

Universidad ORT Uruguay
Facultad de Ingeniería
Escuela de Tecnología

OBLIGATORIO PROGRAMACIÓN 1

DOCUMENTO DE ANÁLISIS



Manuel Bercianos – 309770



Rodrigo Fernández - 365329

M1B

Profesor: Santiago Fagnoni

Analista en tecnologías de la información.

[26-05-2025]

Índice

Análisis del Sistema 3

- Descripción general del problema a resolver
- Tipos de usuario del sistema
- Listado de funcionalidades

Detalle de Funcionalidades 8

- F01 - Registrarse en el sistema
- F02 - Seleccionar y Contratar paseador
- F03 - Cancelar contratación
- F04 - Visualizar listado de paseadores
- F05 - Calificar paseador
- F06 - Visualizar contrataciones pendientes
- F07 - Visualizar perros asignados
- F08 - Iniciar sesión
- F09 - Cerrar sesión

Casos de Prueba 25

- Casos de prueba por funcionalidad
- Datos de prueba precargados

Información Precargada 50

- Datos de clientes
- Datos de paseadores
- Contrataciones de ejemplo

1. Descripción general del problema a resolver

El sistema desarrollado es una aplicación web para conectar paseadores de perros con clientes que necesitan este servicio. La aplicación maneja dos tipos de usuarios: clientes (dueños de perros) y paseadores (son los prestadores del servicio).

1. Tipos de usuario del sistema

Cliente: Usuarios que contratan los servicios de paseadores para sus perros.

Paseador: Proveedor que realizan los paseos diarios con los perros a su cargo

2. Listado de funcionalidades

Funcionalidades del Cliente:

- **F01 - Registrarse en el sistema:** Permite a un nuevo cliente crear una cuenta en la aplicación.
- **F02 - Seleccionar y Contratar paseador:** Permite al cliente elegir y contratar un paseador de la lista de paseadores disponibles.
- **F03 - Cancelar contratación:** Permite al cliente cancelar una contratación pendiente.
- **F04 - Visualizar listado de paseadores:** Permite al cliente ver un listado de todos los paseadores y la cantidad de cupos que tienen disponibles
- **F05 - Calificar paseador:** Permite al cliente calificar y dejar comentarios sobre el servicio del paseador.

Funcionalidades del Paseador:

- **F06 - Visualizar contrataciones pendientes:** Permite al paseador ver la lista de contrataciones que están pendientes de aprobación.
- **F07 - Visualizar perros asignados:** Permite al paseador ver una lista de todos los perros que tiene asignados, junto con información de cupos.

Funcionalidades Compartidas:

- **F08 - Iniciar sesión:** Permite a un paseador/cliente acceder a la aplicación.
- **F09 - Cerrar sesión:** Permite al paseador/cliente cerrar su sesión en la aplicación.

Funcionalidades Auxiliares:

- **F10 - obtenerElemento(arrBuscar, prop, valor)** Busca un objeto específico en un array basándose en una propiedad y su valor.
- **F11 - verificarBotonesDeshabilitados()** Valida y deshabilita botones de "Procesar Contratación" cuando no se pueden aprobar por falta de cupo o incompatibilidad de tamaños.
- **F12 - validarProcesarContratacion()** Configura el estado inicial del paseador al iniciar sesión, resetea propiedades y descuenta cupos según contrataciones aprobadas.
- **F13 - mostrarBotones(tipoUsuario)** Muestra solo los botones correspondientes al tipo de usuario logueado (inicio, cliente o paseador).
- **F14 - ocultarSecciones()** Oculta todas las secciones antes de mostrar una nueva, preparando la interfaz para el cambio de contenido.
- **F15 - funcionEstrella()** Maneja la selección de estrellas en el sistema de calificación, actualiza visualmente las estrellas seleccionadas.
- **F16 - procesarContratacion()** Procesa las contrataciones pendientes de los clientes. Verifica si el paseador tiene cupo suficiente y si el tamaño del perro es compatible con los perros ya asignados.
- **F17 - mostrarContratacionesDelPaseador()** Muestra en una tabla HTML todas las contrataciones aprobadas asignadas al paseador logueado.
- **F18 - funcionComentar()** Procesa la calificación y comentario del cliente hacia quien fue su paseador asignado.

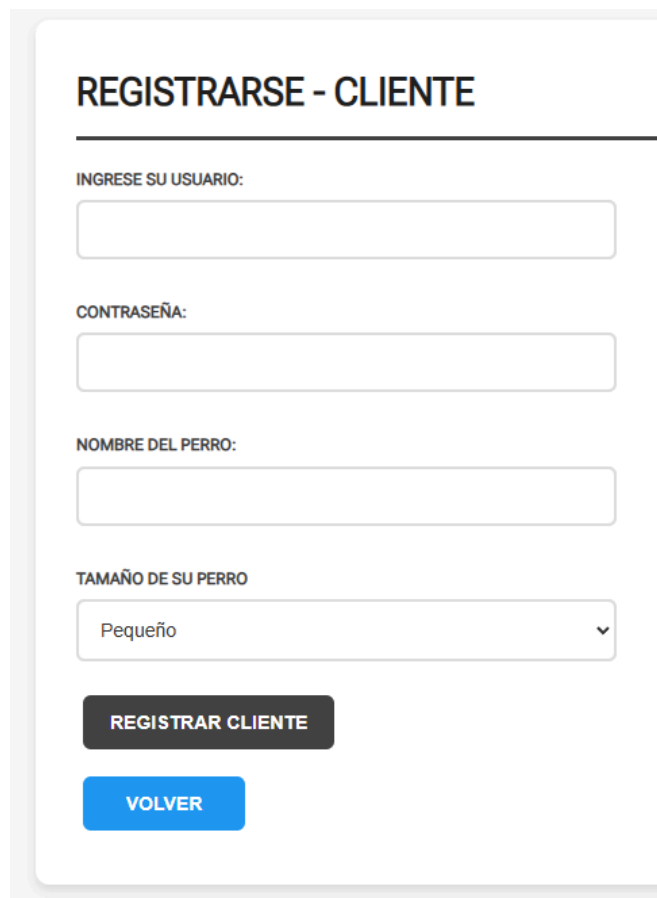
Detalle de Funcionalidades

En la parte siguiente, se presenta el detalle de cada una de las funcionalidades a resolver en el obligatorio. Las funcionalidades están ordenadas por tipo de usuario.

F01 - Registrarse en el sistema.

- **Acceso: Cliente**
- **Descripción:**
 - Esta funcionalidad permite a un nuevo usuario crear una cuenta en la aplicación como cliente. Al registrarse, el usuario deberá proporcionar la información necesaria para utilizar los servicios, incluyendo sus datos personales y los de su perro como su tamaño

Interfaz de usuario:



The image shows a user registration form titled "REGISTRARSE - CLIENTE". The form is contained within a light gray rounded rectangle. It includes four input fields: "INGRESE SU USUARIO:", "CONTRASEÑA:", "NOMBRE DEL PERRO:", and "TAMAÑO DE SU PERRO". The "TAMAÑO DE SU PERRO" field is a dropdown menu with "Pequeño" selected. Below the input fields are two buttons: a dark gray "REGISTRAR CLIENTE" button and a blue "VOLVER" button.

REGISTRARSE - CLIENTE

INGRESE SU USUARIO:

CONTRASEÑA:

NOMBRE DEL PERRO:

TAMAÑO DE SU PERRO

Pequeño

REGISTRAR CLIENTE

VOLVER

- **Validaciones:**

- Nombre de usuario: Obligatorio

- Único en el sistema
 - Mensaje de error: "El nombre de usuario ya existe. Por favor, elija otro."
- Longitud mínima: 5 caracteres (ejemplo)
 - Mensaje de error: "El nombre de usuario debe tener al menos 5 caracteres."

