



**FACULTAD DE INGENIERÍA  
ESCUELA DE INGENIERÍA INFORMÁTICA**

---

**ARQUITECTURA DE SOFTWARE: ENTREGA 3**

---

**INTEGRANTES:  
ABEL BAULLOZA  
DIEGO CARRILLO  
RODRIGO ORDENES  
BENJAMIN TELLO**

**PROFESOR:  
JUAN RICARDO GIADACH**

**ARQUITECTURA DE SOFTWARE**

**SANTIAGO-CHILE  
03 DE JUNIO 2023**

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Descripción, Organización y Área del proyecto</b>	<b>2</b>
<b>3. Sobre el sistema</b>	<b>2</b>
3.1. Objetivos del sistema . . . . .	2
3.2. Usuarios del sistema . . . . .	3
<b>4. Requerimientos funcionales</b>	<b>4</b>
4.1. Para Usuario . . . . .	4
4.2. Para Administrador . . . . .	4
4.3. Para director . . . . .	5
<b>5. Modelo de datos</b>	<b>6</b>
<b>6. Arquitectura del Sistema</b>	<b>8</b>
6.1. Esquema . . . . .	9
<b>7. Componentes</b>	<b>10</b>
7.1. Descripción componentes clientes y servicios: . . . . .	10
7.1.1. Cliente: . . . . .	10
7.1.2. Servicios: . . . . .	14
<b>8. Evidencias</b>	<b>22</b>
8.1. Registro-BBDD . . . . .	22
8.2. Login-BBDD . . . . .	23
<b>9. Análisis Crítico</b>	<b>23</b>
<b>10. Dificultades</b>	<b>24</b>
<b>11. Conclusión</b>	<b>25</b>
<b>12. Anexos</b>	<b>25</b>

## 1. Introducción

Este proyecto tiene como objetivo desarrollar un software que permita el registro y monitoreo eficiente de información relacionada con jardines infantiles para el personal de la Fuerza Aérea de Chile. El software será diseñado para permitir la gestión centralizada y efectiva de matrículas, ingresos y asistencia del personal y alumnos, así como la creación y gestión de usuarios. Se espera que el software sea fácil de usar y cuente con medidas de seguridad adecuadas para garantizar la protección de la información ingresada por los usuarios. El software resultante contribuirá a la mejora de la calidad de la organización jardines infantiles de la fuerza aérea de Chile.

## 2. Descripción, Organización y Área del proyecto

El proyecto busca desarrollar un software para monitorear de manera centralizada e integra los datos de jardines infantiles para el personal de la Fuerza Aérea de Chile, permitiendo a las educadoras de párvulos y directoras de cada jardín ingresar información como matrículas, ingresos y asistencia del personal y alumnos.

El software también permitirá la creación de usuarios y el registro de nuevos trabajadores. El objetivo es mejorar la eficiencia y organización de los procesos de administración de los jardines infantiles.

Considerando que no todas las instituciones de jardines infantiles organizan sus salas de la misma manera, trabajaremos con el supuesto de que una sala de clases se corresponde con un curso determinado y no cambia anualmente.

## 3. Sobre el sistema

### 3.1. Objetivos del sistema

- Facilitar el registro y monitoreo de información relacionada con jardines infantiles para personal de la Fuerza Aérea de Chile, permitiendo una gestión más eficiente y centralizada de los datos.
- Optimizar los procesos de administración de jardines infantiles, incluyendo la gestión de matrículas, ingresos y asistencia del personal y alumnos, así como la creación y gestión de usuarios.
- Garantizar la protección de la información ingresada por los usuarios en el software, mediante medidas de seguridad adecuadas para prevenir posibles vulnerabilidades.
- Ofrecer estadísticas y análisis en tiempo real de la información de los jardines infantiles, permitiendo a los administradores o directores tomar decisiones informadas y realizar ajustes según sea necesario.

- Integrar el software con otras herramientas de la Fuerza Aérea de Chile, como sistemas de nómina y recursos humanos, para mejorar la eficiencia y la calidad de la gestión de los jardines infantiles.

### **3.2. Usuarios del sistema**

Al estar destinado a servir a la organización de jardines infantiles para personal de la Fuerza Aérea de Chile, los usuarios que se contemplan en este software serán los mismos que podemos ver en la organización de estos jardines, esto es:

- Administrador: Es el usuario que se encarga netamente de la manipulación de datos dentro de un jardín específico.
- Director: Es el usuario con mas capacidades dentro del software, además de crear, eliminar y editar entidades, puede acceder a vistas que entregan información y estadísticas.
- Educadora: Es el usuario encargado de generar un monitoreo dentro del aula de clases, manteniendo un registro de niños. Además, es capaz de realizar matriculas y servicios del rol de director.

## 4. Requerimientos funcionales

### 4.1. Para Usuario

ID	Nombre	Descripción	Entrada	Salida	Prioridad
RF-1	Login de usuario	Usuario se identifica para acceder al sistema	Credenciales (Rut, Contraseña)	Acceso concedido o denegado	Alta
RF-2	Registro de Alumno	Usuario registra un nuevo alumno en el sistema	Datos del alumno	Alumno registrado con éxito o error	Alta
RF-3	Actualización de perfil	Usuario actualiza sus datos de perfil	Datos actualizados del usuario	Perfil actualizado con éxito o error	Media

### 4.2. Para Administrador

ID	Nombre	Descripción	Entrada	Salida	Prioridad
RF-4	Registro de Usuario	Administrador registra a nuevos usuarios en el sistema	Datos del nuevo usuario	Usuario registrado con éxito o error	Alta
RF-5	Eliminación de Usuario	Administrador puede eliminar usuarios del sistema	Rut del usuario	Usuario eliminado con éxito o error	Media
RF-6	Creación de perfil de jardín	Administrador crea nuevos perfiles de jardines infantiles	Datos del nuevo jardín	Jardín creado con éxito o error	Alta
RF-7	Actualización de jardín	Administrador puede modificar perfiles de jardines infantiles	Datos actualizados del jardín	Jardín actualizado con éxito o error	Media
RF-8	Eliminación de jardín	Administrador puede eliminar perfiles de jardines infantiles	Nombre del jardín	Jardín eliminado con éxito o error	Media
RF-9	Estadísticas de jardín	Administrador visualiza datos relevantes de un jardín	Nombre del jardín	Datos del jardín o error	Alto

### 4.3. Para director

ID	Nombre	Descripción	Entrada	Salida	Prioridad
RF-10	Actualización de Alumno	Director actualiza los datos de un alumno	Datos actualizados del alumno	Datos del alumno actualizados o error	Media
RF-11	Eliminación de Alumno	Director puede eliminar alumnos del sistema	Rut del alumno	Alumno eliminado con éxito o error	Media
RF-12	Registro de personal	Director registra a nuevo personal en el sistema	Datos del nuevo personal	Personal registrado con éxito o error	Media
RF-13	Control de Asistencia	Director registra la asistencia de los alumnos	Datos de asistencia (Nivel, Rut del alumno, Fecha, Asistencia)	Asistencia registrada o error	Alto
RF-14	Visualización de Asistencia por jardín	Director visualiza la asistencia por jardín	Nombre del jardín, Rango de fechas	Datos de asistencia o error	Alto
RF-15	Comparación de Asistencia	Director visualiza la asistencia por nivel educativo	Rango de nivel educativo, Rango de fechas	Datos de asistencia o error	Alto
RF-16	Asistencia de personal	Director visualiza la asistencia del personal por escuela o en general	Rut de personal, Fecha	Datos de asistencia o error	Alto

