

21-12-2023

Proyecto Práctico de aplicación Integrador: IVR

Cátedra: Diseño de Sistemas 2023

Grupo 7

Docentes:

- + Ing. Meles Judith
- + Ing. Lovay Mónica
- + Ing Sol Zanel María

Integrantes:

- + López Salvucci, Julián: 12405- julianls783@gmail.com
- + Fassetta, Julián: 12029- julianfassetta@gmail.com
- + Pizzi, Ana: 13928- anapizzi02@gmail.com



Facultad Regional
UTN VILLA MARIA

Contenido

Introducción.....	2
Patrón de diseño: Iterador.....	2
Patrón de diseño: Singleton.....	4
Diseño de interfaz de usuario.	6
Diseño de persistencia.	7

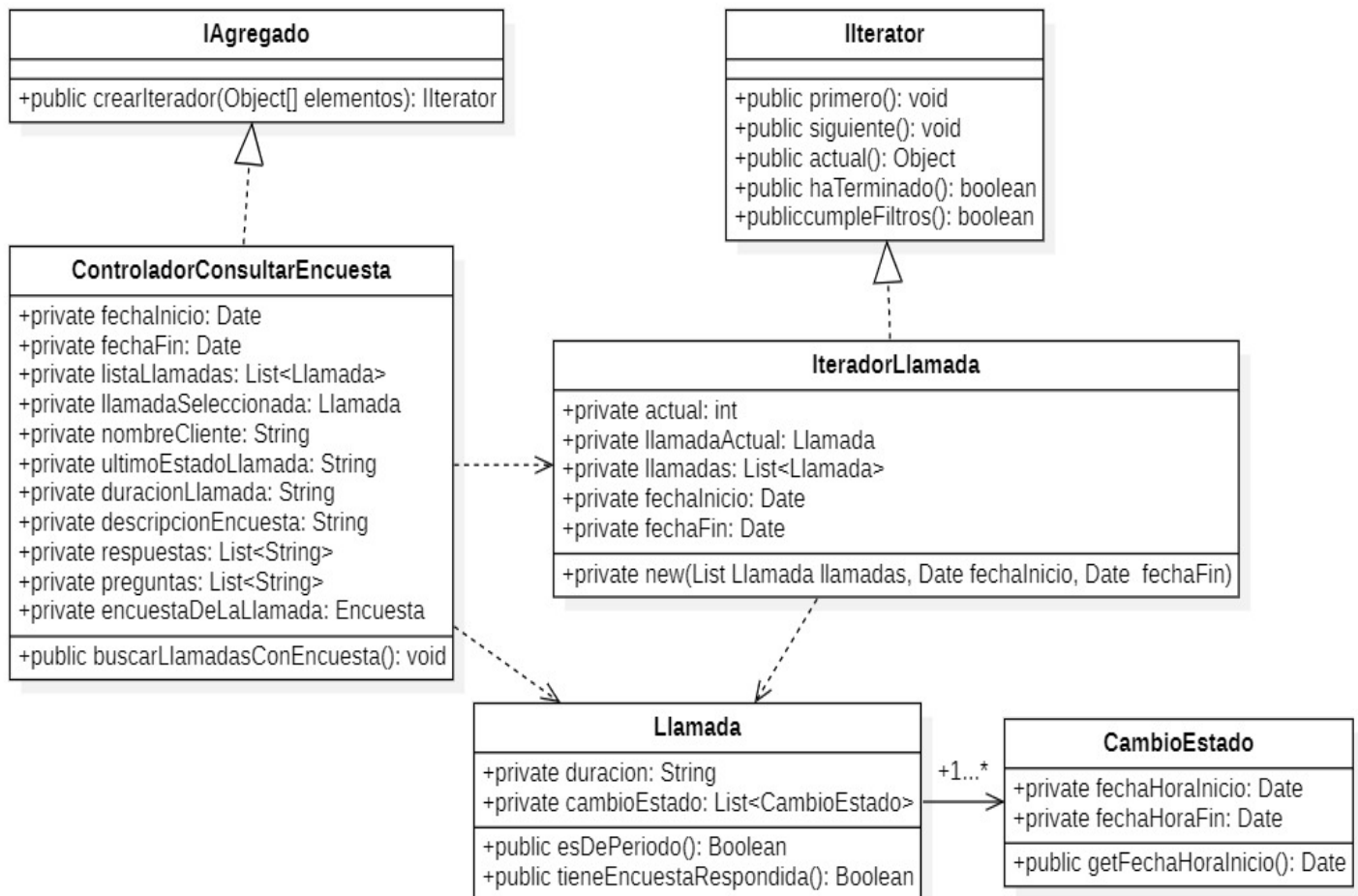
Introducción

En el presente documento se describen los aspectos de diseños trabajados sobre el **CU 44 Consultar encuesta**. Los cuales refieren al desarrollo de las capas de presentación (aplicando patrones IHM), lógica de negocio (Aplicando patrones de diseño Gamma) y persistencia (Aplicando capa de persistencia con fw de persistencia).