- Contraseña: Obligatoria

- Longitud mínima: 5 caracteres

- Mensaje de error: "La contraseña debe tener al menos 5 caracteres."
- Debe contener al menos una letra mayúscula
- Mensaje de error: "La contraseña debe contener al menos una letra mayúscula."
- Debe contener al menos una letra minúscula
- Mensaje de error: "La contraseña debe contener al menos una letra minúscula."
- Debe contener al menos un número
- Mensaje de error: "La contraseña debe contener al menos un número."
- Nombre del perro: Obligatorio
- Tamaño del perro: Obligatorio
 - Debe ser uno de los valores permitidos, según lo indica del combo desplegable: ("Grande", "Mediano", "Chico")

F02 - Seleccionar y Contratar paseador

- **Acceso:** Cliente
- **Descripción:** Esta parte de la funcionalidad permite al cliente elegir un paseador de la lista de paseadores disponibles para contratar sus servicios, la lista de paseadores mostrada al cliente debe tener en cuenta la disponibilidad del paseador (cupos) y por otro lado, las restricciones de tamaño de los perros (un paseador no puede pasear perros grandes y chicos al mismo tiempo).
 - Lista/Combo desplegable de paseadores:
 - Muestra el nombre de cada paseador disponible.
 - Puede incluir información adicional como la cantidad de perros que pasea actualmente o su calificación (si se implementa).
 - Mensaje informativo (si no hay paseadores disponibles): "No hay paseadores disponibles para el tamaño de su perro en este momento."
- **Interfaz de usuario:**

CONTRATAR PASEADOR

NOMBRE DEL PASEADOR	CUPO DISPONIBLE	GESTION
Jorge	10	CONTRATAR PASEADOR
Rodrigo	10	CONTRATAR PASEADOR
Manuel	15	CONTRATAR PASEADOR
Ana	8	CONTRATAR PASEADOR
Lucia	12	CONTRATAR PASEADOR

VOLVER

- **Validaciones:**
 - La lista de paseadores debe filtrarse automáticamente según:
 - Disponibilidad de cupo: El paseador debe tener cupo suficiente para el tamaño del perro del cliente.

- Tamaño del perro: El paseador no debe tener asignados perros de tamaño incompatible (si el cliente tiene un perro chico, solo se muestran paseadores sin perros grandes asignados, y viceversa).
- Si el cliente intenta realizar una contratación sin seleccionar un paseador:
 - Mensaje de error: "Por favor, seleccione un paseador."
- Verificar que se haya seleccionado un paseador (esto ya se validó en F04, pero se puede volver a verificar).
 - Mensaje de error: "Por favor, seleccione un paseador."
- Verificar que el cliente no tenga ya una contratación pendiente.
 - Mensaje de error: "Ya tiene una contratación pendiente. Por favor, espere a que sea aprobada o rechazada, o cancele la contratación actual."
- Mensajes:
 - Mensaje de éxito: "Contratación realizada. Esperando aprobación del paseador."
 - Mensaje de error: "Ya tiene una contratación pendiente. Por favor, espere a que sea aprobada o rechazada, o cancele la contratación actual."

F03 - Cancelar contratación

- **Acceso:** Cliente.
- **Descripción:** Permite al cliente cancelar una contratación pendiente con un paseador.
 - Botón: "Cancelar Contratación" (Visible sólo si hay una contratación pendiente)
 - Mensaje de confirmación: "Contratación cancelada."
 - Mensaje de error: "No hay contratación pendiente." (Si se intenta cancelar sin tener una)
- **Validaciones:**
 - Verificar si existe una contratación pendiente antes de permitir la cancelación.

- **Interfaz de usuario:**

CANCELAR CONTRATACIONES

PASEADOR PENDIENTE	ACCIÓN
Rodrigo	CANCELAR CONTRATACIÓN

VOLVER A OPCIONES

F04 - Visualizar listado de paseadores

- **Acceso:** Cliente
- **Descripción:** Permite al cliente ver un listado de todos los paseadores registrados en la aplicación, junto con información relevante sobre cada uno, como el cupo disponible que tienen actualmente. Esto ayuda al cliente a tener una visión general de los paseadores disponibles.
 - Listado de Paseadores:
 - Muestra el nombre de cada paseador.
 - Muestra la cantidad de perros que tiene asignados cada paseador.
- **Interfaz de usuario:**

TODOS LOS PASEADORES

NOMBRE DEL PASEADOR	CUPO DISPONIBLE
Jorge	10
Rodrigo	10
Manuel	15
Ana	8
Lucia	12

VOLVER

F5 - Calificar paseador

- **Acceso:** Cliente
- **Descripción:** Permite al cliente asignar una calificación (por ejemplo, de 1 a 5 estrellas) y dejar un comentario opcional sobre el servicio proporcionado por el paseador. Esta retroalimentación ayuda a otros clientes a tomar decisiones y a la administración a mantener un registro de la calidad del servicio.
- - **Formulario de Calificación:**
 - Sistema de 5 estrellas clickeables y área de texto para comentarios.
 - Área de texto opcional para el comentario.
 - Botón "Enviar Calificación".
 - **Mensajes:**
 - Mensaje de éxito: Muestra la calificación enviada con formato estructurado
 - Mensaje de error: "Seleccione una cantidad de estrellas. Escriba un comentario."
 - Mensaje de error: "No tienes ningún paseador al cual calificar (Para poder calificar a un paseador debes de tener una contratación aceptada por alguno de ellos)."
 -
- **Validaciones:**
 - Verificar que se haya seleccionado una calificación.
 - Mensaje de error: "Por favor, selecciona una calificación."
 - La calificación debe estar dentro del rango permitido (por ejemplo, de 1 a 5).
 - Mensaje de error: "La calificación debe estar entre 1 y 5." (Si se implementa la validación del rango en el cliente)."
- **Validaciones:**
 - Verificar que se haya seleccionado una calificación (1-5 estrellas)
 - Verificar que el cliente tenga una contratación aprobada
 - El comentario es opcional pero si no se proporciona junto con las estrellas, se muestra error

Interfaz de usuario:

CALIFICAR PASEADOR

Tu Paseador: Rodrigo

RESEÑA:

★★★★★

ENVIAR CALIFICACIÓN

VOLVER A OPCIONES

Cliente: cliente1
Paseador calificado: Rodrigo
Comentario: ¡Excelente Servicio! Gracias Rodrigo
Calificación: ★★★★★

F6 - Visualizar contrataciones pendientes

Acceso: Paseador.

- **Descripción:** Esta funcionalidad permite al paseador ver una lista de todas las contrataciones que se encuentran en estado “pendiente”. Esto le permite al paseador gestionar las solicitudes de servicio de los clientes.
-
- Listado de contrataciones pendientes.
- Muestra información relevante de cada contratación pendiente como:
 - Nombre del cliente
 - Nombre del perro
 - Tamaño del perro
- Para cada contratación, incluye botón:

Aprobar la contratación

(si ya se llegó al cupo automáticamente el sistema rechaza el resto de las solicitudes, consultas de todas formas)

- Mensaje
 - Éxito
 - Error

- Validaciones

Cupo: Rechazar si no hay cupo suficiente (Grande 4, Mediano 2, Chico 1)

Tamaño: Rechazar perros chicos con grandes asignados y viceversa.

