# Trabajo Práctico Estructura de Datos y Programación

Instituto Tecnológico de Buenos Aires

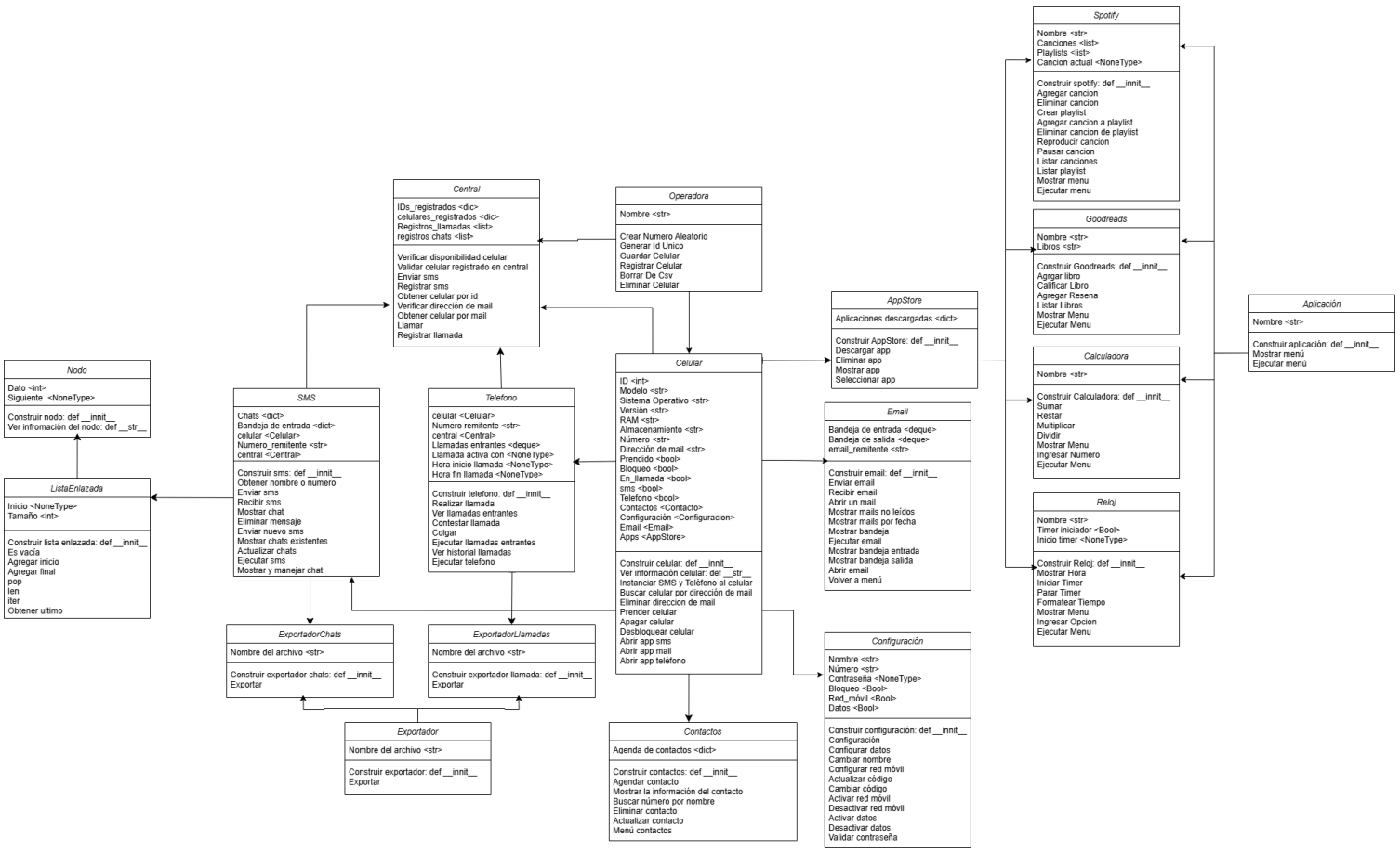
Fecha de entrega: 11/11/2024

Giorgia Barbería (65713), Pilar Basteguieta (65465), Maria Belén Böhtlingk (65162), Joaquin Ignacio Mendivil (65592)

## Aclaración:

Todas las decisiones fueron tomadas con base en lo que consideramos más cercano a la realidad.

## Diagrama de clases:




https://app.diagrams.net/#G1fHZpt-LPpS8H3vMsoELR2s7xrC-unhbj#%7B%22pageId%22%3A%22C5RBs43oDa-KdzZeNtuy%22%7D

## Instructivo de ejecución del programa:

El programa principal se ejecuta desde el archivo menu.py.
El análisis de datos se ejecuta desde el archivo analisis_datos.py.