## 5. Modelo de datos

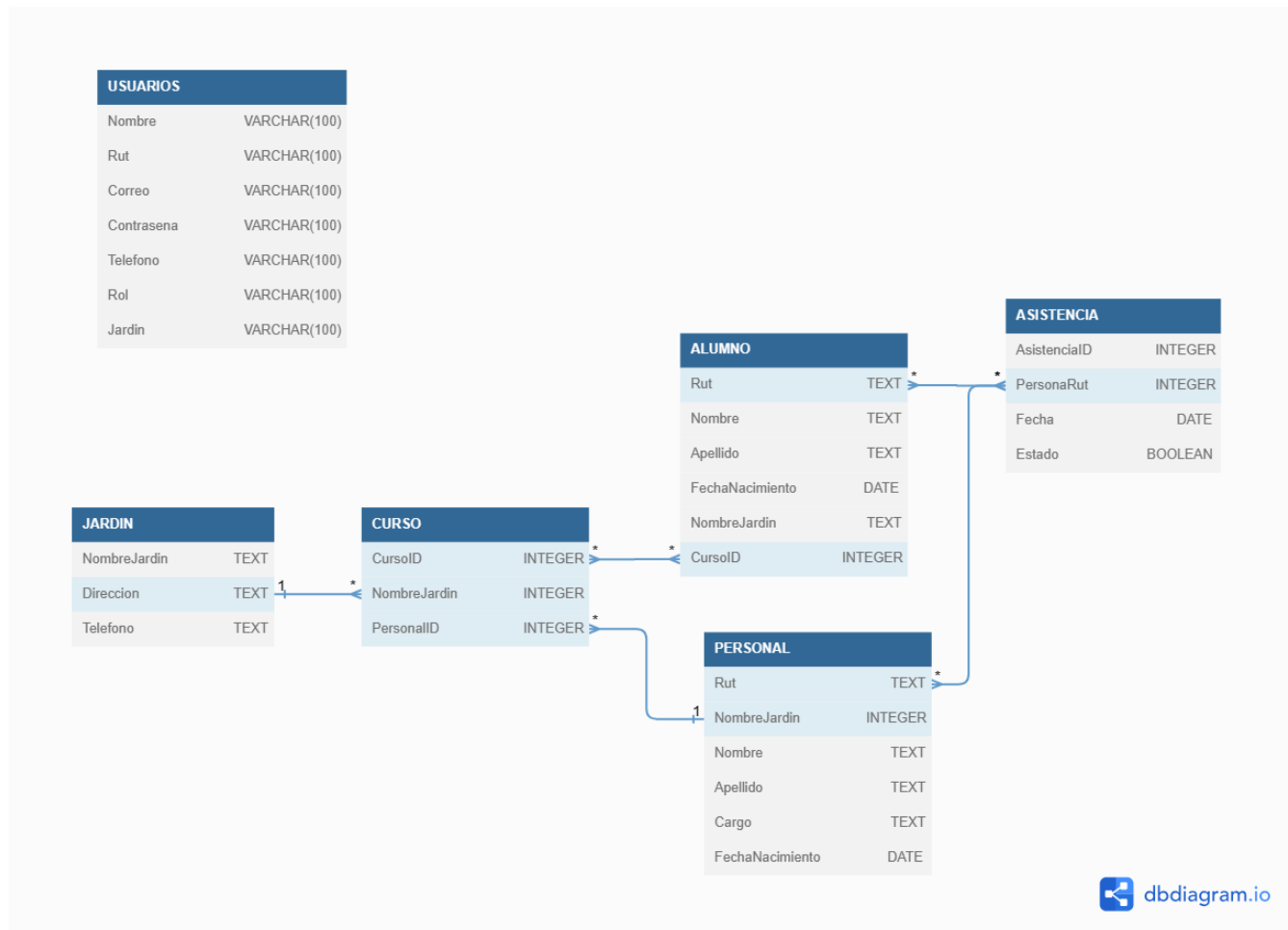


Figura 1: Modelo de la base de datos.

En la figura 1, se puede observar el modelo de la base de datos relacional en *SQLite3*. El cual, tiene 7 tablas:

- **Usuarios:** Tabla que contiene la información de inicio de sesión y otros detalles de los usuarios.
- **Curso:** Tabla que contiene los detalles de los cursos/niveles dentro de su jardín infantil correspondiente.
- **Jardín:** Tabla que contiene información sobre los jardines infantiles registrados en el sistema.
- **Personal:** Tabla que almacena los detalles de los trabajadores dentro de un jardín infantil.
- **Alumno:** Tabla que almacena los detalles de los alumnos en los jardines infantiles.

- **Asistencia:** Esta tabla tendrá la utilidad de registrar la asistencia de los alumnos en sus respectivos jardines infantiles.

La elaboración del proyecto no contempla la manipulación de todas las tablas (*Privilegios* y *Curso*) de la base de datos. Por lo tanto, algunas tablas tendrán información previa con tal de realizar pruebas en el portal.



## 6. Arquitectura del Sistema

La implementación de este sistema estará basada en la arquitectura orientada a servicios (*SOA*), aprovechando los beneficios que este trae consigo como la flexibilidad, mantenibilidad y confiabilidad.

Por ende, la arquitectura de este sistema se compone por un sistema de bus, clientes y servicios:

- **Bus:** El objetivo de este componente es actuar como un intermediario de peticiones entre los clientes y los servicios inicializados.
- **Clientes:** La funcionalidad principal del cliente es permitir la ejecución de peticiones al sistema e interactuar con los servicios.
- **Servicios:** Se encargan de procesar las peticiones recibidas por el bus. Tenemos 3 categorías de servicios; servicios de usuario, director y administrador. Cada tipo de servicio varía en el nivel de permisos de acciones que puede realizar.

## 6.1. Esquema

A continuación, se muestra el esquema de la arquitectura del sistema, donde el bus interactúa entre clientes y servicios desarrollados para la demostración final del proyecto.