Interfaz de Usuario:

CONTRATACIONES PENDIENTES		
NOMBRE DEL PERRO	USUARIO	ACCIÓN
Fafa	Jose	PROCESAR CONTRATACION
Rex	Roberto	SIN CUPO DISPONIBLE
Tití	Simon	PROCESAR CONTRATACION
Mar	Claudia	PROCESAR CONTRATACION
Colo	Pablo	PROCESAR CONTRATACION
Beto	Josefina	PROCESAR CONTRATACION
Firu	Maite	SIN CUPO DISPONIBLE
Dulce	Diana	PROCESAR CONTRATACION
Teco	Julietta	PROCESAR CONTRATACION
severus	Valeria	PROCESAR CONTRATACION

VOLVER A OPCIONES

F7 - Visualizar perros asignados:

- **Acceso:** Paseador.
- **Descripción:** El paseador podrá ver un listado donde tenga información de los perros asignados como también los cupos que tiene en el momento.
 - Listado de perros asignados:
 - Número.
 - Tamaño.
 - Información General:
 - Cupos disponibles.
 - Cupos máximos.
 - Porcentaje de cupos asignados.

Si el paseador no tiene perros asignados entonces se le mostrará un mensaje que diga (“No hay perros asignados actualmente”).

Interfaz de usuario:

PERROS ASIGNADOS		
NOMBRE DEL PERRO	TAMAÑO DEL PERRO	CLIENTE
Sarha	Grande	Tomas
Thor	Mediano	Mario
VOLVER A OPCIONES		

F08 - Iniciar sesión

- **Acceso:** Paseador/Cliente.
- **Descripción:** Esta funcionalidad permite a los usuarios acceder a la aplicación mediante, (“nombre de usuario” y “contraseña”).
 - Formulario de Inicio de Sesión:
 - Campos:
 - Nombre de usuario (text)
 - Contraseña (password)
 - Botón:
 - "Iniciar Sesión"
 - Mensajes:

- Mensaje de éxito Cliente: "Sesión iniciada correctamente [nombre usuario]"
- Mensaje de éxito Paseador: "Sesión iniciada correctamente [nombre usuario]"
- Mensaje de error: "Por favor, ingrese un usuario y contraseña válidos"
- Mensaje de error: "Por favor, complete ambos campos."

Validaciones:

- Verificar que ambos campos estén completos
 - Verificar credenciales contra la base de datos
 - Redirigir al panel correspondiente según tipo de usuario
-
- **Interfaz de usuario:**

INICIAR SESIÓN - CLIENTE

INGRESE SU USUARIO:

CONTRASEÑA:

INICIAR SESIÓN

VOLVER

INICIAR SESIÓN - PASEADOR

INGRESE SU USUARIO:

CONTRASEÑA:

INICIAR SESIÓN

VOLVER

F9 - Cerrar sesión

Acceso: Paseador/Cliente.

- **Descripción:** Esta funcionalidad permite al usuario cerrar su sesión actual en la aplicación y volver a la pantalla de inicio de sesión

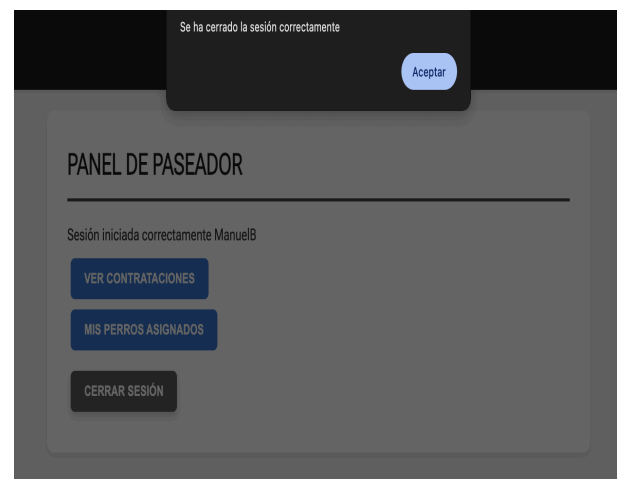
Funcionalidad:

- Limpia todos los campos de entrada

- Resetea las variables de sesión
- Muestra mensaje: "Se ha cerrado la sesión correctamente"
- Redirige a la pantalla de inicio

Botón: "Cerrar Sesión" (Este botón está presente en el panel principal de los dos usuarios)

- **Interfaz de Usuario:**



Funciones auxiliares:

F10 - ObtenerElemento(arrBuscar, prop, valor)

Funcionalidad: Buscar un objeto específico en un array, basándose en alguna propiedad y su valor.

Ejemplo de uso:

```
let cliente = obtenerElemento(sistema.cliente, "usuario", "Marcos");
```

F11 - verificarBotonesDeshabilitados()

Funcionalidad: Validar y deshabilitar botones de “Procesar Contratación” cuando no se pueden aprobar

Qué hace:

- Revisa cada botón de procesamiento en la tabla de contrataciones pendientes
- Calcula si el paseador tiene cupo suficiente para cada tipo de perro
- Verifica las restricciones de compatibilidad (no mezclar perros grandes con chicos)

- Deshabilitar botones y cambia su texto a "Sin cupo disponible" cuando no se puede procesar
- Actualiza la información de cupos disponibles en pantalla

Validaciones que realiza:

- Perros grandes: necesitan 4 cupos y no pueden ir con perros chicos
- Perros medianos: necesitan 2 cupos (sin restricciones)
- Perros chicos: necesitan 1 cupo y no pueden ir con perros grandes

Apoya a: procesarContratacion() y tablaContratacionesPendientes()

F-12. validarProcesarContratacion()

Tipo: Función auxiliar de inicialización

Propósito: Configurar el estado inicial del paseador al iniciar sesión.

Qué hace:

- Resetea las propiedades perroGrande y perroChico del paseador actual
- Recorre todas las contrataciones aprobadas del sistema
- Descuenta cupos de todos los paseadores según sus contrataciones aprobadas
- Marca qué tipos de perros tiene asignados cada paseador
- Actualiza la propiedad cupoDescontado para evitar descuentos duplicados

Cálculo de cupos:

- Perro grande: -4 cupos
- Perro mediano: -2 cupos
- Perro chico: -1 cupo

F-13. MostrarBotones(tipoUsuario)

Tipo: Función auxiliar de interfaz

Propósito: Mostrar solo los botones correspondientes al tipo de usuario logueado.

Qué hace:

- Oculta todos los botones de navegación
- Muestra solo los botones que corresponden al tipo de usuario actual
- Utiliza clases CSS para identificar qué botones mostrar

Tipos de usuario:

- "inicio": Botones de la página principal
- "cliente": Botones del área de cliente
- "paseador": Botones del área de paseador

Apoya a: cambiarSeccion() y las funciones de inicio de sesión.

F14. ocultarSecciones()

Tipo: Función auxiliar de interfaz

Propósito: Ocultar todas las secciones antes de mostrar una nueva.

Qué hace:

- Selecciona todas las secciones con clase .seccion e .inicio
- Establece display: none en todas ellas
- Prepara la interfaz para mostrar la nueva sección

Apoya a: cambiarSeccion() para limpiar la pantalla antes de mostrar nueva contenido.

F15. funcionEstrella() (dentro del bucle de estrellas)

Tipo: Función auxiliar de interfaz

Propósito: Manejar la selección de estrellas en el sistema de calificación.

Qué hace:

- Captura qué estrella fue clickeada
- Actualiza la variable estrellasSeleccionadas
- Cambia visualmente las estrellas (★ para seleccionadas, ☆ para no seleccionadas)
- Actualiza todas las estrellas desde la 1 hasta la seleccionada

Apoya a: funcionComentar() para procesar la calificación del paseador.

Casos de Prueba

Casos de Prueba - F01 (Registrarse en el sistema)

Caso 1: Registro exitoso

- Usuario: "nuevoUsuario"
- Contraseña: "Abc123"
- Nombre perro: "Firulais"
- Tamaño perro: "Grande"
- Resultado esperado: "Cliente registrado correctamente: nuevoUsuario"

Caso 2: Usuario ya existe

- Usuario: "cliente1" (ya existe)
- Resultado esperado: "El usuario ya existe. Por favor, elija otro."

