PSP - Examen 2a Evaluación

Reserva de salas

Queremos hacer una aplicación C/S que gestione la **reserva de salas** en un centro de formación (aulas de informática, salas de reuniones, laboratorios, etc.). La aplicación gestionará:

- Un listado de **salas disponibles** (código, nombre, aforo máximo).
- Un listado de **reservas activas**, asociadas a un usuario y a una sala.
- Un listado de **usuarios conectados**.

El sistema debe permitir:

- Consultar el listado de salas y reservas.
- Crear nuevas reservas asociadas al usuario que lanza la petición.
- Cancelar reservas existentes.
- Desconectarse de la aplicación.

> **Aplicación Multihilo**
>
> El servidor será **multihilo**, de forma que pueda atender a múltiples clientes concurrentemente, compartiendo y sincronizando las estructuras de datos necesarias.

Instrucciones Generales

Elemento	Descripción
Proyecto servidor	Completar la clase `ReservasServidor` y la clase `ReservasWorker`
Proyecto cliente	Completar la clase `ReservasCliente` con la lógica del protocolo.
Estructuras datos	Salas y reservas.
Concurrencia	El servidor debe ser capaz de atender a varios clientes en paralelo, compartiendo la información de las estructuras de datos.
Excepciones	Capturar y gestionar correctamente las excepciones de I/O.

> **Datos de prueba**
>
> Tenéis un conjunto de **datos precargados** para realizar las pruebas.

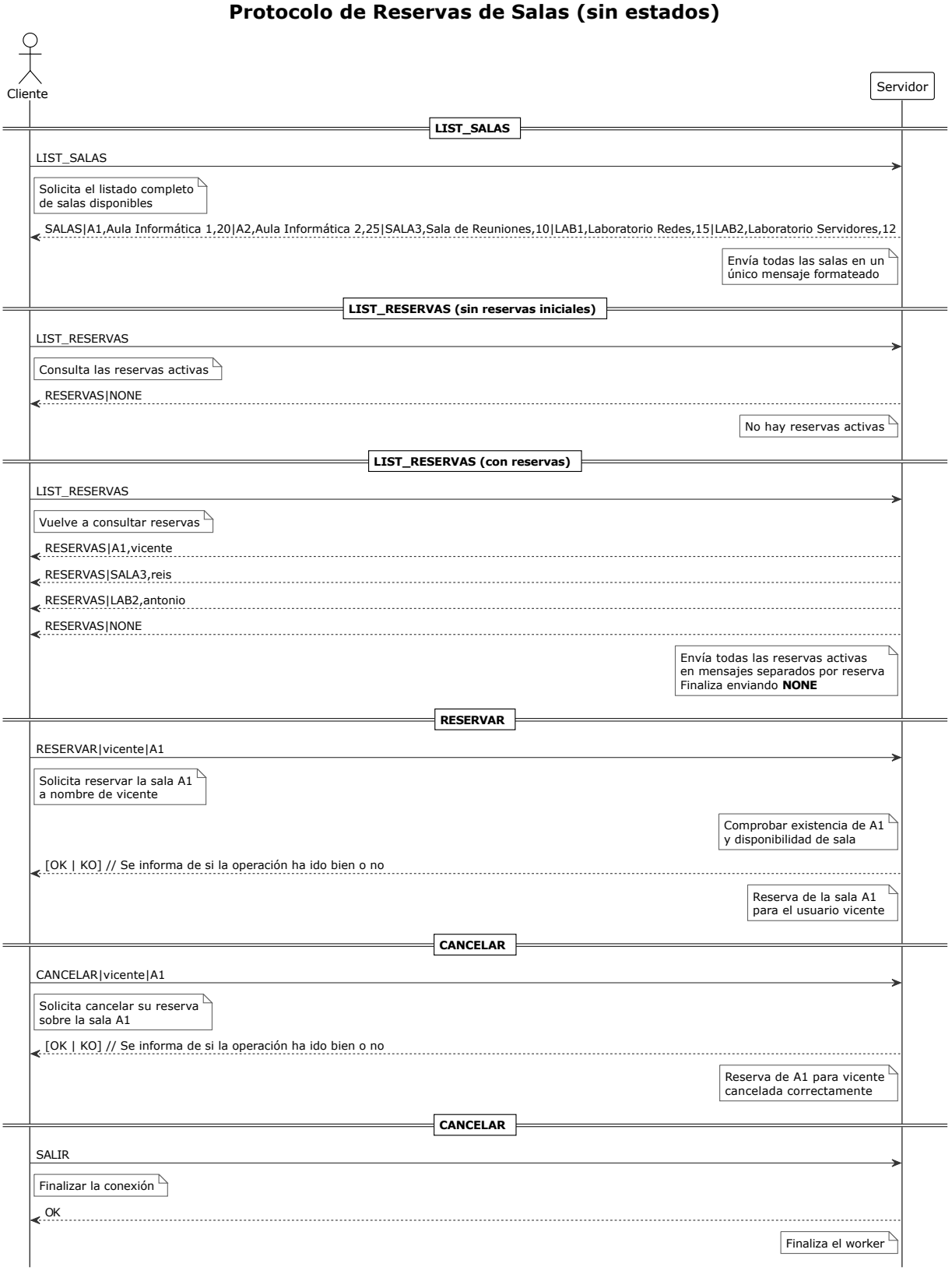
Código	Nombre	Aforo
A1	Aula Informática 1	20
A2	Aula Informática 2	25
SALA3	Sala de Reuniones	10