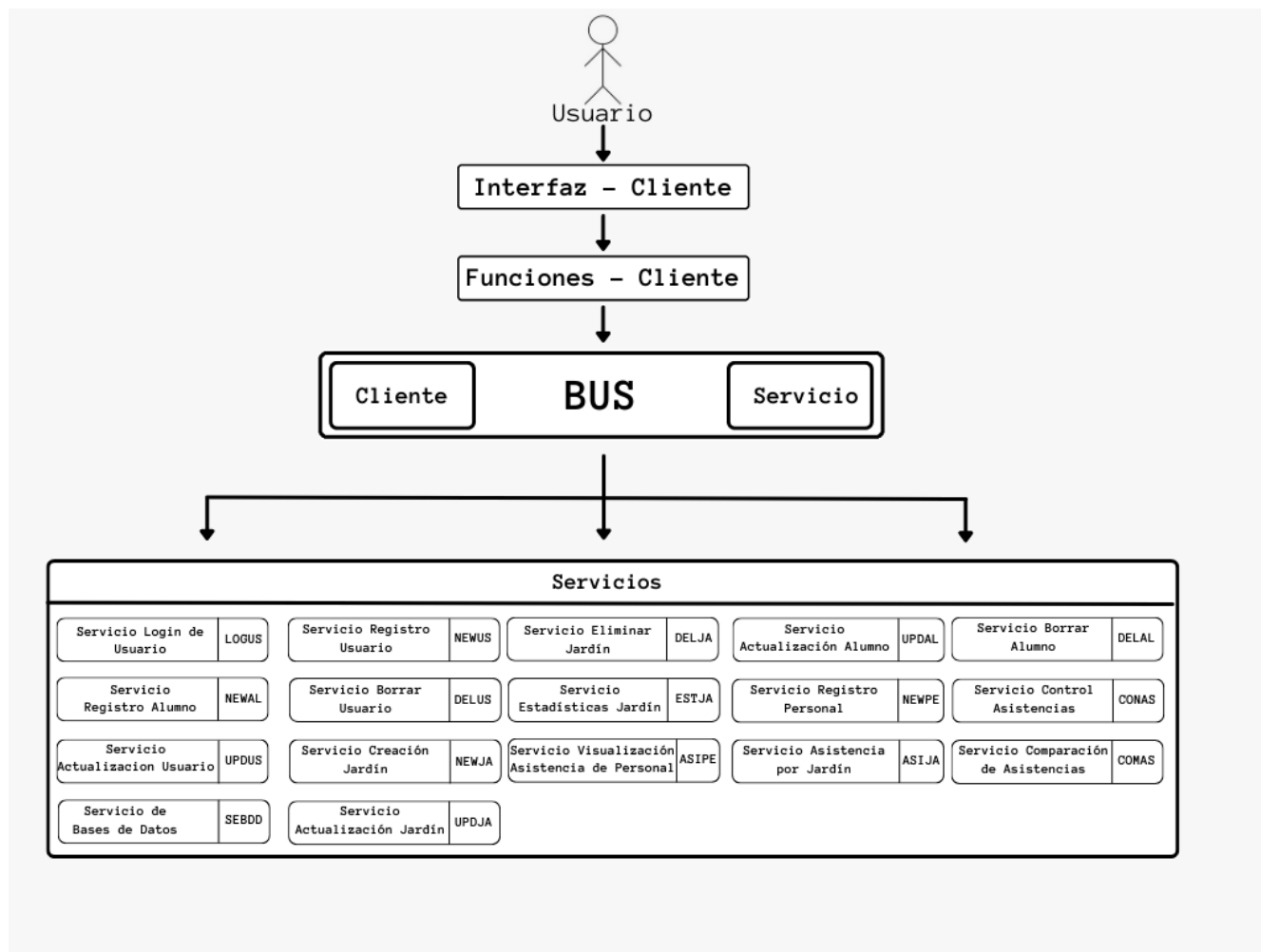


Figura 2: Diagrama de Arquitectura orientada a servicios.

## 7. Componentes

### 7.1. Descripción componentes clientes y servicios:

Cada uno de los componentes, serán descritos por flujos TX - OUT Y TX - IN, siendo TX - OUT la salida desde el componente cliente al bus o el aviso de error correspondiente, y TX - IN la respuesta desde el bus al cliente, en el caso de los servicios este flujo se invierte. Cada servicio internamente se encargará de redirigir la consulta a un único servicio de base de datos donde este trabajará directamente con las consultas (SELECT, INSERT, UPDATE, DELETE) entregando una Respuesta, la cual puede ser éxito/error o un conjunto de datos solicitados.

#### 7.1.1. Cliente:

- - **Login usuario: RF-1** Este cliente permite al usuario iniciar sesión con las credenciales RUT y contraseña. Una vez son ingresados estos datos son enviados al servicio Login siguiendo el siguiente flujo:

TX - OUT — Login — RUT — Contraseña —

TX - IN — Login — Respuesta —

- - **Registro de Estudiante: RF-2** Este cliente le permite al usuario registrar un niño/a dentro del jardín, dada la matricula del alumno. Datos como Rut, Nombre, Apellido, Fecha de nacimiento, Nombre Jardín y Nivel educativo. La respuesta será una operación exitosa o una fallida con la especificación del error; como usuario existente, jardín no existente u otro.

El flujo correspondiente a este servicio es el siguiente:

TX - OUT — RegistroAlumno — Rut — Nombre — Apellido — Fecha de Nacimiento — Nombre Jardin — Nivel educativo —

TX - IN — RegistroAlumno — Respuesta —

- - **Actualización usuario: RF-3** Este cliente permite a un usuario poder actualizar la información de su perfil. Una vez ingresado al proceso se le solicitará al usuario administrador editar o rellenar los datos del usuario. Estos datos serán enviados al servicio de registrar usuario, el cual es el encargado de procesar dicha petición.

El flujo correspondiente a este servicio es el siguiente:

TX - OUT — ActualizarUsuario — Rut — Nombre — Correo — Contraseña — Telefono — Rol — Jardin —

TX - IN — ActualizarUsuario — Respuesta —

- - **Registro usuario: RF-4** Este cliente le permite al administrador registrar un nuevo usuario al sistema, utilizando datos como rol, nombre, apellido, rut, correo, contraseña. Estos datos son enviados al servicio Registro siguiendo el siguiente flujo:

TX - OUT — RegistroUsuario — Rol — Nombre — Apellido — Rut — Correo — Contraseña —

TX - IN — RegistroUsuario — Respuesta —

- - **Borrar usuario: RF-5** Este cliente permite a un usuario con rol de administrador poder borrar el perfil de un usuario. Una vez ingresado al proceso se le solicitará al usuario administrador ingresar el rut del usuario asociado al perfil que se borrará. Estos datos serán enviados al servicio de borrar usuario, el cual es el encargado de procesar dicha petición. Esto puede resultar en un borrado exitoso o fallido si el usuario no existe en la tabla.

TX - OUT — BorrarUsuario — Rut —

TX - IN — BorrarUsuario — Respuesta —

- - **Creación Jardín: RF-6** Este cliente permite a un usuario con rol de administrador poder crear registro de un nuevo jardín. Una vez ingresado al proceso se le solicitará al usuario administrador ingresar los datos del nuevo jardín. Estos datos serán enviados al servicio de crear jardín, el cual es el encargado de procesar dicha petición, si toda la conexión fue bien obtendremos el estado de la petición, esto puede ser una creación exitosa o que el jardín ya existe.