Caso 3: Contraseña inválida

- Usuario: "nuevoUsuario2"
- Contraseña: "abc" (no cumple requisitos)
- Resultado esperado: Mensaje detallando los requisitos no cumplidos

Casos de Prueba - F02/F03 (Seleccionar/Contratar paseador)

Caso 1: Contratación exitosa

- Cliente: "cliente1" (perro Grande)
- Paseador disponible: Jorge (cupó suficiente)
- Resultado esperado: "Solicitud de contratación enviada para Jorge. Estado: Pendiente de aprobación."

Caso 2: Sin cupo disponible

- Cliente con perro Grande, todos los paseadores sin cupo para perros grandes
- Resultado esperado: "No hay paseadores disponibles en este momento."

Caso 3: Contratación duplicada

- Cliente que ya tiene una contratación pendiente
- Resultado esperado: "Ya tienes una contratación pendiente."

Casos de Prueba - F04 (Cancelar contratación)

Caso 1: Cancelación exitosa

- Cliente con contratación pendiente
- Resultado esperado: "Contratación cancelada correctamente."

Caso 2: Sin contrataciones para cancelar

- Cliente sin contrataciones pendientes
- Resultado esperado: Tabla vacía

Casos de Prueba - F06 (Calificar paseador)

Caso 1: Calificación completa

- Cliente: con contratación aprobada
- Estrellas: 5

- Comentario: "Excelente servicio"
- Resultado esperado: Muestra calificación formateada

Caso 2: Sin contratación aprobada

- Cliente sin contrataciones aprobadas
- Resultado esperado: "No tienes ningún paseador al cual calificar..."

Casos de Prueba - F07 (Procesar contrataciones)

Caso 1: Aprobación exitosa

- Paseador: Jorge (cupos disponibles)
- Contratación: Perro mediano (requiere 2 cupos)
- Resultado esperado: "La contratación ha sido aprobada. Debes pasar a buscar a [perro]..."

Caso 2: Sin cupos suficientes

- Paseador sin cupos suficientes
- Resultado esperado: Botón deshabilitado "Sin cupos disponibles"

Casos de Prueba - F09 (Iniciar sesión)

Caso 1: Login cliente exitoso

- Usuario: "cliente1"
- Contraseña: "Aa123"
- Resultado esperado: "Sesión iniciada correctamente cliente1"

Caso 2: Login paseador exitoso

- Usuario: "JorgeB"
- Contraseña: "Jorge123"
- Resultado esperado: "Sesión iniciada correctamente JorgeB"

Caso 3: Credenciales incorrectas

- Usuario: "usuarioQueNoExiste"

- Contraseña: "contraseñaQueNoExiste"
- Resultado esperado: "Por favor, ingrese un usuario y contraseña válidos"

Código JavaScript:

```
// Clase cliente, se encarga de gestionar el apartado de clientes.
// Esta clase es la que se encarga de gestionar los clientes, sus datos y sus
perros.
// Además, se encarga de gestionar las contrataciones de paseadores.
let logeado = false;
let paseadorActual = null;
let proximoID = 31; // ID que aumenta de uno en uno para nuevos registros
let tipoUsuario = "inicio";

// Hacer que cuando alguien te acepte la contratación, el resto de paseadores
se bloquee, hacer que cuando el perro no sea compatible, el paseador se
elimine de la lista de contratar paseadores

class Cliente {
  constructor(idC, usuarioC, contraC, nombreperroC, sizeperroC) {
    this.id = idC;
    this.usuario = usuarioC;
    this.contra = contraC;
    this.nombreperro = nombreperroC;
    this.sizeperro = sizeperroC;
  }
}

// Clase paseador, se encarga de gestionar el apartado de paseadores.
// Esta clase es la que se encarga de gestionar los paseadores, sus datos y
sus cupos máximos.
// Además, se encarga de gestionar las contrataciones de paseadores.

class Paseador {
  constructor(numeroIP, nombreP, cupoMaxPA, cupoMAXP, usuarioP, contraP) {
    this.numeroIdentificador = numeroIP;
    this.nombre = nombreP;
    this.cupoActual = cupoMaxPA; // Cupo disponible actual
    this.cupoMax = cupoMAXP // Cupo máximo total
    this.usuario = usuarioP;
    this.contra = contraP;
  }
}
```

```

}

class Contratacion {
    constructor(numeroID, clienteCP, paseadorCP, estadoCP, nombreperroC,
sizeperroC) {
        this.numeroIdentificador = numeroID;
        this.cliente = clienteCP;
        this.paseador = paseadorCP;
        this.estado = estadoCP; // Estados: "Pendiente", "Aprobada", "Cancelada"
        this.nombreperro = nombreperroC;
        this.sizeperro = sizeperroC;
        this.cupoDescontado = false; // Para controlar si ya se descontó el cupo
    }
}

class paseadorLogeado {
    constructor(idPL, usuarioPL, contraPL) {
        this.id = idPL;
        this.usuario = usuarioPL;
        this.contra = contraPL;
    }
}

// Clase sistema, se encarga de gestionar el sistema en general.
// Esta clase es la que se encarga de gestionar los paseadores, clientes y
contrataciones.

class Sistema {
    constructor() {
        // Datos de prueba hardcodeados
        this.paseador = [
            new Paseador(1, "Jorge", 10, 10, "JorgeB", "Jorge123"), // El orden es
Nombre Paseador, Cupo Máximo, Usuario Paseador, Contraseña Paseador
            new Paseador(2, "Rodrigo", 10, 10, "RodrigoF", "Rodrigo123"),
            new Paseador(3, "Manuel", 15, 15, "ManuelB", "Manuel123"),
            new Paseador(4, "Ana", 8, 8, "AnaG", "Ana123"),
            new Paseador(5, "Lucia", 12, 12, "LuciaM", "Lucia123"),
        ];

        this.cliente = [
            new Cliente(1, "cliente1", "Aa123", "Fido", "Grande"),
            new Cliente(2, "cliente2", "Bb234", "Luna", "Chico"),
            new Cliente(3, "cliente3", "Cc345", "Max", "Mediano"),
            new Cliente(4, "cliente4", "Dd456", "Rex", "Grande"),
        ];
    }
}