Código	Nombre	Aforo
LAB1	Laboratorio Redes	15
LAB2	Laboratorio Servidores	12

Inicialmente, **todas las salas están libres** (sin reservas).

PARTE 1: PROTOCOLO SIN ESTADOS (70%)

En esta primera parte se propone un **protocolo sin estados**, es decir, todos los comandos se pueden procesar en cualquier momento, en cualquier orden y de forma independiente.



Todos los mensajes se pueden procesar en cualquier momento (sin estados).

Descripción del Protocolo (sin estados)

Se utilizarán **4 comandos** mas el comando **SALIR**. Todos los mensajes se intercambian en texto plano, usando como separador el carácter `|` cuando haya varios campos en un mismo mensaje.

1. Comando LIST_SALAS

Devuelve el listado de salas existentes en el sistema.

Cliente

Paso	Acción
1	Envía un mensaje: `LIST_SALAS`.
2	Espera una única respuesta con el listado de salas.
3	Parsea la respuesta y la muestra formateada al usuario.

Servidor

Paso	Acción
1	Recibe `LIST_SALAS`.
2	Recorre la estructura de salas
3	Devuelve un único mensaje: `SALAS|codigo1,nombre1,aforo1|codigo2,nombre2,aforo2|...|codigoN,nombreN,aforoN`.

Salida en el cliente

```
Introduce comando: LIST_SALAS
** Salas disponibles **
A1 - Aula Informática 1 (Aforo: 20)
A2 - Aula Informática 2 (Aforo: 25)
SALA3 - Sala de Reuniones (Aforo: 10)
LAB1 - Laboratorio Redes (Aforo: 15)
LAB2 - Laboratorio Servidores (Aforo: 12)
```

Salida en el servidor

```
--> Cliente: LIST_SALAS
Se envía el listado de 5 salas al usuario.
```

2. Comando LIST_RESERVAS

Devuelve el listado de **todas** las reservas activas del sistema.

Cliente

Paso	Acción
1	Envía: `LIST_RESERVAS`.
2	Espera un mensaje por cada una de las reservas, hasta que recibe el mensaje `RESERVAS|NONE`
3	Parsea y muestra la información al usuario.

Servidor

Paso	Acción
1	Recibe `LIST_RESERVAS`.
2	Accede a la estructura de reservas.
3	Recorre la lista de reservas activas.
4	Devuelve: `RESERVAS|sala1,usuario1`
	`RESERVAS|sala2,usuario2`
	`RESERVAS|...`
	`RESERVAS|salaK,usuarioK`
	Si no hay reservas o no hay más reservas: `RESERVAS|NONE`.

Salida en el cliente

Con reservas:

```
Introduce comando: LIST_RESERVAS
** Reservas activas **
A1 - vicente
SALA3 - juan
LAB1 - ana
```

Sin reservas:

```
Introduce comando: LIST_RESERVAS
** Reservas activas **
(No hay reservas activas)
```

Salida en el servidor

```
--> Cliente: LIST_RESERVAS
Se envía el listado de reservas: A1 (vicente), SALA3 (juan), LAB1 (ana).
```

3. Comando RESERVAR

Permite reservar una sala, asociándola al usuario que lanza la petición.

