# **Iteración 3: FestivAndes – Implementación de aplicación**



Sneider Velandia Gonzalez

Universidad de los Andes, Bogotá, Colombia

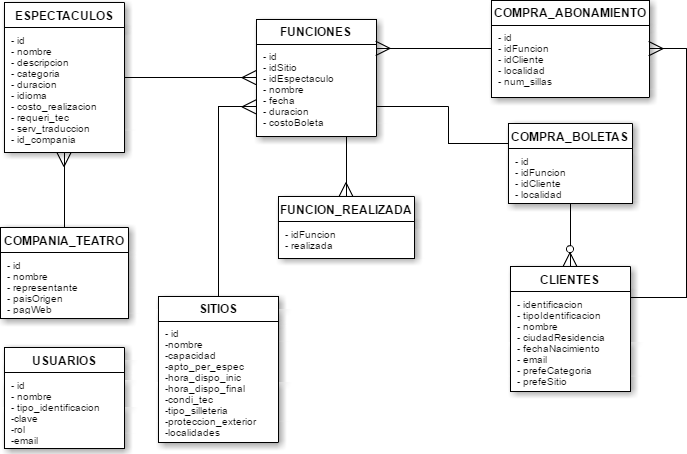
Ingeniería de Sistemas y Computación

Sistemas Transaccionales

Bogotá. DC. 12 de abril de 2017

Tabla de contenido

1. Modelo relacional
2. Diseño de la aplicación
3. **Modelo relacional actualizado**



1. **Diseño de la aplicación**

El impacto que representa la introducción de los nuevos requerimientos y restricciones con respecto al nuevo caso de estudio se ven reflejados en la implementación de pocos atributos y una nueva tabla en la base de datos. La implementación de la nueva lógica permite:

* Registrar compras múltiples de boletas
* Registrar compra de un abonamiento
* Devolver una boleta
* Devolver un abonamiento
* Cancelar una función
* Consultar asistencia al festival de un cliente registrado
* Consultar compañía

La nueva tabla que permite cumplir con los nuevos requerimientos es:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| COMPRA\_ABONAMIENTO | | | | |
| IDBOLETA | IDFUNCION | IDCLIENTE | LOCALIDAD | NUM\_SILLAS |
| PK, ND | FK,NN | FK | NN | NN |

Para garantizar las propiedades ACID de los nuevos requerimientos se utilizó niveles de aislamiento Serialisable en las transacciones que piden registro y devoluciones, pero se usó Read Uncommitted para las transacciones que piden únicamente consulta de información; para hacer uso de estos niveles de aislamiento se desactiva la opción de AutoCommit y se agregan puntos de Rollback durante la transacción por sii se presenta un error de negocio o con el sistema de base de datos, finalmente se hace Commit en las transacciones que son satisfechas y se activa la opción AutoCommit para que no cause problemas en las otras operaciones ya establecidas por el sistema.

Bono:

¿Qué diferencia hay en el manejo transaccional por parte de un contenedor de aplicaciones con respecto al manejo dado por parte del programador de la aplicación?

|  |  |
| --- | --- |
| Contenedor de aplicaciones | Programador de aplicaciones |
| * Mayor escalabilidad. * Accesos seguros con autenticación. * Permite comunicación con otros sistemas de forma eficiente y fácil. * Ofrecen interfaces de fácil uso para el programador. * Prestan soporte en las herramientas usadas como MDB. * Soluciones más costosas pero con mayor respaldo. * Permite accesos concurrentes. * Delega la preocupación de la seguridad y la concurrencia al sistema y el programador puede concentrarse en la solución del negocio. * Implementación de menos líneas de código y disminuye el acoplamiento. | * Libertad de programación. * Propenso a cometer más errores. * El soporte estará basado en el conocimiento del programador. * El código puede ser complicado para otro colega si no está bien estructurado y documentado. * La escalabilidad dependerá del conocimiento del programador. * La solución no será la más rápida ni la más eficiente, ya que el cargo de responsabilidad aumenta y tiene que tener en cuenta todas las posibles variables que se pueden presentar en la implementación de la solución. * Las líneas de código son mayor * Existe más acoplamiento en el código. |