```

```

    new Cliente(5, "cliente5", "Ee567", "Tina", "Chico"),
    new Cliente(6, "cliente6", "Ff678", "Rocky", "Mediano"),
    new Cliente(7, "cliente7", "Gg789", "Bella", "Grande"),
    new Cliente(8, "cliente8", "Hh890", "Coco", "Chico"),
    new Cliente(9, "cliente9", "Ii901", "Bruno", "Chico"),
    new Cliente(10, "cliente10", "Jj012", "Kira", "Chico"),
    new Cliente(11, "Marcos", "Marcos123", "Firulais", "Grande"),
    new Cliente(12, "Laura", "Laura123", "Luna", "Chico"),
    new Cliente(13, "Carlos", "Carlos123", "Snoppy", "Mediano"),
    new Cliente(14, "Ana", "Ana123", "Simba", "Chico"),
    new Cliente(15, "Selma", "Selma123", "Juan", "Grande"),
    new Cliente(16, "Pedro", "Pedro123", "Gusi", "Mediano"),
    new Cliente(17, "Sofia", "Sofia123", "Tobias", "Chico"),
    new Cliente(18, "Miguel", "Miguel123", "Tortuga", "Grande"),
    new Cliente(19, "Elena", "Elena123", "Mesi", "Mediano"),
    new Cliente(20, "Roberto", "Roberto123", "Jamon", "Chico"),
    new Cliente(21, "Lucia", "Lucia123", "Bob", "Grande"),
    new Cliente(22, "Andres", "Andres123", "Lola", "Mediano"),
    new Cliente(23, "Angel", "Angel123", "Kiara", "Chico"),
    new Cliente(24, "Mariana", "Mariana123", "Theo", "Grande"),
    new Cliente(25, "Manuela", "Manuela123", "Sasha", "Mediano"),
    new Cliente(26, "Diego", "Diego123", "Duke", "Grande"),
    new Cliente(27, "Paula", "Paula123", "Cordero", "Chico"),
    new Cliente(28, "Matias", "Matias123", "Maxi", "Mediano"),
    new Cliente(29, "Susana", "Susana123", "Max", "Mediano"),
    new Cliente(30, "Georgina", "Georgina123", "Pia", "Chico"),
];

this.contrataciones = [
    new Contratacion(1, "Jose", 2, "Pendiente", "Fafa", "Mediano"),
    new Contratacion(2, "Roberto", 2, "Pendiente", "Rex", "Grande"),
    new Contratacion(4, "Simon", 2, "Pendiente", "Titi", "Chico"),
    new Contratacion(5, "Claudia", 2, "Pendiente", "Mar", "Mediano"),
    new Contratacion(7, "Pablo", 2, "Pendiente", "Colo", "Chico"),
    new Contratacion(9, "Josefina", 2, "Pendiente", "Beto", "Mediano"),
    new Contratacion(10, "Maite", 2, "Pendiente", "Firu", "Grande"),
    new Contratacion(11, "Celeste", 2, "Pendiente", "Toto", "Chico"),
    new Contratacion(12, "Diana", 2, "Pendiente", "Dulce", "Mediano"),
    new Contratacion(13, "Julieta", 2, "Pendiente", "Teco", "Mediano"),
    new Contratacion(14, "Valeria", 2, "Pendiente", "severus", "Mediano"),

    new Contratacion(13, "Gonzalo", 1, "Aprobada", "Poncho", "Chico"),
    new Contratacion(14, "Santiago", 1, "Aprobada", "Maru", "Mediano"),
    new Contratacion(17, "Tomas", 3, "Aprobada", "Sarha", "Grande"),

```

```

        new Contratacion(18, "Mario", 3, "Aprobada", "Thor", "Mediano"),
        new Contratacion(19, "Angela", 4, "Aprobada", "Kia", "Chico"),
        new Contratacion(20, "Andrea", 4, "Aprobada", "Leo", "Mediano"),
        new Contratacion(21, "Lucas", 5, "Aprobada", "Berto", "Grande"),
        new Contratacion(22, "Romina", 5, "Aprobada", "Juju", "Mediano")
    ];
}
}
let sistema = new Sistema();

// Función para obtener un elemento de un array por una propiedad y su valor.
// Esta función recorre un array de objetos y devuelve el primer objeto que
// tenga la propiedad especificada con el valor dado.

function obtenerElemento(arrBuscar, prop, valor) {
    let objDev = null;
    for (let i = 0; i < arrBuscar.length; i++) {
        const obj = arrBuscar[i];
        if (obj[prop] === valor) {
            objDev = obj;
            break;
        }
    }
    return objDev;
}

// Muestra en tabla HTML todas las contrataciones pendientes que puede aceptar
// el paseador logueado
function tablaContratacionesPendientes() {
    let tablaHTML = "";
    if (logueado) {
        for (let i = 0; i < sistema.contrataciones.length; i++) {
            const unObjetoContratacion = sistema.contrataciones[i];

            // Solo mostrar contrataciones con estado "Pendiente"
            if (unObjetoContratacion.estado === "Pendiente") {
                tablaHTML += `<tr>
                    <td>${unObjetoContratacion.nombreperro}</td>
                    <td>${unObjetoContratacion.cliente}</td>
                    <td>${unObjetoContratacion.sizeperro}</td>
                    <td><input type="button" value="Procesar Contratacion"
class="botonProcesar"
data-id="${unObjetoContratacion.numeroIdentificador}"></td>
                </tr>`;
            }
        }
    }
}

```



```

    }
  }
  document.querySelector("#tblContratacionesPendientes").innerHTML =
tablaHTML;

  // Agregar event listeners a los botones de procesar
  let botonesProcesar = document.querySelectorAll(".botonProcesar");
  for (let i = 0; i < botonesProcesar.length; i++) {
    const boton = botonesProcesar[i];
    boton.addEventListener("click", procesarContratacion);
  }
}
}

// Lógica principal para que el paseador acepte o rechace contrataciones según
disponibilidad y restricciones
function procesarContratacion() {
  let idProcesarContratacion = Number(this.getAttribute("data-id"));
  let costoCupo = 0;

  for (let i = 0; i < sistema.contrataciones.length; i++) {
    let cupoActual = paseadorActual.cupoActual;

    if (sistema.contrataciones[i].numeroIdentificador ===
idProcesarContratacion && sistema.contrataciones[i].estado === "Pendiente") {

      // Calcular costo de cupo según tamaño del perro
      if (sistema.contrataciones[i].sizeperro === "Grande") {
        paseadorActual.perroGrande = true; // Marcar que tiene perro grande
        console.log("unPerroGrande : Existe")
        costoCupo = 4;
      } else if (sistema.contrataciones[i].sizeperro === "Mediano") {
        costoCupo = 2;
      } else if (sistema.contrataciones[i].sizeperro === "Chico") {
        paseadorActual.perroChico = true; // Marcar que tiene perro chico
        console.log("unPerroChico: Existe")
        costoCupo = 1;
      }
      console.log("Cupo actual del paseador: " + cupoActual);

      // Verificar si tiene cupo suficiente Y si no hay incompatibilidad de
tamaños
      // Regla: no puede tener perros grandes Y chicos al mismo tiempo

```

```

        if (paseadorActual.cupoActual >= costoCupo &&
(sistema.contrataciones[i].sizeperro === "Mediano" ||
(paseadorActual.perroChico !== paseadorActual.perroGrande))) {
            console.log("Entró al IF, va a aprobar");
            sistema.contrataciones[i].estado = "Aprobada";
            sistema.contrataciones[i].cupoDescontado = true;
            sistema.contrataciones[i].paseador =
paseadorActual.numeroIdentificador;
            paseadorActual.cupoActual -= costoCupo; // Descontar cupo
            alert(`La contratación ha sido aprobada. Debes pasar a buscar a
${sistema.contrataciones[i].nombreperro} ; Cupo:
${paseadorActual.cupoActual}`);
            tablaContratacionesPendientes();
            verificarBotonesDeshabilitados();
        }
    }
}

// Deshabilita botones de contrataciones que el paseador no puede aceptar por
falta de cupo o incompatibilidad
function verificarBotonesDeshabilitados() {
    let botones = document.querySelectorAll(".botonProcesar");
    let cupoActual = paseadorActual.cupoActual;
    let cupoTotal = paseadorActual.cupoMax;

    for (let j = 0; j < botones.length; j++) {
        let botonIdContratacion = Number(botones[j].getAttribute("data-id"));

        for (let k = 0; k < sistema.contrataciones.length; k++) {
            if (sistema.contrataciones[k].numeroIdentificador === botonIdContratacion
&& sistema.contrataciones[k].estado === "Pendiente") {

                // Deshabilitar si no hay cupo suficiente o hay incompatibilidad de
tamaños
                if ((sistema.contrataciones[k].sizeperro === "Grande" && cupoActual <
4) || (paseadorActual.perroChico === true &&
sistema.contrataciones[k].sizeperro === "Grande")) {
                    botones[j].setAttribute("value", "Sin cupo disponible");
                    botones[j].setAttribute("disabled", "disabled");
                }
                if (sistema.contrataciones[k].sizeperro === "Mediano" && cupoActual <
2) {
                    botones[j].setAttribute("value", "Sin cupo disponible");