Cliente

Paso	Acción
1	Solicita por teclado el código de sala a reservar.
2	Solicita por teclado el usuario que quiere realizar la reserva
3	Envía: `RESERVAR|nombreUsuario|codigoSala`
4	Espera respuesta: `OK` o `KO`.

Servidor

Paso	Acción
1	Recibe `RESERVAR|nombreUsuario|codigoSala`.
2	Comprueba que la sala existe.
3	Comprueba si la sala está disponible (no tiene reserva activa).
4	Si está disponible, crea una entrada asociando sala y usuario en las reservas y responde `OK`.
5	Si la sala no existe o está ocupada, responde `KO`.

Salida en el cliente

Éxito:

```
Introduce comando: RESERVAR
Indica el código de la sala que quieres reservar: A1
Indica el nombre del usuario que quiere realizar la reserva: pedro
Reserva realizada correctamente.
```

Error:

```
Introduce comando: RESERVAR
Indica el código de la sala que quieres reservar: A1
Indica el nombre del usuario que quiere realizar la reserva: pedro
No se ha podido realizar la reserva (sala no existe u ocupada).
```

Salida en el servidor

Éxito:

```
--> Cliente: RESERVAR|pedro|A1
La sala A1 ha sido reservada por el usuario pedro.
```

Error:

```
--> Cliente: RESERVAR|pedro|A1
No se puede reservar la sala A1 para pedro (ocupada o inexistente).
```

4. Comando CANCELAR

Permite cancelar una reserva existente asociada a un usuario.

Cliente

Paso	Acción
1	Solicita por teclado el código de la sala.
2	Solicita por teclado el usuario que quiere realizar la reserva.
3	Envía: `CANCELAR|nombreUsuario|codigoSala`.
3	Espera respuesta: `OK` o `KO`.

Servidor

Paso	Acción
1	Recibe `CANCELAR|nombreUsuario|codigoSala`.
2	Busca en la estructura de reservas una reserva de esa sala asociada a ese usuario.
3	Si existe, elimina la reserva de las reservas y responde `OK`.
4	Si no existe, responde `KO`.

Salida en el cliente

Éxito:

```
Introduce comando: CANCELAR
Indica el código de la sala que quieres cancelar: A1
Indica el nombre del usuario que quiere cancelar la reserva: pedro

La reserva de la sala A1 se ha cancelado correctamente.
```

Error:

```
Introduce comando: CANCELAR
Indica el código de la sala que quieres cancelar: A1
Indica el nombre del usuario que quiere realizar la reserva: pedro
No tienes ninguna reserva activa sobre la sala A1.
```

Salida en el servidor

Éxito:

```
--> Cliente: CANCELAR|vicente|A1
La reserva de la sala A1 del usuario vicente ha sido eliminada.
```

Error:

```
--> Cliente: CANCELAR|vicente|A1
No existe reserva de la sala A1 para el usuario vicente.
```

5. Comando SALIR

Cuando el cliente envía el comando `SALIR` se cierra la comunicación con el servidor.
El servidor finaliza la ejecución del worker que estaba atendiendo al cliente.

Salida en el cliente

```
Introduce comando: EXIT
Sesión finalizada. Gracias por usar el sistema de reservas.
```

Salida en el servidor

```
--> Cliente: EXIT
El usuario se desconecta de la aplicación.
Conexión cerrada.
```

PARTE 2: PROTOCOLO CON ESTADOS (30%)

