Contratos e Interfaces

Gestión de Eventos:

Contratos:

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

Interfaces:

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

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:

- inscribirCiudadanoRequest: {idEvento, nombre, correo, teléfono}
- inscribirCiudadanoResponse: {idInscripción, estado = activo/listaEspera, mensajeConfirmación}
- cancelarInscripciónRequest: {idInscripción}
- cancelarInscripciónResponse: {resultado = éxito/error, cuposLiberados}

Interfaces:

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

Referencia:

```
public interface InscripcionService {
  InscripcionDTO inscribirCiudadano(String idEvento, CiudadanoDTO datos);
  ResultadoDTO cancelarInscripcion(String idInscripcion);
  List<InscripcionDTO> consultarInscripciones(String idEvento);
}
```

Presupuestos

Contratos:

- solicitarPresupuestoRequest: {idEvento, rubros[{concepto, monto}], adjuntos[]}
- solicitarPresupuestoResponse: {idSolicitud, estado = pendiente}
- consultarPresupuestoRequest: {idSolicitud}
- consultarPresupuestoResponse: {idSolicitud, estado, montoAprobado, motivoRechazo, fechaDecisión}

Interfaces:

- solicitarPresupuesto(idEvento, rubros, adjuntos)

- consultarPresupuesto(idSolicitud)

Referencia:

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

Reportes

Contratos:

- generarReporteParticipaciónRequest: {rangoFechas, idEvento?, categoría?}
- generarReporteParticipaciónResponse: {idReporte, inscritos, asistentes}
- generarReporteCostosRequest: {rangoFechas, idEvento?}
- generarReporteCostosResponse: {idReporte, presupuestoEstimado, aprobado, costosReales}
- exportarReporteRequest: {idReporte, formato = PDF/CSV}
- exportarReporteRessponse: {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:

- listarEventosRequest: {fecha, categoría, lugar}
- listarEventosResponse: [{idEvento, nombre, fecha, lugar, cupoDisponible, estado}]
- verEventoRequest: {idEvento}
- verEventoResponse: {nombre, descripción, fecha, lugar, costoEstimado, cupoDisponible, políticasCancelación}

Interfaces:

- listarEventos(filtros)
- verEvento(idEvento)

Referencia:

```
public interface CalendarService {
        <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:

- enviarCorreoRequest: {destinatarios[], asunto, mensaje}
- enviarCorreoResponse: {totalEnviados, totalFallidos}
- enviarPushRequest: {destinatarios[], mensaje}
- enviarPushResponse: {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:

- autenticarUsuarioRequest: {usuario, contraseña}
- autenticarUsuarioResponse: {idUsuario, tokenSesion, rol}
- registrarUsuarioRequest: {nombre, correo, rol, contraseña}
- registrarUsuarioResponse: {idUsuario, estado = creado}
- consultarRolRequest: {idUsuario}
- consultarRolResponse: {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);
}
```