TX - OUT — CrearJardin — Nombre Jardin — Dirección — Teléfono —

TX - IN — CrearJardin — Respuesta —

- - **Actualización Jardín: RF-7** Este cliente permite a un usuario con rol de administrador poder actualizar el perfil de un jardín. Una vez ingresado al proceso se le solicitará al usuario administrador ingresar los datos a actualizar del perfil asociado al jardín. Estos datos serán enviados al servicio de actualizar jardín, el cual es el encargado de procesar dicha petición, si la conexión no tiene problemas, obtendremos el estado de la petición, puede ser una actualización exitosa o que el jardín no existe.

TX - OUT — ActualizarJardin — Nombre actual — Nombre nuevo — Dirección — Teléfono —

TX - IN — ActualizarJardin — Respuesta —

- - **Eliminar Jardín: RF-8** Este cliente permite a un usuario con rol de administrador poder eliminar el perfil de un jardín. Una vez ingresado al proceso se le solicitará al usuario administrador ingresar el Nombre del jardín asociado al perfil que se borrará. Estos datos serán enviados al servicio de borrar jardín, el cual es el encargado de procesar dicha petición, si no existe problema de conexión obtendremos el estado de la consulta, esta puede ser una eliminación exitosa, o que el jardín no existe.

TX - OUT — BorrarJardin — Nombre Jardin —

TX - IN — BorrarJardin — Respuesta —

- - **Estadísticas Jardín: RF-9** Este cliente permite a un usuario con rol de administrador poder visualizar las estadísticas o información asociada a un jardín. Una vez ingresado al proceso se le solicitará al usuario administrador ingresar el Nombre del jardín asociado al perfil del jardín. Estos datos serán enviados al servicio de estadísticas de jardín, el cual es el encargado de procesar dicha petición.

TX - OUT — EstadisticasJardin — Nombre Jardin —

TX - IN — EstadisticasJardin — total\_alumnos, total\_trabajadores —

- - **Actualización alumno: RF-10** Este cliente le permite al usuario con rol de director actualizar los datos de un niño/a en base a un Rut específico y actualizar datos como Nombre, Apellido, Fecha de nacimiento, Nombre de Jardín y/o Nivel educativo. La respuesta será actualización exitosa o fallida. En base al siguiente flujo:

TX - OUT — ActualizarAlumno — Rut — Nombre — Apellido — Fecha de nacimiento — Nombre Jardín —CursoID —

TX - IN — ActualizarAlumno — Respuesta —

- - **Borrar Alumno: RF-11** Este cliente le permite al usuario con rol de director eliminar a un niño/a del sistema en base a su Rut. con el siguiente flujo:

TX - OUT — BorrarAlumno — Rut —

TX - IN — BorrarAlumno — Respuesta —

- - **Registro de Personal: RF-12** Este cliente le permite al usuario con rol de director registrar personal de trabajo para un jardín infantil específico. La respuesta será una operación exitosa o una fallida con la especificación del error; como usuario no existente, jardín no existente u otro.

TX - OUT — RegistroPersonal — Rut — Nombre Jardín — Nombre — Apellido — Cargo — FechaNacimiento —

TX - IN — RegistroPersonal — Respuesta —

- - **Control de Asistencia: RF-13** Este cliente le permite al usuario con rol de director registrar la asistencia de un alumno o personal en una fecha específica.

TX - IN — ControlAsistencia — Rut — Fecha — Estado de Asistencia —

TX - OUT — ControlAsistencia — Respuesta —

- - **Asistencia por Jardín: RF-14** Este cliente permite a un usuario con rol de director poder visualizar la asistencia por jardín. Una vez ingresado al proceso se le solicitará al usuario director ingresar el Nombre del jardín. Estos datos serán enviados al servicio de visualización de asistencia, el cual es el encargado de procesar dicha petición.

TX - OUT — AsistenciaPorJardin — Nombre — Fecha desde — Fecha Hasta —

TX - IN — AsistenciaPorJardin — Conjunto de asistencias tomadas —

- - **Comparación de Asistencia: RF-15** Este cliente permite a un usuario con rol de director poder visualizar la asistencia entre un rango de nivel educativo y rango de fechas independiente del jardín. Una vez ingresado al proceso se le solicitará al usuario director ingresar el rango de nivel educativo y rango de fechas. Estos datos serán enviados al servicio de comparación de asistencia, el cual es el encargado de procesar dicha petición.

TX - OUT — ComparacionAsistencia — Nivel educativo desde — Nivel educativo hasta — Fecha desde — Fecha hasta —

TX - IN — ComparacionAsistencia — Conjunto de asistencias tomadas —

- - **Visualización asistencia de personal: RF-16** Este cliente permite a un usuario con rol de director poder visualizar la asistencia del personal. Una vez ingresado al proceso se le solicitará al usuario director ingresar el rut asociado a un personal y la fecha de asistencia. Estos datos serán enviados al servicio de visualización de asistencia de personal, el cual es el encargado de procesar dicha petición.

TX - OUT — VisualizarAsistenciaPersonal — Rut Personal — Fecha asistencia —

TX - IN — VisualizarAsistenciaPersonal — Asistencia tomada un día específico —

### 7.1.2. Servicios:

Las solicitudes no se realizan directamente desde los servicios especificados , sino que tenemos un servicio extra de base de datos, el cual provee de la ejecución de las *querys*.

#### 1. Usuario:

- - **Login: RF-1** Este servicio tiene por objetivo permitir a un usuario ingresar al sistema. Para ello, este servicio primeramente se conecta al bus con el código “LOGIN”. Una vez conectado, este se encuentra constantemente escuchando una solicitud, cuyo formato corresponde a las credenciales del usuario. Una vez recibida la transacción desde un cliente el servicio realiza una consulta a la base de datos, siendo esta:

```
1 SELECT UsuarioID FROM USUARIO
2 WHERE Rut = '<rut>' AND Contraseña = '<contrasena>';
```

Donde *rut* y *contrasena* son 2 cadenas de texto que representan a las credenciales necesarias que serán buscadas dentro de la tabla “*USUARIOS*” de la base de datos.

Si existe un usuario válido, entonces el servicio retorna un mensaje “Login Exitoso” y “Error en iniciar sesión” en caso contrario.

A continuación se muestra el esquema de transferencia de datos del servicio.

TX - IN — Login — RUT — Contraseña —

TX - OUT — Login — Respuesta —

- - **Registro Alumno: RF-2** Este servicio tiene por objetivo permitir registrar un alumno en el sistema. Para ello, este servicio primeramente se conecta al bus con el código “NEWAL”. Una vez conectado, este se encuentra constantemente escuchando una solicitud, cuyo formato corresponde a los datos del nuevo alumno. Una vez recibida la transacción desde un cliente el servicio realiza una consulta a la base de datos, siendo esta:

```
1 INSERT INTO ALUMNO (CursoID, JardinID, Nombre, Apellido, Rut, FechaNacimiento)
2 VALUES ('<CursoID>', '<JardinID>', '<nombre>', '<apellido>', '<rut>', '<FechaNacimiento>');
```

Donde *Rut*, *Nombre*, *Apellido*, *FechaNacimiento*, *NombreJardin*, *CursoID* corresponden a la información principal de un alumno, el cual se inscribe en jardín infantil y se le asigna el curso de este.