## Instructivo de ejecución de llamada por pasos:

Debido a la naturaleza de ejecución del trabajo práctico, y las limitaciones de solo poder ejecutar en una terminal, el proceso de llamada entre dos celulares resultó ser no del todo intuitivo. Por esta razón, se detallan a continuación los pasos a seguir:

1. Asegurarse de tener dos celulares creados. Los llamaremos celular1 y celular2 para que se entienda mejor la explicación.
2. Entrar en celular1 para prenderlo, salir presionando 0 para dejar la red móvil prendida (que es lo que permite que pueda recibir llamadas).
3. Entrar en celular2, entrar a teléfono, y llamar al celular1. Salir presionando 0 para dejar la red móvil prendida (si se apaga se cortará la llamada).
4. Entrar nuevamente al celular1, que ahora está recibiendo la llamada, y decidir si atender o colgar sin atender.

## Consideraciones:

**Menús**:

- Lógica: Implementamos un menú principal donde el usuario puede elegir entre ingresar como celular u operadora, con submenús específicos para cada uno.
- Opciones de Menú: En algunos menús, se saltan números (por ejemplo, opciones 1, 2, 3, 6, 7, 0). Consideramos esto correcto porque en los menús en el celular, las opciones no son siempre consecutivas.

**Validaciones**:

- Programación defensiva: Como a futuro nuestro programa podría expandirse, y se podrían rehusar funciones, muchas veces validamos que un dato sea válido en varios pasos de su recorrido por funciones, aunque sepamos que por como es nuestro programa jamás le llegará un dato inválido, así previniendo futuros errores.

**Central**:

- Registros de llamadas: se pensó como una lista, donde por cada llamada se agrega un diccionario, ya que almacena todos los atributos del evento en una sola estructura y facilita el acceso a un registro.
- IDs y Números de Celular:
    - Los IDs son permanentes y únicos, mientras que los números de celular pueden reutilizarse una vez que un celular se elimina.
    - Todos los celulares tienen la misma central.
    - Ids y celulares registrados son diccionarios con claves de id y número de celular, que son únicos. Al eliminar un celular, se elimina de su diccionario ya que este muestra los activos.

Operadora:

- Al registrar un celular, se le asigna un número aleatorio, simulando la realidad donde los usuarios no eligen su número.

Email:

- Por simplicidad, asumimos que cada celular solo puede y debe tener un email, y este no se puede repetir. Esto se valida mediante un set como atributo de la clase celular, que es ideal para evitar duplicados.
- Al eliminar un celular, el email vuelve a estar disponible.
- Para poder enviar un email, se debe ingresar al menos un carácter como asunto, o uno como cuerpo.
- En la bandeja de entrada y salida de emails decidimos usar pilas, ya que facilita el acceso rápido a los correos más recientes, permite la inserción eficiente de nuevos mensajes, y simula el comportamiento natural de visualización de correos electrónicos donde los más recientes son los primeros en aparecer.

App Store:

- Si una aplicación ya está descargada, el usuario puede elegir entre ingresar a ella o volver al menú, simulando el comportamiento real de una App Store.

Contactos:

- Validaciones al agendar contactos:
    - Al agendar un contacto, se valida que el número telefónico tenga un formato válido, pero no se verifica su existencia en la central. Se tomó esta decisión porque así funciona la app contactos.
    - Si se elimina un número, las personas que lo tenían agendado seguirán teniendo el contacto.
    - Al mandar un mensaje o llamada a un número inexistente, se notifica al usuario.
- Se permite agendar múltiples contactos con el mismo nombre. En caso de querer mandarle un mensaje o llamar a uno de esos contactos por nombre, entonces se le mostrarán al usuario los números correspondientes a todos los contactos con ese nombre, y el usuario deberá elegir a cual se refería.

SMS:

- Es posible mandar mensajes de un celular a ese mismo celular.
- Cuando se pregunta a qué chat se desea ingresar, el usuario puede poner el nombre de contacto o el número telefónico e intenta encontrarlo por ambos parámetros (no se pregunta si desea buscar por contacto o por número telefónico).
- Los chats los almacenamos en un diccionario donde la clave es el número telefónico con el que se está chateando y el valor es una lista enlazada que contiene los mensajes de este chat. Implementamos esta estructura de datos para que así el usuario pueda eliminar el mensaje que quiera del chat, ya que si hubiéramos implementado una pila solamente debería poder eliminar el último mensaje del chat.

- Se usa el metodo magico __iter__ en la clase de lista enlazada para convertir la clase en iterable, esto hace que se puedan recorrer en un bucle como for, devolviendo cada dato de los nodos en secuencia
- La opción eliminar mensaje sólo aparecerá si ya existe una conversación con el número seleccionado.
- Para eliminar un mensaje, figuran por índice los mensajes de la conversación, y se debe ingresar el que corresponda para poder eliminarlo. Este, una vez eliminado, aparece como "mensaje eliminado" **y con la hora en la que se eliminó.**
- Mientras un celular esté apagado (y sin red), no podrá recibir mensajes. Los mensajes que le manden a este celular serán almacenados en un diccionario "bandeja_entrada", hasta que se vuelva a prender. Ahí, los mensajes de "bandeja_entrada" se transfieren al diccionario de chats del celular, y "bandeja_entrada" se blanquea.