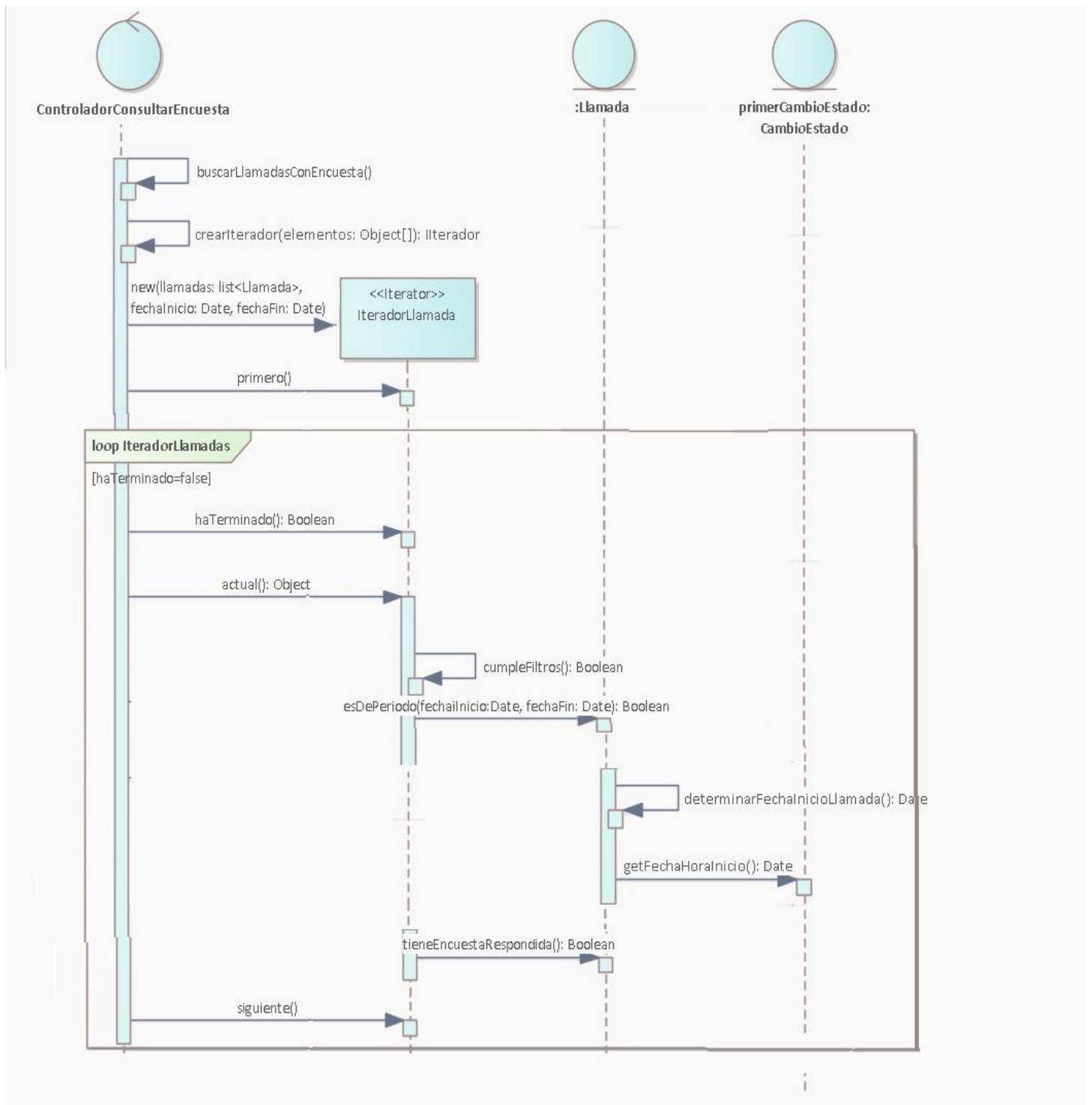
Patrón de diseño: Iterador

Aplicación: Para obtener las llamadas con encuesta asociada se aplica el patrón iterador, lo que permite delegar el recorrido y el filtrado a una nueva clase de fabricación pura iteradora.

Vista de estructura:



Vista dinámica:



Explicación/Pseudocódigo:

:ControladorConsultarEncuesta

- **buscarLlamadasConEncuesta():** Método de enganche con el análisis. El controlador inicia la búsqueda de llamadas con encuesta asociada.
- **crearIterador(elementos:Object): Iterator** : Método encargado de crear el iterador de llamadas.

:IteradorLlamada

- **new(llamadas: List<Llamada>, fechalnicio: Date, fechaFin: Date):** método constructor del iterador concreto que recibe por parámetros y el listado de elementos a filtrar (llamadas) y las fechas que servirán para filtrar.
- **primero()** : obtiene el primer elemento de la lista de llamadas.
- **haTerminado(): Boolean** : Corrobora si existen más elementos para iterar, caso contrario el ciclo concluye.
- **Actual(): Object** : Retorna el elemento actual de la lista en iteración.
- **cumpleFiltros(): Boolean** : Corrobora que se cumplan los filtros provistos.
- **Siguiente()** : Incrementa el índice de iteración en una unidad para acceder al siguiente elemento de la lista.

:Llamada

- **esDePeriodo(fechalnicio : Date, fechaFin : Date): Boolean** : Verifica si una determinada instancia de Llamada se encuentra comprendida entre el rango de fechas dado.
- **determinarFechalnicioLlamada():Date** : Retorna la fecha de inicio del primer cambio de estado que tiene la llamada, lo que también se traduce como la fecha de inicio de la llamada en concreto.
- **tieneEncuestaRespondida(): Boolean** : Verifica si tiene instancias de RespuestaDeCliente asociadas.