Al agregar de forma exitosa al usuario se muestra el mensaje “Alumno registrado con éxito”. En caso de algún error se muestra “Error en inscribir alumno”.

A continuación se muestra el esquema de transferencia de datos del servicio.

TX - IN — RegistroAlumno — Rut — Nombre — Apellido — Fecha de Nacimiento —  
Nombre Jardin — Nivel educativo —

TX - OUT — RegistroAlumno — Respuesta —

- - **Actualización usuario: RF-3** Este servicio tiene por objetivo permitir al usuario actualizar sus datos. Para ello, este servicio primeramente se conecta al bus con el código “UPDUS”. Una vez conectado, este se encuentra constantemente escuchando una solicitud, cuyo formato corresponde dependiendo a que datos se desean actualizar. Una vez recibida la transacción desde un cliente el servicio realiza una consulta a la base de datos, siendo esta:

```
1 UPDATE USUARIO
2 SET Nombre = '<nombre>', Rut = '<rut>', Email = '<email>', Contraseña = '<contrasena>', Telefono = '<telefono>'
3 WHERE Rut = '<rut>' ;
```

Donde *Rut* será el indicador para actualizar al usuario con sus nuevos datos; *Nombre*, *Correo*, *Contrasena*, *Telefono*, *Rol* y/o *Jardin*.

Al actualizar de forma exitosa al usuario se muestra el mensaje “Datos actualizados correctamente”. En caso de algún error se muestra “Error al actualizar información del alumno”.

A continuación se muestra el esquema de transferencia de datos del servicio.

TX - IN — ActualizarUsuario — Nombre — Rut — Correo — Contraseña — Teléfono — Rol — Jardin —

TX - OUT — ActualizarUsuario — Respuesta —

## 2. Administrador:

- - **Registro usuario: RF-4** Este servicio tiene por objetivo permitir al administrador registrar un usuario en el sistema. Para ello, este servicio primeramente se conecta al bus con el código “REGIS”. Una vez conectado, este se encuentra constantemente escuchando una solicitud, cuyo formato corresponde a los datos del nuevo usuario. Una vez recibida la transacción desde un cliente el servicio realiza una consulta a la base de datos, siendo esta:

```
1 INSERT INTO USUARIO (PrivilegioID, Nombre, Apellido, Rut, Email, Contraseña)
2 VALUES ('<PrivilegioID>', '<nombre>', '<apellido>', '<rut>', '<email>', '<contrasena>');
```

Donde *Nombre*, *Rut*, *Correo*, *Contrasena*, *Telefono*, *Rol* y *Jardin* corresponden a la información para registrar a un usuario y darle permisos de acceso.

Al agregar de forma exitosa al usuario se muestra el mensaje “Usuario registrado con éxito”. En caso de algún error se muestra “Error en registrar usuario”.

A continuación se muestra el esquema de transferencia de datos del servicio.

TX - IN — RegistroUsuario — Nombre — RUT — Correo — Contraseña — Teléfono — Rol — Jardín —

TX - OUT — RegistroUsuario — Respuesta —



- - **Borrar usuario: RF-5** Este servicio tiene por objetivo permitir al administrador borrar un usuario en el sistema. Para ello, este servicio primeramente se conecta al bus con el código “DELUS”. Una vez conectado, este se encuentra constantemente escuchando una solicitud, cuyo formato corresponde al Rut identificador de un usuario existente. Una vez recibida la transacción desde un cliente el servicio realiza una consulta a la base de datos, siendo esta:

```
1 DELETE FROM USUARIO
2 WHERE Rut = '<rut>';
```

Donde *rut* es el indicar para borrar un usuario asociado a este.

Al borrar de forma exitosa al usuario se imprime un mensaje de operación exitosa o fallida, especificando el error.

A continuación se muestra el esquema de transferencia de datos del servicio.

TX - IN — BorrarUsuario — Rut —

TX - OUT — BorrarUsuario — Respuesta —

- - **Creación Jardín: RF-6** Este servicio tiene por objetivo permitir al administrador crear un jardín en el sistema. Para ello, este servicio primeramente se conecta al bus con el código “NEWJA”. Una vez conectado, este se encuentra constantemente escuchando una solicitud, cuyo formato corresponde a los datos del nuevo usuario. Una vez recibida la transacción desde un cliente el servicio realiza una consulta a la base de datos, siendo esta:

```
1 INSERT INTO JARDIN (NombreJardin, Direccion, Telefono)
2 VALUES ('<NombreJardin>', '<Direccion>', '<Telefono>');
```

Donde *NombreJardin*, *Direccion* y *Telefono* son los datos para registrar un nuevo jardín al portal.

Al agregar de forma exitosa al jardín se muestra la siguiente respuesta: “Jardín creado correctamente”. En caso de algún error la respuesta será “Error en agregar jardín”.

A continuación se muestra el esquema de transferencia de datos del servicio.

TX - IN — CrearJardin — Nombre — Dirección — Teléfono —

TX - OUT — CrearJardin — Respuesta —

- - **Actualización Jardín: RF-7** Este servicio tiene por objetivo permitir al administrador actualizar los datos de un jardín en el sistema. Para ello, este servicio primeramente se conecta al bus con el código “UPDJA”. Una vez conectado, este se encuentra constantemente escuchando una solicitud, cuyo formato corresponde a los datos de un jardín ya existente. Una vez recibida la transacción desde un cliente el servicio realiza una consulta a la base de datos, siendo esta:

```
1 UPDATE JARDIN
2 SET NombreJardin = '<NombreJardinFinal>',
3     Direccion = '<Direccion>',
4     Telefono = '<Telefono>'
5 WHERE NombreJardin = '<NombreJardinInicial>;'
```

Donde *Nombre1* será el indicador para actualizar el jardín con sus nuevos datos; *Nombre2*, *Direccion* y *Telefono*.

Al actualizar de forma exitosa al jardín se muestra la siguiente respuesta: “Éxito en actualizar la información del jardín”. En caso de algún error la respuesta será “Error al actualizar información del jardín”.

A continuación se muestra el esquema de transferencia de datos del servicio.

TX - IN — ActualizarJardin — Nombre actual— Dirección — Teléfono — Nombre nuevo —

TX - OUT — ActualizarJardin — Respuesta —

- - **Eliminar Jardín: RF-8** Este servicio tiene por objetivo permitir al administrador eliminar un jardín en el sistema. Para ello, este servicio primeramente se conecta al bus con el código “DELJA”. Una vez conectado, este se encuentra constantemente escuchando una solicitud, cuyo formato corresponde al identificador de un jardín ya existente. Una vez recibida la transacción desde un cliente el servicio realiza una consulta a la base de datos, siendo esta:

```
1 DELETE FROM JARDIN
2 WHERE NombreJardin = '<NombreJardin>;'
```

Donde *NombreJardin* es el indicar para borrar el jardín asociado a este.

