Contratos e Interfaces

Gestión de Eventos:

Contratos:

- crearEvento: {nombre, descripción, fechaInicio, fechaFin, lugar, costoEstimado, categoría, cupoMáximo, políticasCancelación}
- crearEvento: {idEvento, estado = “Pendiente de Aprobación”}
- notificarCambios: {idEvento, mensaje, canal}
- notificarCambios: {resultado = éxito/error, totalDestinatarios}

Interfaces:

- crearEvento(datosEvento)
- actualizarEvento(idEvento, cambios)
- consultarEvento(idEvento, rolUsuario)
- notificarCambios(idEvento, mensaje)

Referencia:

```
public interface Evento {
   CreateEventResponse createEvent(CreateEventRequest request);
   EventDTO updateEvent(String eventId, EventDTO updates);
   EventDTO getEvent(String eventId, Role requesterRole);
   NotificationResultDTO notifyChange(String eventId, NotificationRequest
notification);
   EventDTO changeEventState(String eventId, EventState newState, String reason);
}
```

Inscripciones

Contratos:

- inscribirCiudadano: {idEvento, nombre, correo, teléfono}
- inscribirCiudadano: {idInscripción, estado = activo/listaEspera, mensajeConfirmación}
- cancelarInscripción: {idInscripción}
- cancelarInscripción: {resultado = éxito/error, cuposLiberados}

Interfaces:

- inscribirCiudadano(idEvento, datosCiudadano)
- cancelarInscripción(idInscripción)
- consultarInscripciones(idEvento)

Referencia:

```
public interface EventoService {
  void registrarEvento(Evento evento);
  Evento obtenerEvento(Long eventoId);
  Reporte generarReporte(Long eventoId);
}
```

Presupuestos

Contratos:

- solicitarPresupuesto: {idEvento, rubros[{concepto, monto}], adjuntos[]}
- solicitarPresupuesto: {idSolicitud, estado = pendiente}
- aprobarPresupuesto: {idSolicitud, montoFinal}
- aprobarPresupuesto: {estado = aprobado, fechaDecisión}
- rechazarPresupuesto: {idSolicitud, motivo}
- rechazarPresupuesto: {estado = rechazado, motivo, fechaDecisión}

Interfaces:

- solicitarPresupuesto(idEvento, rubros, adjuntos)
- aprobarPresupuesto(idSolicitud, montoFinal)
- rechazarPresupuesto(idSolicitud, motivo)
- consultarPresupuesto(idSolicitud)

Referencia:

```
public interface BudgetService {
  BudgetResponseDTO requestBudget(BudgetRequestDTO request);
  BudgetResponseDTO approveBudget(BudgetDecisionDTO decision);
  BudgetResponseDTO rejectBudget(BudgetDecisionDTO decision);
  BudgetResponseDTO getBudgetRequest(String requestId);
}
```

Reportes

Contratos:

- generarReporteParticipación: {rangoFechas, idEvento?, categoría?}
- generarReporteParticipación: {idReporte, inscritos, asistentes}
- generarReporteCostos: {rangoFechas, idEvento?}
- generarReporteCostos: {idReporte, presupuestoEstimado, aprobado, costosReales}
- exportarReporte: {idReporte, formato = PDF/CSV}
- exportarReporte: {archivo, estado = generado}

Interfaces:

- generarReporteParticipación(filtros)
- generarReporteCostos(filtros)
- exportarReporte(idReporte, formato)

Referencia:

```
public interface ReportService {
  ParticipationReportDTO generateParticipationReport(ReportFilter filter);
  CostReportDTO generateCostReport(ReportFilter filter);
  byte[] exportReport(String reportId, String format); // format = "PDF"|"CSV"
}
```

Calendario

Contratos:

- listarEventos: {fecha?, categoría?, lugar?}
- listarEventos: [{idEvento, nombre, fecha, lugar, cupoDisponible, estado}]
- verEvento: {idEvento}
- verEvento: {nombre, descripción, fecha, lugar, costo, cupoDisponible, políticasCancelación}

Interfaces:

- listarEventos(filtros)
- verEvento(idEvento)

Referencia:

```
public interface CalendarService {
  List<CalendarEventDTO> listEvents(CalendarFilter filter);
  CalendarEventDTO getEvent(String eventId);
}

// calendar filters
public class CalendarFilter {
  private LocalDateTime from;
  private LocalDateTime to;
  private String category;
  private String place;
}
```

Notificaciones

Contratos:

- enviarCorreo: {destinatarios[], asunto, mensaje}
- enviarCorreo: {totalEnviados, totalFallidos}
- enviarPush: {destinatarios[], mensaje}
- enviarPush: {totalEnviados, totalFallidos}

Interfaces:

- enviarCorreo(destinatarios, asunto, mensaje)
- enviarPush(destinatarios, mensaje)

Referencia:

```
public interface NotificationService {
  NotificationResultDTO sendEmail(List<String> recipients, String subject, String
message);
  NotificationResultDTO sendPush(List<String> recipients, String message);
  NotificationResultDTO send(NotificationRequest request);
}
```

Usuarios y Roles

Contratos:

- autenticarUsuario: {usuario, contraseña}
- autenticarUsuario: {idUsuario, tokenSesion, rol}
- registrarUsuario: {nombre, correo, rol, credenciales}
- registrarUsuario: {idUsuario, estado = creado}
- consultarRol: {idUsuario}
- consultarRol: {rol}

Interfaces:

- autenticarUsuario(credenciales)
- registrarUsuario(datosUsuario)
- consultarRol(idUsuario)

Referencia:

```
public interface UserService {
  AuthResponse authenticate(String username, String password);
  UserDTO registerUser(UserRegistrationRequest req);
  Role getRole(String userId);
  UserDTO getUser(String userId);
}
```