```

```

        botones[j].setAttribute("disabled", "disabled");
    }

    if ((sistema.contrataciones[k].sizeperro === "Chico" && cupoActual < 1)
|| (paseadorActual.perroGrande === true && sistema.contrataciones[k].sizeperro
=== "Chico")) {
        botones[j].setAttribute("value", "Sin cupo disponible");
        botones[j].setAttribute("disabled", "disabled");
    }

    console.log("Cupo actual del paseador: " + paseadorActual.cupoActual);
    // Actualizar información de cupos en pantalla
    document.querySelector("#pTotalCuposMax").innerHTML = "Total de cupos
disponibles: " + paseadorActual.cupoActual + "/" + cupoTotal;
    document.querySelector("#pPorcentajeCupos").innerHTML = "Porcentaje
ocupado: " + Math.round(((cupoTotal - paseadorActual.cupoActual) / cupoTotal)
* 100) + "%";

    break;
}
}
}
}

// Esta función se encarga de mostrar las contrataciones del paseador logueado
en una tabla HTML.

function mostrarContratacionesDelPaseador() {
    let tablaHTML = "";
    if (logueado && paseadorActual !== null) {
        for (let i = 0; i < sistema.contrataciones.length; i++) {
            const unaContratacion = sistema.contrataciones[i];

            // Mostrar solo contrataciones aprobadas del paseador actual
            if (unaContratacion.estado === "Aprobada" &&
Number(unaContratacion.paseador) === paseadorActual.numeroIdentificador) {
                tablaHTML += `<tr>
                    <td>${unaContratacion.nombreperro}</td>
                    <td>${unaContratacion.sizeperro}</td>
                    <td>${unaContratacion.cliente}
                </tr>`;
            }
        }
    }

    if (tablaHTML === "") {

```

```

        tablaHTML = `<tr><td>No tenés perros asignados aún.</td></tr>`;
    }
    document.querySelector("#tblPerrosAsignados").innerHTML = tablaHTML;
}
}

// Esta función se encarga de cerrar la sesión del cliente logueado, limpiando
los campos de entrada y el aviso.

document.querySelector("#btnCerrarSesionCliente").addEventListener("click",
cerrarSesionCliente);

function cerrarSesionCliente() {
    // Limpiar todos los campos del formulario
    document.querySelector("#txtUsuarioCliente").value = "";
    document.querySelector("#txtContraCliente").value = "";
    document.querySelector("#txtNombrePerroCliente").value = "";
    document.querySelector("#txtSizePerroCliente").value = "";
    document.querySelector("#pAvisosCliente").innerHTML = "";
    clienteLogueado = null;
    alert("Se ha cerrado la sesión correctamente");
    tipoUsuario = "inicio";
    cambiarSeccion("inicio");
}

// Esta función se encarga de cerrar la sesión del paseador logueado,
limpiando los campos de entrada y el aviso.

document.querySelector("#btnCerrarSesionPaseador").addEventListener("click",
cerrarSesionPaseador);

function cerrarSesionPaseador() {
    logeado = false;
    paseadorActual = null;
    // Limpiar campos del formulario
    document.querySelector("#txtUsuarioPaseador").value = "";
    document.querySelector("#txtContraPaseador").value = "";
    alert("Se ha cerrado la sesión correctamente");
    tipoUsuario = "inicio";
    cambiarSeccion("inicio");
}

let clienteLogueado = null;

```

```

// Esta función se encarga de iniciar sesión como cliente, verificando el
usuario y la contraseña ingresados.

document.querySelector("#btnIniciarSesionCliente").addEventListener("click",
iniciarSesionCliente);

function iniciarSesionCliente() {
    let usuarioCliente = document.querySelector("#txtUsuarioCliente").value;
    let contraCliente = document.querySelector("#txtContraCliente").value;
    let cliente = obtenerElemento(sistema.cliente, "usuario", usuarioCliente); //
    Buscar cliente por usuario

    if (!usuarioCliente || !contraCliente) {
        document.querySelector("#pAvisosCliente").innerHTML = "Por favor, complete
    ambos campos.";
        return;
    }

    // Verificar credenciales
    if (cliente && cliente.usuario === usuarioCliente && cliente.contra ===
    contraCliente) {
        document.querySelector("#pSesionIniciadaCliente").innerHTML =
            `Sesión iniciada correctamente ${cliente.usuario}`;
        clienteLogueado = cliente;
        tipoUsuario = "cliente";
        cambiarSeccion("seccionClienteLogueado");
    } else {
        document.querySelector("#pAvisosCliente").innerHTML = "Por favor, ingrese
    un usuario y contraseña válidos";
    }
}

// Registra nuevos clientes validando que no existan y que la contraseña
cumpla requisitos
document.querySelector("#btnRegistrarCliente").addEventListener("click",
registrarCliente);

function registrarCliente() {
    let usuario = document.querySelector("#txtUsuarioClienteReg").value;
    let nombrePerro = document.querySelector("#txtNombrePerroCliente").value;
    let sizePerro = document.querySelector("#txtSizePerroCliente").value;
    let contra;
    let usuarioYaExiste = false;

```

```

// Verificar si el usuario ya existe
for (let i = 0; i < sistema.cliente.length; i++) {
    if (sistema.cliente[i].usuario.toLowerCase() === usuario.toLowerCase()) {
        usuarioYaExiste = true;
        break;
    }
}

if (usuarioYaExiste) {
    document.querySelector("#pAvisosClienteReg").innerHTML = "El usuario ya
existe. Por favor, elija otro.";

} else {
    // Variables para validar la contraseña
    let minimoDeCaracteres = "No cumple";
    let condicionDeMayuscula = "No cumple";
    let condicionDeMinuscula = "No cumple";
    let condicionDeNumero = "No cumple";
    let condicionDeAusenciaDeEspacios = "Cumple";
    contra = "";
    document.querySelector("#pAvisosClienteReg").innerHTML = "";
    contra = document.querySelector("#txtContraClienteReg").value;

    // Validar longitud mínima de la contraseña
    if (contra.length >= 5) {
        minimoDeCaracteres = "Cumple";
    }

    // Verificar espacios
    for (let i = 0; i < contra.length; i++) {
        if (contra.charAt(i) === " ") {
            condicionDeAusenciaDeEspacios = "No cumple";
        }
    }

    // Validar mayúsculas, minúsculas y números
    for (let i = 0; i < contra.length; i++) {
        let caracter = contra.charAt(i);

        if (!isNaN(caracter) && i !== 0) { // Número que no esté en primera
posición
            condicionDeNumero = "Cumple";
        }

        if (caracter === caracter.toUpperCase() && caracter !==
caracter.toLowerCase()) {