Al borrar de forma exitosa al jardín se muestra la siguiente respuesta: “El jardin *NombreJardin* se ha eliminado correctamente”. En caso de algún error la respuesta será: “Error en borrar jardín”.

A continuación se muestra el esquema de transferencia de datos del servicio.

TX - IN — BorrarJardin — Nombre —

TX - OUT — BorrarJardin — Respuesta —

- - **Estadísticas Jardín: RF-9** Este servicio tiene por objetivo permitir al administrador revisar las estadísticas (cantidad de alumno, cantidad de personal) de un jardín en el sistema. Para ello, este servicio primeramente se conecta al bus con el código “ESTJA”. Una vez conectado, este se encuentra constantemente escuchando una solicitud, cuyo formato corresponde al identificador de un jardín ya existente. Una vez recibida la transacción desde un cliente el servicio realiza una consulta a la base de datos, siendo esta:

```
1 SELECT COUNT(*) FROM ALUMNO WHERE NombreJardin = 'NombreJardin';
2 SELECT COUNT(*) FROM PERSONAL WHERE NombreJardin = 'NombreJardin';
```

Donde *NombreJardin* es el indicar para extraer información de una jardín específico. Para este servicio solo entregaremos la cuenta total de alumnos y personal pertenecientes a un jardín infantil.

A continuación se muestra el esquema de transferencia de datos del servicio.

TX - IN — EstadisticasJardin — NombreJardin —

TX - OUT — EstadisticasJardin — [total\_alumnos, total\_personal] —

### 3. Director:

- - **Actualización de alumno: RF-10** Este servicio tiene por objetivo permitir al director actualizar los datos de un alumno. Para ello, este servicio primeramente se conecta al bus con el código “UPDAL”. Una vez conectado, este se encuentra constantemente escuchando una solicitud, cuyo formato corresponde a los datos de un alumno ya existente. Una vez recibida la transacción desde un cliente el servicio realiza una consulta a la base de datos, siendo esta:

```
1 UPDATE ALUMNO
2   SET Nombre = '<Nombre>',
3       Apellido = '<Apellido>',
4       FechaNacimiento = '<FechaNacimiento>',
5       NombreJardin = '<NombreJardin>',
6       CursoID = '<CursoID>'
7 WHERE Rut = '<Rut>';
```

Donde *Rut* será el indicador para actualizar la información del alumno con sus nuevos datos; *Nombre*, *CursoID*, *Apellido*, *FechaNacimiento* y *NombreJardin*.

Al actualizar de forma exitosa la información del alumno se muestra el mensaje “Alumno actualizado con éxito”. En caso de algún error se muestra “Error al actualizar información del alumno”, especificando el tipo de error que puede ser si el alumno o el jardín es inexistente, entre otros.

A continuación se muestra el esquema de transferencia de datos del servicio.

TX - IN — ActualizarAlumno — Rut — Nombre — Apellido — FechaNacimiento — NombreJardin — CursoID —

TX - OUT — ActualizarAlumno — Respuesta —

- - **Borrar Alumno: RF-11** Este servicio tiene por objetivo permitir al director eliminar a un alumno del sistema. Para ello, este servicio primeramente se conecta al bus con el código “DELAL”. Una vez conectado, este se encuentra constantemente escuchando una solicitud, cuyo formato corresponde al id de un alumno ya existente. Una vez recibida la transacción desde un cliente el servicio realiza una consulta a la base de datos, siendo esta:

```
1 DELETE FROM ALUMNO
2 WHERE Rut = '<rut>';
```

Donde *rut* es el indicativo para borrar al alumno asociado a este.

Al borrar de forma exitosa al alumno se muestra el mensaje “Alumno borrado con éxito”. En caso de algún error se muestra “Error en borrar alumno”.

A continuación se muestra el esquema de transferencia de datos del servicio.

TX - IN — BorrarAlumno — Rut —

TX - OUT — BorrarAlumno — Respuesta —

- - **Registro de Personal: RF-12** Este servicio tiene por objetivo permitir al director para registrar trabajadores de un jardín infantil correspondiente. El servicio se conecta al bus con el código “NEWPE”. Una vez conectado, queda constantemente escuchando una solicitud, cuyo formato corresponde a los datos del nuevo personal. Una vez recibida la transacción desde un cliente el servicio realiza una consulta a la base de datos, siendo esta:

```
1 INSERT INTO PERSONAL (Rut, NombreJardin, Nombre, Apellido, Cargo, FechaNacimiento)
2 VALUES ('<Rut>', '<NombreJardin>', '<Nombre>', '<Apellido>', '<Cargo>', '<FechaNacimiento>');
```

Donde *NombreJardin*, *Nombre*, *Apellido*, *Rut* *Cargo* y *FechaNacimiento* corresponden a la información para registrar a un trabajador de una institución específica.

Al agregar de forma exitosa al trabajador se muestra el mensaje “Personal registrado correctamente”. En caso de algún error se muestra “Error en registrar trabajador”.

A continuación se muestra el esquema de transferencia de datos del servicio.

TX - IN — RegistrarPersonal — NombreJardin — Nombre — Apellido — Rut — Cargo — FechaNacimiento

TX - OUT — RegistrarPersonal — Respuesta —

- - **Control de Asistencia: RF-13** Este servicio tiene por objetivo permitir al director ingresar la asistencia correspondiente a un alumno o personal en una fecha específica. El servicio se conecta al bus con el código “CONAS”. Una vez conectado, queda constantemente escuchando una solicitud, cuyo formato corresponde a los datos de asistencia de un alumno existente. Una vez recibida la transacción desde un cliente el servicio realiza una consulta a la base de datos, siendo esta:

```
1 SELECT * FROM ASISTENCIA
2 WHERE PersonaRut = '<PersonaRut>' AND Fecha BETWEEN '<FechaInicio>' AND '<FechaFin>';
```

Donde *PersonaRut*, *Fecha* y *Estado* serán los indicativos para filtrar la asistencia de un alumno o trabajador en un rango de fechas específicas. La respuesta será operación exitosa o fallida, dependiendo si el rut se encuentra registrado en las tablas *Alumno* o *Personal*.

