Criterios GRASP aplicados:

* CONTROLLER: La estrategia que utilizamos para diseñar los Controladores es un híbrido entre Fachada y un controlador por CU. Agrupamos varios CU por Controlador tratando de que tengan una Alta Cohesión y de no asignarles demasiadas responsabilidades. Diseñamos 4 controladores que agrupan ciertos CU detallados a continuación:

|  |  |  |  |
| --- | --- | --- | --- |
| ControladorEstadia | ControladorUsuario | ControladorReserva | ControladorHostal |
| FinalizarEstadia  CalificarEstadia  ConsultaEstadia  RegistrarEstadia | AltaUsuario  ConsultaUsuario  ComentarCalificacion | BajaReserva  ConsultaReserva  RealizarReserva | AltaHostal  AltaHabitacion  AsignarEmpleadoHostal  ConsultaHostal  ConsultarTop3 |

Cada controlador es responsable de mantener las colecciones de cada clase asignada.

Por ejemplo, ControladorEstadia es responsable de mantener la colección Estadia.

ControladorReserva es responsable de mantener la colección Reserva.

En particular, ControladorUsuario mantiene una colección de Empleados y una colección de Huespedes por separado.

* EXPERT: En gran medida aplicamos expert ya que es un criterio bastante genérico, así conservamos el encapsulamiento, debido a que los objetos se valen de información propia para hacer lo que se les pide. Esto soporta un Bajo Acoplamiento y una Alta Cohesión.

Por ejemplo, en la operación buscarComentario():DTComentario del CU ConsultaDeEstadia. El ControladorEstadia delega la responsabilidad a Estadia (aplico Expert) ya que este es el experto en información y conoce el comentario de la calificación por estar vinculado con Calificacion. Ahora delego la responsabilidad a Calificacion (aplico Expert) porque este es el experto en información y conoce la respuesta del comentario ya que está vinculado con Comentario. Ahora delego la responsabilidad a Comentario (aplico Expert) ya que este es el experto en la información que necesito para los requerimientos de la operación.

* CREATOR: Para aplicar este criterio asumimos que la clase creadora tiene una responsabilidad muy fuerte para instanciar a la otra clase, ya sea que está agregada, contenida, registra instancias de ella, utiliza objetos en forma exclusiva o es el Experto en crear instancias.

Ej: Diag. de comunicación ComentarCalificacion, operación responderCalificacion.

Es razonable que por Creator se decida que la clase Empleado es responsable de crear instancias de Comentario, ya que la clase Empleado mantiene una asociación con la clase Comentario, un empleado puede hacer varios comentarios. Y también Empleado tiene los datos de inicialización que serán transmitidos a Comentario cuando este objeto sea creado, así que Empleado es un Experto respecto a la creación de Comentario.

* NHCE: En líneas generales aplicamos NHCE asignando responsabilidades y tratando de no ganar visibilidad sobre un objeto “indirecto” para no quedar acoplado a este.

Este criterio se viola cuando un Controlador necesita acceder a otro Controlador, de esa manera ganamos visibilidad sobre un elemento que no conocemos. El motivo de dicha violación es que necesitamos información en particular que la puede brindar otro Controlador.

Ej: Diag. de comunicación RegistrarEstadia, operación obtenerHostales.

El ControladorEstadia le pide información al ControladorHostal.

* BAJO ACOPLAMIENTO: En su mayoría aplicamos este criterio, tratando de que una clase dependa de pocas clases, y que así sea fácil de comprender y reutilizar.

Nos guiamos por las asociaciones que ya estaban en el Modelo de Dominio y no se agregaron nuevas relaciones y dependencias.

Los Controladores son una excepción a este criterio y mantienen un Alto Acoplamiento. Como se refleja en el DCD:

\_ControladorHostales está acoplado con ControladorEstadias.

\_ControladorEstadias está acoplado con ControladorHostales, con ControladorUsuarios y con ControladorReservas.

\_ControladorUsuarios está acoplado con ControladorEstadias.

\_ControladorReservas está acoplado con ControladorHostales y con ControladorUsuarios.

Por ejemplo, ControladorReservas está acoplado con ControladorHostales porque en la operación listarHostales() el ControladorReservas le delega la responsabilidad a ControladorHostales de dicha operación.

* ALTA COHESION: En el diseño de Controladores aplicamos este criterio, buscamos que tengan un número relativamente pequeño de operaciones y que sus funcionalidades estén muy relacionadas.