocality []);**

**Prototipos núcleo**

**Plastics**

/// @fn int PLA\_initArray(sPlastics\*, int)

/// @brief Inicia el Array de ordenes en vacio.

///

/// @param plasticsList

/// @param plasticsLen

/// @return devuelve -1 si no se pudo realizar y 0 si se pudo.

**int PLA\_initArray (sPlastics\* plasticsList, int plasticsLen);**

/// @fn int PLA\_freeArraySpot(sPlastics\*, int)

/// @brief busca una posicion disponible para guardar el pedido procesado junto con todos sus plasticos.

///

/// @param plasticsList

/// @param plasticsLen

/// @return devuelve -1 si no se pudo realizar y 0 si se pudo.

**int PLA\_freeArraySpot (sPlastics\* plasticsList, int plasticsLen);**

/// @fn int PLA\_processPlastics(sPlastics\*, int, int, int)

/// @brief Procesa el pedido completo que le llega por parametros desde la funcion de procesar de ordenes.

///

/// @param plasticsList

/// @param plasticsLen

/// @param id

/// @param weightOrder

/// @return devuelve -1 si no se pudo realizar y 0 si se pudo.

**int PLA\_processPlastics (sPlastics\* plasticsList, int plasticsLen, int id, int weightOrder);**

| **Universidad Tecnológica Nacional Facultad**  **Regional Avellaneda** | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Técnico Universitario en Programación  Técnico Universitario en Sistemas Informáticos | | | | | | | | |
| Materia: LABORATORIO I | | | | | | | | |
| Apellido: | Lucchetta | | | Fecha: | | 24/11/2021 | | |
| Nombre:  División: | Lucchetta Giovanni  C | | | Docente:  Nota: | | Scarafilo German | | |
| Legajo: |  | | | Firma: | |  | | |
| Instancia | PP |  | RPP X | SP |  | RSP | FIN |  |

**Recuperatorio Laboratorio**

**Primer parcial**

1. **Alumnos que hayan obtenido un 4 y deseen levantar nota:** deberán presentar el mismo trabajo que en la instancia anterior con las modificaciones que marcamos en las devoluciones (Parte 1 y 2 del parcial).

2. **Alumnos que aún no hayan aprobado el parcial**: deberán presentar el mismo trabajo que en la instancia anterior (parte 1 y 2) agregando lo siguiente: a. Una estructura más, debidamente relacionada con las anteriores. La composición de la estructura deberá ser compleja: un id, un campo alfanumérico y uno entero o flotante (para utilizar en algún filtro).

b. Un filtro **complejo distinto a los trabajados con anterioridad**. Este filtro deberá utilizar la mayoría de las estructuras incluyendo la estructura agregada en el ítem anterior.

**¿Qué material debo entregar?**

1. Proyecto completo en eclipse.

2. Un video entre 10 y 15 minutos de duración, en donde deberán realizar la defensa del proyecto. Recuerden que deberán explicar, sin leer el código (pero si mostrándolo), las funciones más importantes del programa. Pueden tomarse unos minutos para demostrar cómo funciona la app. Tengan en cuenta que el video debe evidenciar las funciones más importantes del programa (Altas, Bajas, Modificaciones, Filtros complejos). Deberán estar con cámara prendida (Por ejemplo pueden presentar escritorio con zoom y estar con cámara prendida).

3. Un documento con:

a. Carátula con datos del alumno y la materia.

b. El detalle de la estructura agregada y su relación con las otras estructuras.

Realizar un diagrama entidad relación DER. (2b)

c. El enunciado del informe anexado (2b).

d. Prototipo de cada función del núcleo del programa con la documentación correspondiente.

e. Enlace a drive con el video explicativo.

Aclaraciones:

El trabajo deberá ser subido ÍNTEGRAMENTE a un repositorio de Github que deberán informar mediante el formulario que está en el espacio del examen. No se corregirán parciales que sean entregados por otro medio.

El video se debe ver y oír de manera clara. Por favor chequeen este punto antes de entregar. **Fecha de entrega:**

Viernes 26/11 Hasta las 18 Hs.