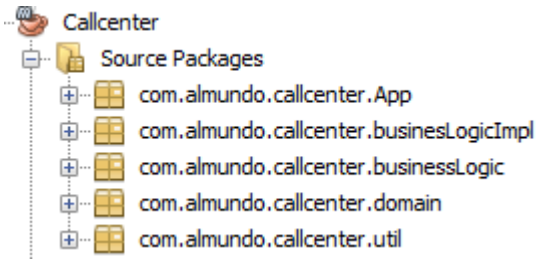
CALLCENTER ALMUNDO

Para el desarrollo de la aplicación se pensó en una solución que no estuviera diseñada solo para unos requerimientos limitados, si no que pudiera ser escalable.

La aplicación tiene la siguiente arquitectura



El **paquete App** contiene la clase Main para ejecutar la aplicación.

El **paquete businessLogic** contiene interfaces con métodos referencia a ser utilizados en la aplicación. Esto con el propósito de manejar inyección de dependencias y hacer que la aplicación sea escalable adaptandose por ejemplo a que sean varios callcenter los que se van a manejar y cada uno de ellos tengan propiedades diferentes.

Para nuestro caso se crean las interfaces:

- **BusinessThread:** Encargada de los métodos para el hilo principal de la en las operaciones del callcenter.

- **BusinessCall:** Encargada de manejar los métodos relacionados con las llamadas

- **Dispatcher:** Encargada de los métodos de asignación de las llamadas.

El **paquete businessLogicImpl** contienen clases que implementan las interfaces y adecua los métodos según los requerimientos del sistema.

- **BusinessThreadImpl:** Clase que implementa BusinessThread. En esta clase se establece el hilo executor del callcenter y se establece el pool de conexiones que este va a manejar.

- **BusinessCallImpl:** Clase que implementa la interfaz BusinessCall y implementa el método para la generación de llamadas y las envía al dispatcher.

- **DispatcherImpl:** Clase que implementa las interfaz Dispatcher. Implementa el método que asigna la llamada entrante a un recurso.

El **paquete domain** contiene los objetos propios del dominio

El **paquete util** contiene las clases que ayudan en la inicialización de elementos en la aplicación o clases con métodos que son generalizados y pueden ser utilizados en cualquier otra clase de cualquier paquete.

Contiene las clases :

- **Initializer**: donde se crean los empleados que van a atender las llamadas

- **Util**: donde se maneja un Atomic long para generar los id al momento de crear un empleado o una llamada. Y contiene un método para generar un numero randomico entre un rango establecido, el cual en este caso es la duración que puede tener una llamada.

Solución Extras

1. Dar alguna solución sobre qué pasa con una llamada cuando no hay ningún empleado libre

Uno de los problemas que debían ser abordados antes de iniciar el desarrollo, era la disponibilidad de los recursos del callcenter, en este caso los empleados.La solución a la que llegue, fue utilizar una PriorityBlockingQueue la cual implementa la interfaz BlockingQueue. Esta interfaz cuenta con un método en especial llamado Take() el cual funciona de la siguiente manera:

- Cuando una llamada trata de acceder a un recurso(un empleado) pero no hay disponible, este método bloquea el hilo que quiere adquirir el recurso mientras este no este disponible. En el momento en que se libere un recurso, este método automáticamente desbloquea el hilo, asigna el recurso y permite que termine su operación.

Adicional, mediante este PriorityBlockingQueue se maneja el orden en el que se deben contestar las llamadas según la jerarquía de los empleados. Ya que al momento de adicionar un nuevo recurso a la cola, esta lo agrega según la referencia de prioridad que se establezca.

2. Dar alguna solución sobre qué pasa con una llamada cuando entran mas de 10 llamadas concurrentes.

Cuando se generen mas de 10 llamadas concurrentes la aplicación las acepta pero si no hay recursos disponibles para contestarlas, las bloquea hasta que encuentre uno.

3. Agregar test unitarios que se crean convenientes

Se agregan Test unitarios para:

- Inicializar la cola de empleados.
- Comprobar el autoincremento del AtomicLong
- Generar un randomico entre un rango especifico.

- Verificar que el valor max siempre sea mayor que el minio para generar el randomico
- Que el rango máximo y el rango mínimo sean >= a 0
- Crear 10 llamadas
- Realizar 10 llamadas concurrentes
- Crear 20 llamadas
- Realizar 20 llamadas concurrentes
- Crear un hilo con 10 pools de conexiones y verificar si se encuentra en ejecución.
- Detener la ejecución del hilo principal

4. Agregar documentación de código

Cada uno de los métodos se documento.