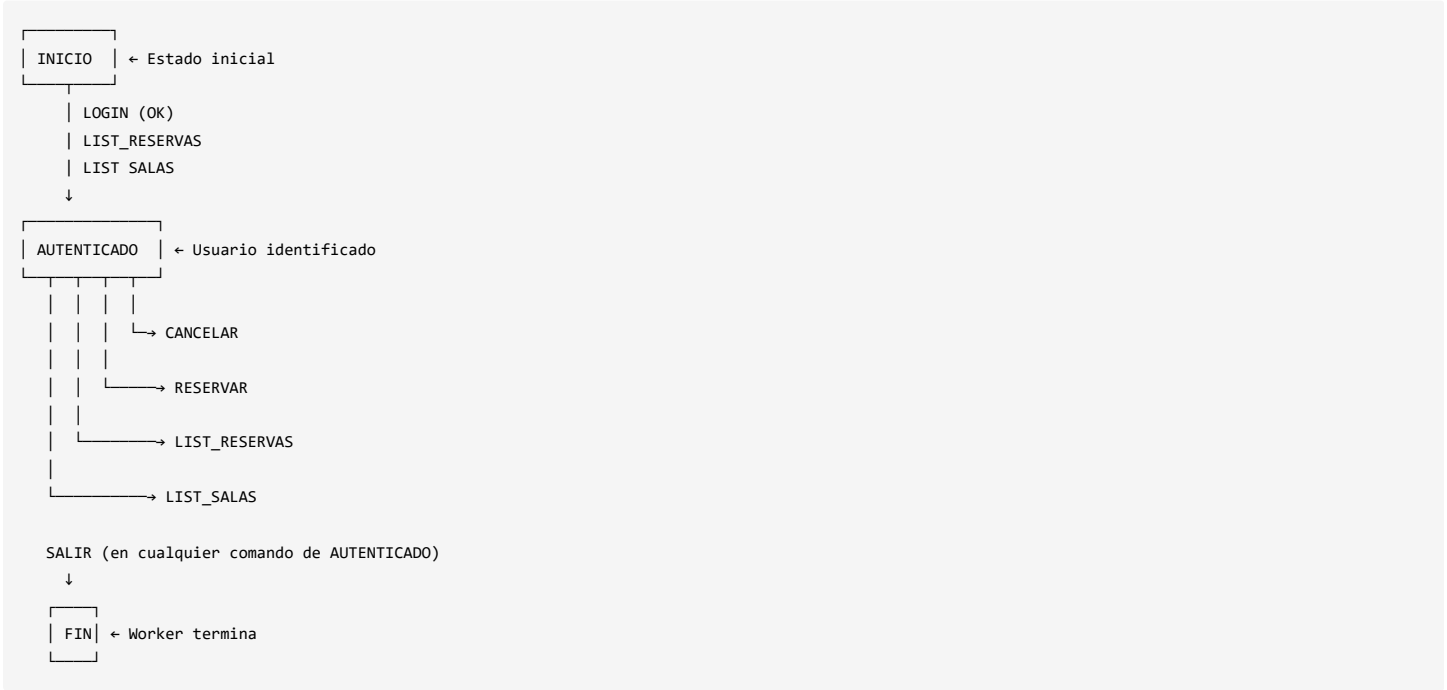
> ⚠ **Cliente sin estados**
>
> En la aplicación cliente solo se hacen los cambios necesarios para gestionar el nuevo comando `LOGIN` y las modificaciones en los comandos `CANCELAR` y `RESERVAR`.
> No se realiza ningún cambio relacionado con la gestión de los estados.

En esta segunda parte se **añade el comando LOGIN** y se mantiene **el mismo conjunto de comandos de la Parte 1**, pero definiendo un **protocolo con estados**, donde el orden de los comandos está restringido y se definen transiciones de estado claras.

Todos los comandos deben cumplir una secuencia de estados.

Máquina de Estados

Se propone la siguiente máquina de estados en el lado del servidor (worker):



Estado	Descripción	Comandos permitidos
INICIO	El usuario aún no se ha identificado.	`LOGIN`, `LIST_SALAS`, `LIST_RESERVAS`
AUTENTICADO	El usuario se ha identificado correctamente.	`LIST_SALAS`, `LIST_RESERVAS`, `RESERVAR`, `CANCELAR`, `EXIT`
FIN	El usuario ha salido; el worker termina.	Ninguno (la conexión se cierra).