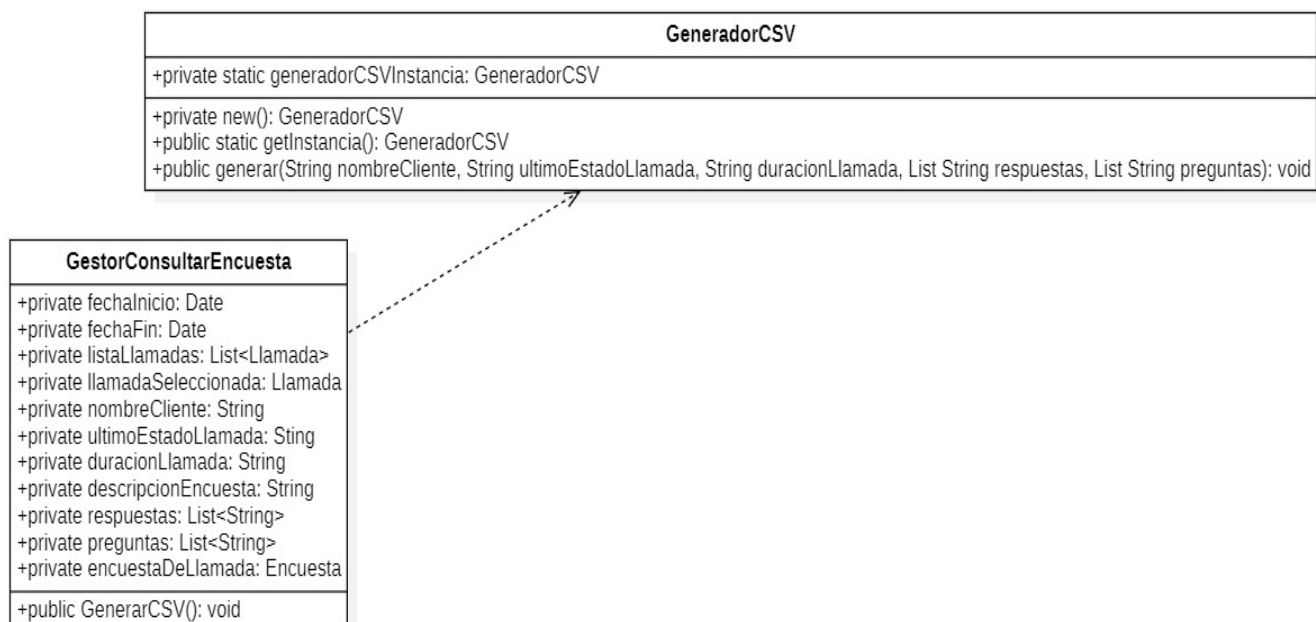
PrimerCambioEstado: CambioEstado

- **getFechaHoralnicio(): Date** : Retorna la fecha de inicio del primer cambio de estado asociado a la instancia de llamada consultada.