Configuración:

- Añadimos la opción "visualizar información" para que cada celular pueda ver sus propios datos. Nos pareció importante porque sino todos los datos que se ingresaron al crear el celular no tendrían ninguna utilidad (almacenamiento, RAM, etc.)

Teléfono:

- Las llamadas entrantes se almacenan en una cola, ya que queríamos que se almacenen en orden de llegada (First In First Out).
- En caso de tener más de una llamada entrante, se deberá decidir qué hacer con la primera (atender o no), luego la segunda, y así sucesivamente.
- Si tiene llamadas entrantes y el celular se prende, entonces antes de poder usar cualquiera de las funcionalidades del celular se debe atender o no dichas llamadas. Tomamos esta decisión ya que en la realidad, si alguien está llamando a un celular y este se prende, lo primero que aparecerá será la llamada.
- Para poder llamar a un número y que la llamada salga, el otro celular debe estar encendido (y conectado a la red). Para esto es que pusimos la opción de "salir y dejar el celular prendido en el menú principal del celular".
- Si un celular está en llamada y desactiva su red móvil, automáticamente se corta la llamada.

Eliminar apps:

- Eliminar app está en el menú, y no una vez que se ingresa a la app porque consideramos que el menú principal del celular es su pantalla de inicio. En la realidad para eliminar una app hay que mantenerla apretada desde la pantalla de inicio (sin ingresar en ella), entonces simulamos esto cómo seleccionar la opción 7 (Eliminar app).

- Si no está descargada ninguna app, no aparece la opción eliminar en el celular por lo asumido anteriormente con eliminar apps. Como interpretamos esta función del celular como mantenerla apretada, si no hay ninguna app no hay ninguna que se pueda mantener apretada.
- Asumimos que las aplicaciones como teléfono, sms y email no se pueden descargar ni eliminar, ya vienen por default en cada celular.

## Análisis de datos:

Gráfico 1: Cantidad de installs según tamaño de la dating app (dispersión, logarítmico):

- Se utilizó un gráfico de dispersión para identificar patrones.
- Se utilizó una escala logarítmica porque al haber mucha diferencia entre la cantidad de instalaciones, una escala lineal habría hecho muy difícil de comprender el gráfico y analizarlo apropiadamente.
- Para un desarrollador es importante saber si se descargan más las aplicaciones más ligeras o las que tienen más funcionalidades y son más pesadas. Se consideró que las apps válidas tengan un rating mayor a 2.8 pues las apps con un rating bajo son más probables de ser eliminadas, y por lo tanto no generarán ganancia.
- Además, se puede visualizar la competencia. Para una empresa, es importante poder posicionarse mejor en el mercado y distinguirse y para eso se debe conocer al usuario.

Gráfico 2: Top 5 categorías más demandadas (barras):

- El éxito se midió en cantidad de installs, siempre que el promedio del rating de las apps de dicha categoría tengan un rating mayor a 3,7. Esta información es relevante porque al desarrollar una app, es importante conocer las tendencias actuales del mercado.
- Además, al invertir, es importante tener en cuenta donde hay más probabilidad de un buen rendimiento financiero, y al considerar un rating mayor a 3.7 hay más probabilidad de que los usuarios sean leales a las apps de esas categorías.

Gráfico 3: Distribución de Ratings para la categoría "SPORTS" (histograma):

- Esta información permite a desarrolladores de apps de deportes conocer la calidad de su app en comparación con otras dentro del mismo rubro. No es lo mismo tener un rating de 3.5 en un mercado donde la mayoría tiene uno de 2.0, que en uno donde los ratings son muy altos.
- Además, conocer categorías donde las aplicaciones tienen calificaciones bajas puede servir a las empresas para reconocer áreas donde existe la necesidad de una buena app.

Gráfico 4: Distribución de ratings por género en la categoría 'GAME' (dispersión):

- Se trata de un gráfico más específico que el anterior, cuyo objetivo es encontrar nichos. Permite a las empresas que se dedican a juegos, identificar qué géneros son los que más les gustan a las personas.
- Además, las empresas pueden conocer cuál es la competencia dentro de su mismo género. Esto es importante porque por ejemplo, a una app de Puzzles, le importará más la opinión de la gente sobre las apps de Puzzles que de las de juegos en general.

Gráfico 5: Cantidad de instalaciones y rating promedio según año de la última actualización (gráfico de línea de dos ejes):

- Permite saber que las apps cuya última actualización ha sido más reciente tienen mucho más éxito que las que no se actualizan hace mucho puede justificar la inversión para desarrolladores de actualizar sus apps más regularmente.
- Además, es interesante si por ejemplo se observa que las apps actualizadas en 2016 tienen ratings muy alto, entonces se podría analizar qué es lo que se hizo en 2016 que logró esto, y aplicar lo descubierto en la actualidad.