Reglas de Transición

1. **Desde INICIO:**
   - Si se recibe `LOGIN|nombreUsuario`:
     - Si es exitoso (no está conectado), cambia a `AUTENTICADO` y responde `OK`.
     - Si falla (usuario ya conectado), responde `KO` y permanece en `INICIO`.
   - Si se recibe LIST_SALAS se procede como en la parte 1 y permanece en `INICIO`.
   - Si se recibe LIST_RESERVAS se procede como en la parte 1 y permanece en `INICIO`.
   - Si se recibe cualquier otro comando, responde `UNEXPECTED_CMD` y permanece en `INICIO`.
2. **Desde AUTENTICADO:**
   - Se permiten todos los comandos funcionales (`LIST_SALAS`, `LIST_RESERVAS`, `RESERVAR`, `CANCELAR`). Se ejecutan los comandos y se permanece en `AUTENTICADO`.
   - Si se recibe `EXIT`, cambia a `FIN`, responde `OK` y cierra el socket.
   - Si se recibe `LOGIN`, responde `UNEXPECTED_CMD` y permanece en `AUTENTICADO`.
3. **Desde FIN:**
   - El worker finaliza y no procesa más mensajes. La conexión se cierra.

Descripción de Comandos con Estados

Comando LOGIN (con estados)

Elemento	Detalle
Estados válidos	Solo `INICIO`.
Transición	Si OK: `INICIO` → `AUTENTICADO`. Si KO: permanece en `INICIO`.
En otro estado	Responder `UNEXPECTED_CMD`.

Salida en el cliente

Éxito:

```
Introduce tu nombre de usuario: pedro
Usuario identificado correctamente.
```

Error:

```
Introduce tu nombre de usuario: pedro
No se ha podido iniciar sesión. Prueba con otro nombre.
```

Salida en el servidor

Éxito:

```
--> Cliente: LOGIN|vicente
El usuario vicente ha entrado en el sistema.
Transición: INICIO → AUTENTICADO
```

Error:

```
--> Cliente: LOGIN|pedro
No se puede registrar al usuario pedro (ya conectado).
Permanece en estado INICIO.
```

Comando no esperado:

```
--> Cliente: LOGIN|pedro
Comando no esperado. El usuario ya está autenticado.
```

Comando RESERVAR (con estados)

Permite reservar una sala, asociándola al usuario que lanza la petición.

> **Cambios respecto a sin estados**
>
> El principal cambio es que en el cliente ya ni se solicita ni se envía el nombre de usuario como parte del mensaje.
>
> El servidor, usa el **nombre del usuario autenticado en la sesión** para intentar realizar la reserva.

Cliente

Paso	Acción
1	Solicita por teclado el código de sala a reservar.
2	Envía: `RESERVAR|nombreUsuario|codigoSala`
3	Espera respuesta: `OK` o `KO`.

Servidor

Paso	Acción
1	Recibe `RESERVAR|codigoSala`.
2	Comprueba que la sala existe.
3	Comprueba si la sala está disponible (no tiene reserva activa).
4	Si está disponible, crea una entrada asociando sala y **usuario autenticado** en las reservas y responde `OK`.
5	Si la sala no existe o está ocupada, responde `KO`.

> **Salida en el cliente**
>
> Éxito:
>
> ```
> Introduce comando: RESERVAR
> Indica el código de la sala que quieres reservar: A1
> Reserva realizada correctamente.
> ```
>
> Error:
>
> ```
> Introduce comando: RESERVAR
> Indica el código de la sala que quieres reservar: A1
> No se ha podido realizar la reserva (sala no existe u ocupada).
> ```

> **Salida en el servidor**
>
> Éxito:
>
> ```
> --> Cliente: RESERVAR|A1
> La sala A1 ha sido reservada por el usuario pedro.
> ```
>
> Error:

```
--> Cliente: RESERVAR||A1
No se puede reservar la sala A1 para pedro (ocupada o inexistente).
```

4. Comando CANCELAR (con estados)

Permite cancelar una reserva existente asociada a un usuario.

> **Cambios respecto a sin estados**
>
> El principal cambio es que en el cliente ya ni se solicita ni se envía el nombre de usuario como parte del mensaje.
>
> El servidor, usa el **nombre del usuario autenticado en la sesión** para intentar realizar la cancelación.
> Sólo se realiza la cancelación si la sala la tiene reservada el usuario.

Cliente

Paso	Acción
1	Solicita por teclado el código de la sala.
2	Envía: `CANCELAR|codigoSala`.
3	Espera respuesta: `OK` o `KO`.

Servidor

Paso	Acción
1	Recibe `CANCELAR|codigoSala`.
2	Busca en la estructura de reservas una reserva de esa sala asociada al **usuario autenticado**.
3	Si existe elimina la reserva de las reservas y responde `OK`.
4	Si no existe o **no está reservada por el usuario autenticado**, responde `KO`.