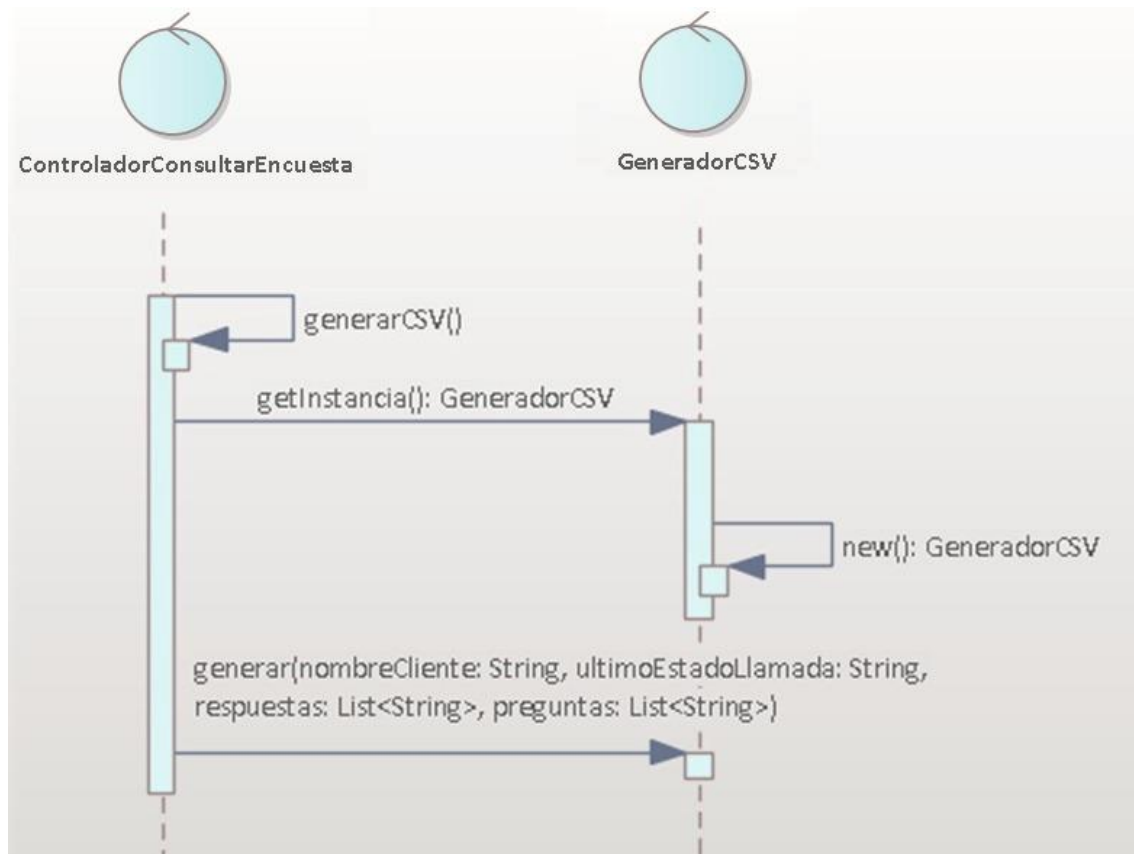
Patrón de diseño: Singleton

Aplicación: Para no generar múltiples instancias con el mismo propósito. Se aplica el patrón singleton para que solamente sea una única instancia la que se encargue de la generación de documentos CSV.

Vista de estructura:



Vista dinámica:



Explicación/Pseudocódigo:

:ControladorConsultarEncuesta

- **generarCSV()** : Método de enganche del análisis que delega la generación de documentos CSV a al nuevo objeto de la clase de fabricación pura **GeneradorCSV**.

:GeneradorCSV

- **getInstancia() : GeneradorCSV** : Operación que verifica la existencia de una instancia de tipo **GeneradorCSV**, si existe, retorna la misma, sino la crea. Es el método representativo del singleton.
- **new() : GeneradorCSV** : Método constructor.
- **generar(nombreCliente: String, ultimoEstadoLlamada: String, respuestas: List<String>, preguntas: List<String>)** : Comportamiento que contiene el algoritmo de generación de documentos CSV y que reciben por parámetros la información a plasmar en dicho documento.

Diseño de interfaz de usuario.

Para el desarrollo de la interfaz de usuario se aplicaron los siguientes patrones:




- **Lista Maestra/Detalle:** Para listar las llamadas y mostrar sus datos al seleccionar una en concreto, además de anexar paginación para no expandir por demás la pantalla.

Llamada		
Llamada 1		
Llamada 2		
Llamada 3		
Llamada 4		
Llamada 5		

LLAMADA SELECCIONADA		
Descripción de Encuesta:	preferencia y recurrencia	