TX - IN — ControlAsistencia — PesonaRut — Fecha — Estado —

TX - OUT — ControlAsistencia — Conjunto de asistencias tomadas —

- - **Asistencia por Jardín: RF-14** Este servicio tiene por objetivo permitir al director ver la asistencia correspondiente a cada jardín. El servicio se conecta al bus con el código “ASIIJA”. Una vez conectado, queda constantemente escuchando una solicitud, cuyo formato corresponde al nombre de un jardín existente. Una vez recibida la transacción desde un cliente el servicio realiza una consulta a la base de datos, siendo esta:

```
1 SELECT ALUMNO.NombreJardin, ALUMNO.CursoID, ALUMNO.Rut, ASISTENCIA.Fecha, ASISTENCIA.Estado
2 FROM ASISTENCIA JOIN ALUMNO ON ASISTENCIA.PersonaRut = ALUMNO.Rut
3 WHERE ALUMNO.NombreJardin = '<NombreJardin>' AND ASISTENCIA.Fecha BETWEEN '<FechaInicial>' AND '<FechaFinal>';
```

Donde *NombreJardin*, *fecha\_inicio* y *fecha\_fin* serán los indicativos para filtrar la asistencia de todos los alumnos y/o personal pertenecientes a un jardín infantil específico dentro de un rango de fechas. En caso de algún error, la respuesta será 'Jardín no existe' o 'No existen asistencias registradas en el rango de fechas'.

TX - IN — AsistenciaPorJardin — NombreJardin — Fecha\_inicio — Fecha\_fin —

TX - OUT — AsistenciaPorJardin — [Nombre\_jardin, CursoID, Alumno\_rut, Fecha\_asistencia, Estado\_asistencia] —

- - **Comparación de Asistencia: RF-15** Este servicio tiene por objetivo permitir al director ver la asistencia correspondiente a un nivel educativo y rango de fechas de manera general independiente del jardín. El servicio se conecta al bus con el código “COMAS”. Una vez conectado, queda constantemente escuchando una solicitud, cuyo formato corresponde al curso/nivel educativo y rango de fechas. Una vez recibida la transacción desde un cliente el servicio realiza una consulta a la base de datos, siendo esta:

```
1 SELECT CURSO.CursoID, ALUMNO.Nombre, ALUMNO.Apellido, ASISTENCIA.Fecha, ASISTENCIA.Estado FROM ASISTENCIA
2 JOIN ALUMNO ON ASISTENCIA.PersonaRut = ALUMNO.Rut
3 JOIN CURSO ON ALUMNO.CursoID = CURSO.CursoID
4 WHERE CURSO.CursoID IN ('<CursoID1>','<CursoID2>') AND ASISTENCIA.Fecha BETWEEN '<FechaInicial>' AND '<FechaFinal>';
```

Donde *CursoID1*, *CursoID2*, *fecha\_inicio* y *fecha\_fin* serán los indicativos para filtrar la asistencia total entre 2 cursos (pueden ser de distinto jardín) en un rango de fechas específicas. En caso de algún error, la respuesta será 'No hay asistencias registradas a los cursos' o 'No existen asistencias registradas a los cursos en el rango de fechas'.

TX - IN — ComparacionAsistencia — CursoID1 — CursoID2 — Fecha\_inicio — Fecha\_fin —

TX - OUT — ComparacionAsistencia — [CursoID, Nombre\_alumno, Apellido\_alumno, Fecha\_asistencia, Estado\_asistencia] —

- - **Visualización asistencia de personal: RF-16** Este servicio tiene por objetivo permitir al director ver la asistencia del personal correspondiente a cada jardín. El servicio se conecta al bus con el código “ASIPE”. Una vez conectado, queda constantemente escuchando una solicitud, cuyo formato corresponde al nombre del jardín. Una vez recibida la transacción desde un cliente el servicio realiza una consulta a la base de datos, siendo esta:

```
1 SELECT ASISTENCIA.PersonaRut, ASISTENCIA.Fecha, ASISTENCIA.Estado FROM ASISTENCIA JOIN PERSONAL ON ASISTENCIA.PersonaRut = PERSONAL.Rut
2 WHERE PERSONAL.Rut = '<Rut>' AND ASISTENCIA.Fecha = '<Fecha>';
```

Donde *PersonalID*, *fecha* serán los indicativos para filtrar la asistencia de un trabajador en un día específico. En caso de algún error, la respuesta sera 'Rut de personal no registrado' o 'No existe asistencia del personal correspondiente a la fecha'.

TX - IN — VisualizarAsistenciaPersonal — PersonaRut — Fecha\_asistencia —

TX - OUT — VisualizarAsistenciaPersonal — PersonaRut, Fecha\_asistencia, Estado\_asistencia —

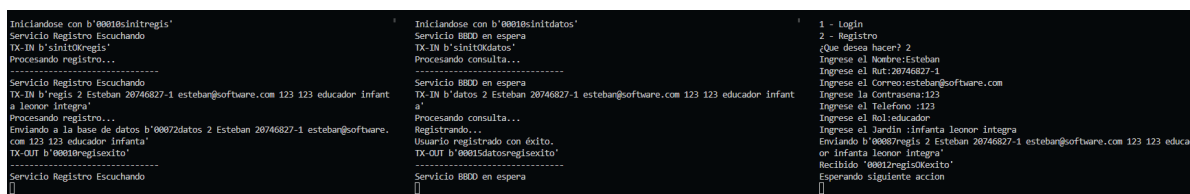
## 8. Evidencias

Para poner a prueba el software, implementamos en un inicio los servicios de login y registro de usuario, además de base de datos, junto a un cliente que invocará estos servicios de manera directa e indirecta.

El flujo de información, como ya se especificó anteriormente será Cliente — Servicio Login/Registro — Servicio Base de datos — Servicio Login/Registro — Cliente. Para poner a prueba estos tres al mismo tiempo registraremos un usuario nuevo y luego iniciaremos sesión con sus credenciales específicas.

Según lo conversado en la presentación con el profesor, el resto de evidencias se encuentra validado en un video demostrativo, el cual se encuentra en la sección de Anexos 12.

### 8.1. Registro-BBDD



The image displays three terminal windows illustrating the communication flow for user registration:

- Left Terminal (Client):** Shows the client initiating a registration request. It sends a message to the registration service and receives a response indicating success.
- Middle Terminal (Registro Service):** Shows the registration service receiving the request from the client, processing it, and then sending a message to the database service to store the user data.
- Right Terminal (BBDD Service):** Shows the database service receiving the request from the registration service, processing the data, and returning a success message to the registration service.

Como podemos observar, lo primero que tenemos es que tanto el servicio de registro como el de base de datos envían un mensaje al bus para identificarse como servicios, posteriormente quedan a la escucha de nuevas consultas.

En el cliente lo que haremos es seleccionar la opción registro e ingresaremos toda la información necesaria. Para este ejemplo nos interesa conocer el Rut y la Contraseña para el posterior inicio de sesión, siendo estos 20746827-1 y 123 respectivamente.

Como podemos ver toda la data es recibida por el servicio registro el cual luego de procesarla, la envía en un formato específico al bus, específicamente al servicio de base de datos, donde es recibida, comienza a procesar la consulta, se registra dentro de la base de datos y como no existió ningún tipo de problema retorna un mensaje de éxito, además de indicarle al bus que fue procesada correctamente la consulta. Este mensaje retorna al Registro el cual entiende que todo fue correctamente y este finalmente puede retornar un mensaje exitoso al cliente finalizando el proceso de registro.