```

```

        condicionDeMayuscula = "Cumple";
    }

    if (caracter === caracter.toLowerCase() && caracter !==
caracter.toUpperCase()) {
        condicionDeMinuscula = "Cumple";
    }
}
}

// Verificar si todas las condiciones se cumplen
if (minimoDeCaracteres === "Cumple" && condicionDeMayuscula === "Cumple" &&
condicionDeMinuscula === "Cumple" && condicionDeAusenciaDeEspacios ===
"Cumple") {
    if (usuario && contra && nombrePerro && sizePerro) {
        let nuevoCliente = new Cliente(proximoID++, usuario, contra,
nombrePerro, sizePerro);
        sistema.cliente.push(nuevoCliente);
        alert("Cliente registrado correctamente: " + nuevoCliente.usuario);
        console.log(nuevoCliente.usuario)
        // Limpiar formulario
        document.querySelector("#txtUsuarioClienteReg").value = "";
        document.querySelector("#txtContraClienteReg").value = "";
        document.querySelector("#txtNombrePerroCliente").value = "";
        document.querySelector("#txtSizePerroCliente").value = "";
    }
    document.querySelector("#pAvisosClienteReg").innerHTML =
        "Su contraseña es apta para nuestro sistema y fue validada
correctamente." +
        "<br><br>Su contraseña fue: " + contra;
} else {
    // Mostrar qué condiciones no se cumplen
    document.querySelector("#pAvisosClienteReg").innerHTML =
        "Su contraseña aún no es apta para nuestro sistema, revise los
siguientes puntos: <br><br>" +
        "Minimo de 5 caracteres: " + minimoDeCaracteres + "<br>" +
        "Tener al menos una mayúscula (Usuario y Contraseña): " +
condicionDeMayuscula + "<br>" +
        "Tener al menos una minúscula (Usuario y Contraseña): " +
condicionDeMinuscula + "<br>" +
        "No tener espacios: " + condicionDeAusenciaDeEspacios + "<br>" +
        "Tener un número en algún lugar que no sea el primero: " +
condicionDeNumero + "<br><br>" +
        "Su contraseña fue: " + contra;
}

```

```

    }

    if (!usuario || !contra || !nombrePerro || !sizePerro) {
        document.querySelector("#pAvisosCliente").innerHTML =
            "Por favor, complete todos los campos.";
    }
}
}

// Esta función se encarga de iniciar sesión como paseador, verificando el
usuario y la contraseña ingresados.

document.querySelector("#btnIniciarSesionPaseador").addEventListener("click",
iniciarSesionPaseador);

function iniciarSesionPaseador() {
    let usuarioPaseador = document.querySelector("#txtUsuarioPaseador").value;
    let contraPaseador = document.querySelector("#txtContraPaseador").value;
    let paseador = obtenerElemento(sistema.paseador, "usuario", usuarioPaseador);
    // Buscar paseador por usuario

    if (paseador && usuarioPaseador === paseador.usuario && contraPaseador ===
paseador.contra) {
        logeado = true;
        paseadorActual = paseador;
        // Inicializar variables de tipos de perro
        paseadorActual.perroGrande = false;
        paseadorActual.perroChico = false;
        tipoUsuario = "paseador";
        cambiarSeccion("seccionPaseadorLogueado");

        document.querySelector("#pSesionIniciadaPaseador").innerHTML = "Sesión
iniciada correctamente " + paseador.usuario;
        validarProcesarContratacion(); // Actualizar cupos según contrataciones
existentes
    } else {
        logeado = false;
        paseadorActual = null;
        document.querySelector("#pAvisosPaseador").innerHTML = "Por favor, ingrese
un usuario y contraseña válidos";
    }
}
}

```



```

// Función compleja que actualiza los cupos de todos los paseadores según las
contrataciones aprobadas
// También marca si el paseador logueado ya tiene perros grandes o chicos
function validarProcesarContratacion() {

    paseadorActual.perroGrande = false;
    paseadorActual.perroChico = false;

    for (let i = 0; i < sistema.contrataciones.length; i++) {
        const ObjCont = sistema.contrataciones[i];
        if (ObjCont.estado === "Aprobada") {
            const ObjPas = obtenerElemento(sistema.paseador, "numeroIdentificador",
ObjCont.paseador);
            if (ObjPas) {
                // Descontar cupo según tamaño del perro
                if (ObjCont.sizeperro === "Grande") {
                    if (!ObjCont.cupoDescontado) {
                        ObjPas.cupoActual -= 4;
                    }
                    if (ObjPas === paseadorActual) {
                        paseadorActual.perroGrande = true;
                    }
                } else if (ObjCont.sizeperro === "Mediano") {
                    if (!ObjCont.cupoDescontado) {
                        ObjPas.cupoActual -= 2;
                    }
                } else if (ObjCont.sizeperro === "Chico") {
                    if (!ObjCont.cupoDescontado) {
                        ObjPas.cupoActual -= 1;
                    }
                }

                if (ObjPas === paseadorActual) {
                    paseadorActual.perroChico = true;
                }
            }
            ObjCont.cupoDescontado = true;
        }
    }
}

let botones = document.querySelectorAll(".boton");

```

```

for (let i = 0; i < botones.length; i++) {
  const botonHTML = botones[i];
  botonHTML.addEventListener("click", mostrarSeccion);
}

// Esta función se encarga de mostrar la sección correspondiente al botón que
se ha pulsado.

function mostrarSeccion() {
  let idBoton = this.getAttribute("id");
  let idSeccion = idBoton.charAt(3).toLowerCase() + idBoton.substring(4); //
  Convertir ID del botón a ID de sección
  cambiarSeccion(idSeccion);
}

// Esta función se encarga de mostrar los botones correspondientes al tipo de
usuario que ha iniciado sesión.

function mostrarBotones(tipoUsuario) {
  // Ocultar todos los botones primero
  let botones = document.querySelectorAll(".boton");
  for (let i = 0; i < botones.length; i++) {
    const botonHTML = botones[i];
    botonHTML.style.display = "none";
  }

  // Mostrar solo los botones del tipo de usuario actual
  let botonesMostrar = document.querySelectorAll("." + tipoUsuario);
  for (let i = 0; i < botonesMostrar.length; i++) {
    const botonHTML = botonesMostrar[i];
    botonHTML.style.display = "block";
  }
}

// Muestra paseadores disponibles que tienen cupo suficiente para el perro del
cliente logueado
function tablaPaseadoresDisponibles() {
  let tablaHTML = "";

  for (let i = 0; i < sistema.paseador.length; i++) {
    const unPaseador = sistema.paseador[i];
    let costoCupo = 0

    if (clienteLogueado.sizeperro === "Grande") {

```

```

    costoCupo = 4;
  } else if (clienteLogueado.sizeperro === "Mediano") {
    costoCupo = 2;
  } else if (clienteLogueado.sizeperro === "Chico") {
    costoCupo = 1;
  }

  // Solo mostrar paseadores que tengan cupo disponible
  if (unPaseador.cupoActual >= costoCupo) {
    console.log(costoCupo)
    tablaHTML += `<tr>
      <td>${unPaseador.nombre}</td>
      <td>${unPaseador.cupoActual}</td>
      <td><input type="button" value="Contratar Paseador"
class="botonContratar" data-id="${unPaseador.numeroIdentificador}"></td>
    </tr>`;
  }
}

if (tablaHTML === "") {
  tablaHTML = `<tr><td>No hay paseadores disponibles en este
momento.</td></tr>`;
}

document.querySelector("#tblPaseadoresDisponibles").innerHTML = tablaHTML;

let botonesContratar = document.querySelectorAll(".botonContratar");
for (let i = 0; i < botonesContratar.length; i++) {
  const boton = botonesContratar[i];
  boton.addEventListener("click", contratarPaseador);
}

let botonesCancelar = document.querySelectorAll(".botonCancelar");
for (let i = 0; i < botonesCancelar.length; i++) {
  const botonC = botonesCancelar[i];
  botonC.addEventListener("click", cancelarContratacionDelPaseador)
  if (Number(botonC.getAttribute("data-id")) !== idPaseadorSeleccionado) {
    botonC.hidden = true;
  }
}
}
}

//Muestra todos los paseadores.