> **Salida en el cliente**
>
> Éxito:
>
> ```
> Introduce comando: CANCELAR
> Indica el código de la sala que quieres cancelar: A1
>
> La reserva de la sala A1 se ha cancelado correctamente.
> ```
>
> Error:
>
> ```
> Introduce comando: CANCELAR
> Indica el código de la sala que quieres cancelar: A1
> No tienes ninguna reserva activa sobre la sala A1.
> ```

> **Salida en el servidor**
>
> Éxito:
>
> ```
> --> Cliente: CANCELAR|A1
> La reserva de la sala A1 del usuario pedro ha sido eliminada.
> ```
>
> Error:

```
--> Cliente: CANCELAR|A1
No existe reserva de la sala A1 para el usuario pedro.
```

Comando EXIT (con estados)

Elemento	Detalle
Estados válidos	Solo `AUTENTICADO`.
Transición	`AUTENTICADO` → `FIN`.
Mensajes	Cliente envía `EXIT`. Servidor responde `OK` y cierra la conexión.
En otro estado	Si se recibe en `INICIO`, responder `UNEXPECTED_CMD`.

Salida en el cliente

```
Introduce comando: EXIT
Sesión finalizada. Gracias por usar el sistema de reservas.
```

Salida en el servidor

```
--> Cliente: EXIT
El usuario pedro sale de la aplicación.
Transición: AUTENTICADO → FIN
Conexión cerrada.
```

Ejemplo de Ejecución (Parte 2)

Caso 1: Secuencia válida

Cliente

```
Introduce tu nombre de usuario: vicente
Usuario identificado correctamente.

Selecciona una opción: 2
** Salas disponibles **
A1 - Aula Informática 1 (Aforo: 20)
A2 - Aula Informática 2 (Aforo: 25)
SALA3 - Sala de Reuniones (Aforo: 10)
LAB1 - Laboratorio Redes (Aforo: 15)
LAB2 - Laboratorio Servidores (Aforo: 12)

Selecciona una opción: 4
Indica el código de la sala que quieres reservar: A1
Reserva realizada correctamente.

Selecciona una opción: EXIT
Sesión finalizada. Gracias por usar el sistema de reservas.
```

Servidor

```
Escuchando: ServerSocket[addr=0.0.0.0/0.0.0.0,localport=4444]

--> Cliente: LOGIN|vicente
El usuario vicente ha entrado en el sistema.
Transición: INICIO → AUTENTICADO

--> Cliente: LIST_SALAS
Se envía el listado de 5 salas al usuario.
Estado actual: AUTENTICADO

--> Cliente: RESERVAR|vicente|A1
La sala A1 ha sido reservada por el usuario vicente.
Estado actual: AUTENTICADO

--> Cliente: EXIT
El usuario vicente sale de la aplicación.
Transición: AUTENTICADO → FIN
Conexión cerrada.
```

Caso 2: Comando no esperado en INICIO

Cliente

```
Introduce tu nombre de usuario:
(El cliente intenta un comando antes de LOGIN)
Comando no esperado. Primero debes iniciar sesión.

Introduce tu nombre de usuario: vicente
Usuario identificado correctamente.
```

Servidor

```
--> Cliente: LIST_SALAS
Comando no esperado. El usuario debe autenticarse primero.
Estado actual: INICIO

--> Cliente: LOGIN|vicente
El usuario vicente ha entrado en el sistema.
Transición: INICIO → AUTENTICADO
```

Caso 3: Comando no esperado en AUTENTICADO

Cliente

```
Introduce tu nombre de usuario: vicente
Usuario identificado correctamente.

Selecciona una opción: 1
Comando no esperado. Ya estás autenticado.

Selecciona una opción: 2
** Salas disponibles **
...
```

Servidor

```
--> Cliente: LOGIN|vicente
El usuario vicente ha entrado en el sistema.
Transición: INICIO → AUTENTICADO

--> Cliente: LOGIN|juan
Comando no esperado. El usuario ya está autenticado.
Estado actual: AUTENTICADO

--> Cliente: LIST_SALAS
Se envía el listado de 5 salas al usuario.
Estado actual: AUTENTICADO
```