Cliente:	Jose Terreno	
Estado Actual:	escucha_correcta	
Duración:	00:05:55.9278501	

Pregunta	DescripciónPregunta	RespuestaSeleccionada
volver a adquirir prod/serv	prod/serv no necesita revision	satisfactorio
recibir informacion	la informacion sobre prod/serv, anteriormente brindada al cliente fue excesiva	no satisfactorio

- **Patrón de Barra de Herramientas:** Las funcionalidades fueron ubicadas en una barra inferior

 Generar CSV	 Imprimir	 Cancelar
-------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------

Diseño de persistencia.

Para el diseño de persistencia se desarrolló una nueva capa la cual es encargada de administrar las funciones de materialización de desmaterialización de objetos. Al separar la administración de datos de la lógica de negocio, nos permitirá construir una estructura mayormente escalable en la cual, si se requiere cambiar el método de manejo de persistencia como, por ejemplo, un cambio de ORM, no afectará a la capa de lógica de negocio.

Lenguaje de programación:

- Lenguaje de programación Java JDK 17:



Framework:

- java.swing v17



Base de Datos:

- Base de datos PostgreSQL v15:



Framework de persistencia:

- JPA v5.6
- Hibernate v5.6



Tecnología utilizada:

- Escritorio

Descripción Para la implementación se creó una aplicación de escritorio con java versión 17. La misma incorpora persistencia de objetos gracias a la herramienta de mapeo objeto-relacional (ORM) hibernate y motor de base de datos relacional postgresSQL