## 8.2. Login-BBDD

En el caso del login el flujo de información es el mismo. Comenzando por el cliente y la base de datos identificándose al bus como servicios login y datos.

```

benjamin.tello@dcencia:~/software$ python3 login.py
Iniciandose con b'0001login'
Servicio Login escuchando
TX-IN b'sinitOlogin'
Procesando inicio de sesion...
-----
Servicio Login escuchando
TX-IN b'login 1 20746827-1 123'
Procesando inicio de sesion...
Enviando a la base de datos b'0002datos 1 20746827-1 123'
TX-OUT b'0001loginexito'
-----
Servicio Login escuchando
[]

benjamin.tello@dcencia:~/software$ python3 connection.py
Iniciandose con b'0001initdatos'
Servicio BBDD en espera
TX-IN b'sinitOdatos'
Procesando consulta...
-----
Servicio BBDD en espera
TX-IN b'datos 1 20746827-1 123'
Procesando consulta...
Ingresando...
Login Exitoso.
TX-OUT b'0001datosloginexito'
-----
Servicio BBDD en espera
[]

benjamin.tello@dcencia:~/software$ python3 client.py
connecting to localhost port 5000
Menu:
1 - Login
2 - Registro
¿Que desea hacer? 1
Ingreso el Rut:20746827-1
Ingreso la Contraseña:123
Enviando b'0002login 1 20746827-1 123'
Recibido '0001loginOexito'
Esperando siguiente accion
[]

```

En el cliente indicamos que queremos iniciar sesión con las credenciales mencionada anteriormente. Estas son enviadas al servicio login, el cual las procesa en un formato para redirigirlas a la base de datos. Como las credenciales son validas no hubo ningún problema al buscar a la persona en la base de datos y este servicio retorna un mensaje de éxito. Ya en el servicio login, al recibir que todo fue exitoso en la base de datos, retorna al cliente su propia confirmación de que todo el proceso ha ido correctamente.

## 9. Análisis Crítico

Los servicios propuestos para el desarrollo del sistema resultaron ser exitosos, cumpliendo con una correcta ejecución de las tareas desarrolladas como registros, obtención, actualización y borrado de datos. Es decir, los servicios se encuentran bien logrados si se analizan individualmente, sin embargo, no es un sistema que pueda considerarse completo ya que faltan numerosos servicios clave que componen el funcionamiento total, por ejemplo, validaciones, servicios y una base de datos coherente con el contexto de la problemática.

Para los roles, es importante mantener una coherencia de privilegios. Por lo tanto, sería una buena practica añadir una tabla de privilegios que represente el nivel de permisos que tenga un usuario, pudiendo darle un conjunto de privilegios para realizar distintas acciones en el sistema. Por ejemplo, que el administrador tenga permisos para ejecutar todos los servicios, a diferencia de un usuario normal (profesores) que deben manipular datos correspondientes solamente al jardín que tienen asociados.

Para las validaciones, un aspecto destacable es la falta de filtros en las entradas de usuarios que eviten ingresar datos incoherentes o maliciosos; agregando filtros que garanticen el ingreso de datos de un tipo correspondiente, evitando vulnerabilidades, también saneamiento de inyecciones SQL y XSS, entre otros.

Para los servicios, cabe resaltar la falta de funcionalidades para que el sistema esté completo; como la creación, actualización y borrado de cursos, entre otros.

Finalmente, las tablas de la base de datos podrían corregirse puesto que algunas columnas como números telefónicos soportan hasta 100 caracteres, siendo que bajo el contexto Chileno debería ser hasta 8 o 12 en caso de considerar la '+569'.



## 10. Dificultades

En lo que respecta con el desarrollo del proyecto, hubo una serie de dificultades que retrasaron y complicaron el trabajo de esta propuesta.

Primero, está el entorno de desarrollo, en donde todos los integrantes del grupo debieron diseñar el código del proyecto. Aquí los principales problemas estuvieron en la conexión del bus e implementación de servicios, ya que, los estudiantes no tenían conocimiento en el desarrollo para este tipo de arquitecturas. Esto implicó de una gran cantidad de horas de investigación e implementación, junto con las pruebas que se debieron realizar.

Segundo, para poder tener éxito en la implementación del proyecto y que fuera funcional, se debía de realizar una conexión al servidor de la universidad, en donde se accede al equipo que poseía el BUS de servicios respectivo. Esto generó nuevos inconvenientes, debido a la poca información que existía para poder acceder a esto.

Tercero, el bus de servicio que estaba siendo utilizado presentó una gran cantidad de problemas, los cuales fueron críticos y dejaban nuestro avance inutilizable. Esto sucedió una cantidad incontable de veces, es más, esto impidió una ejecución de la presentación del proyecto como debió ser realizada. Junto con esto, se suma el hecho de que en una arquitectura SOA, los servicios operan de forma independiente, coordinarlos para que trabajen juntos, las pruebas y depuraciones que se debieron realizar es un desafío al momento de ir desarrollando el sistema en un comienzo.

Cuarto, si bien la arquitectura orientada a servicios nos permite distribuir las actividades a realizar, sin embargo, existe una gran falencia en los servicios. Esta actividad está destinada a ser realizada en conjunto con dos secciones correspondientes al ramo, en donde existen aproximadamente 10 grupos y cada grupo posee 15 servicios o más. Estos durante el proceso de desarrollo generaron colisiones constantemente, impidiendo el ingreso de datos o bien, el choque de datos con otras personas que utilizaban los servicios.

## 11. Conclusión

El componente “bus” resultó ser el componente principal al implementar un sistema con arquitectura orientada a servicios *SOA*, permitiendo la comunicación entre el cliente con los distintos servicios. Para una compatibilidad del sistema con el bus, se han establecido los componentes *Cliente* y *Servicio* quienes facilitan la transferencia de datos y el establecimiento de conexiones con servicios.

Cómo el bus ya está desarrollado, se encuentra fuera de nuestro control total, lo cual resultó en un gran reto entender la lógica de este. Actuando como un obstáculo que dificultó el progreso y pruebas del sistema. Sin embargo, nos beneficia para entender los beneficios que trae esta arquitectura; como:

- Modularidad: Separando todos los servicios en distintos módulos de código.
- Adaptabilidad: Permitiendo realizar cambios rápidamente en los servicios de forma individual. Pudiendo ser reconfigurados en caso de errores o adaptarse a las demandas y oportunidades actuales.
- Interoperabilidad: Permite comunicarse fácilmente con otros sistemas gracias a sus estándares de comunicación. Esto no fue comprobado de forma técnica pero si entendimos la teoría por la que el bus permite comunicarse con distintos sistemas.
- Mantenibilidad: Como está hecho de forma modular, será fácil en un futuro que adquiera nuevas funcionalidades.

## 12. Anexos

- Repositorio GitHub: <https://github.com/Des-Tello/ArquiSW/tree/final>
- Video Demostrativo: <https://youtu.be/xKWEt3m98R0>