```

```

function tablaPaseadoresTotales() {
  let tablaHTML = "";

  for (let i = 0; i < sistema.paseador.length; i++) {
    const unPaseador = sistema.paseador[i];
    tablaHTML += `<tr>
      <td>${unPaseador.nombre}</td>
      <td>${unPaseador.cupoActual}</td>
    </tr>`;
  }

  if (tablaHTML === "") {
    tablaHTML = `<tr><td>No hay paseadores.</td></tr>`;
  }

  document.querySelector("#tblPaseadoresTotales").innerHTML = tablaHTML;
}

//Es la encargada de hacer el proceso de contratar a un paseador si cumple con
los requerimientos.

function contratarPaseador() {
  let idPaseadorSeleccionado = Number(this.getAttribute("data-id"));
  let costoCupo = 0;
  let existeContratacion = false

  for (let i = 0; i < sistema.contrataciones.length; i++) {
    let contratacion = sistema.contrataciones[i];
    if (contratacion.cliente === clienteLogueado.usuario &&
(contratacion.estado === "Pendiente" || contratacion.estado === "Aprobada")) {
      existeContratacion = true;
    }
  }

  if (existeContratacion) {
    alert("Ya tienes una contratacion pendiente.");
  }

  if (clienteLogueado.sizeperro === "Grande") {
    costoCupo = 4;
  } else if (clienteLogueado.sizeperro === "Mediano") {
    costoCupo = 2;
  } else if (clienteLogueado.sizeperro === "Chico") {
    costoCupo = 1;
  }
}

```

```

    let paseadorSeleccionado = obtenerElemento(sistema.paseador,
"numeroIdentificador", idPaseadorSeleccionado);

    if (paseadorSeleccionado && paseadorSeleccionado.cupoActual >= costoCupo &&
!existeContratacion) {
        let nuevaContratacion = new Contratacion(proximoID++,
clienteLogueado.usuario, idPaseadorSeleccionado, "Pendiente",
clienteLogueado.nombreperro, clienteLogueado.sizeperro);
        sistema.contrataciones.push(nuevaContratacion);

        document.querySelector("#pAvisosCliente").innerHTML = `Solicitud de
contratación enviada para ${paseadorSeleccionado.nombre}. Estado: Pendiente de
aprobación.`;

        tablaPaseadoresDisponibles();
    } else {
        document.querySelector("#pAvisosCliente").innerHTML =
            `No se pudo contratar al paseador. Cupo insuficiente para perros de
tamaño ${clienteLogueado.sizeperro}.`;
    }
}

// Tabla donde esta la interfaz para poder cancelar una contratacion, que solo
va a aparecer si esta en pendiente.
function tablaCancelarContratacion() {

    if (clienteLogueado) {
        let tablaHTML = "";

        for (let i = 0; i < sistema.contrataciones.length; i++) {
            const unObjetoContratacion = sistema.contrataciones[i];
            let paseadorObj = obtenerElemento(sistema.paseador,
"numeroIdentificador", unObjetoContratacion.paseador);
            let unPaseador;

            if (paseadorObj) {
                unPaseador = paseadorObj.nombre;
            }

            if (unObjetoContratacion.cliente === clienteLogueado.usuario &&
unObjetoContratacion.estado === "Pendiente") {
                tablaHTML += `<tr>
                <td>${unPaseador}</td>

```

```

        <td><input type="button" value="Cancelar Contratación"
class="botonCancelar"
data-idCancelar="{unObjetoContratacion.numeroIdentificador}"></td>
    </tr>`;
    }
    document.querySelector("#tblCancelarContratacion").innerHTML = tablaHTML;

    let botonesCancelar = document.querySelectorAll(".botonCancelar");
    for (let i = 0; i < botonesCancelar.length; i++) {
        botonesCancelar[i].addEventListener("click", cancelarContratacion);
    }
}
}
}

// Es la encargada de que el cliente puede cancelar su contratacion si no la
Acepto el paseador.
function cancelarContratacion() {
    let idCancelarContratacion = Number(this.getAttribute("data-idCancelar"));

    for (let i = 0; i < sistema.contrataciones.length; i++) {
        const contratacion = sistema.contrataciones[i];
        if (contratacion.numeroIdentificador === idCancelarContratacion &&
contratacion.estado === "Pendiente") {
            contratacion.estado = "Cancelada";
            document.querySelector("#pAvisosCliente").innerHTML = "Contratación
cancelada correctamente.";
            break;
        }
    }
    tablaCancelarContratacion();
}
mostrarBotones("inicio");

// Esta función se encarga de cambiar la sección visible en la interfaz,
ocultando las demás secciones y mostrando la nueva sección.

function cambiarSeccion(seccionNueva) {
    ocultarSecciones();
    const seccion = document.querySelector("#" + seccionNueva);
    seccion.style.display = "block";

    switch (seccionNueva) {
        case "inicio":

```

```

        mostrarBotones("inicio");
        break;
    case "cliente":
    case "seccionCliente":
    case "seccionClienteIniciarSesion":
    case "seccionClienteRegistrarse":
    case "seccionClienteLogueado":
    case "seccionCalificarPaseador":
        mostrarBotones("cliente");
        break;
    case "seccionContratarPaseador":
        mostrarBotones("cliente");
        tablaPaseadoresDisponibles();
        break;
    case "seccionTodoPaseadores":
        tablaPaseadoresTotales();
        break;
    case "seccionCancelarContratacion":
        mostrarBotones("cliente");
        tablaCancelarContratacion();
        break;
    case "paseador":
    case "seccionPaseador":
    case "seccionPaseadorIniciarSesion":
    case "seccionPaseadorLogueado":
        mostrarBotones("paseador");
        break;
    case "seccionContrataciones":
        mostrarBotones("paseador");
        tablaContratacionesPendientes();
        verificarBotonesDeshabilitados();
        break;
    case "seccionPerrosAsignados":
        mostrarBotones("paseador");
        mostrarContratacionesDelPaseador();
        break;
    default:
        break;
}
}

// Esta función se encarga de ocultar todas las secciones de la interfaz,
excepto la sección de inicio.
function ocultarSecciones() {
    let secciones = document.querySelectorAll(".seccion, .inicio");

```

```

for (let i = 0; i < secciones.length; i++) {
    const seccionHTML = secciones[i];
    seccionHTML.style.display = "none";
}
}

// Variable global para saber cuántas estrellas eligió el usuario
let estrellasSeleccionadas = 0;

// Configuramos el sistema de estrellas interactivo de (1 a 5 estrellas)
for (let i = 1; i <= 5; i++) {
    let estrella = document.querySelector("#Estrella" + i);

    estrella.addEventListener("click", funcionEstrella);

    // Función que maneja el click en cada estrella
    function funcionEstrella() {
        estrellasSeleccionadas = i; // Guardamos cuántas estrellas seleccionó
        for (let j = 1; j <= 5; j++) {
            let est = document.querySelector("#Estrella" + j);
            est.value = j <= i ? "★" : "☆";
        }
    }
}

// Conectamos el botón de calificar con su función
document.querySelector("#btnCalificarPaseador").addEventListener("click",
funcionComentar);

// Función principal que procesa la calificación del paseador
function funcionComentar() {
    let comentario = document.querySelector("#txtCalificarPaseador").value;
    let mensaje = "";

    // Chequea que haya estrellas y comentario
    if (estrellasSeleccionadas === 0 && comentario === "") {
        document.querySelector("#pRevisionPaseador").innerHTML = `Seleccione una
cantidad de estrellas. <br> Escriba un comentario.`;
    } else {

        // Buscamos si el cliente logueado tiene alguna contratación aprobada
        let contratacionCliente = null;
        for (let i = 0; i < sistema.contrataciones.length; i++) {
            const contratacion = sistema.contrataciones[i];

```



```

        if (contratacion.cliente === clienteLogueado.usuario &&
contratacion.estado === "Aprobada") {
            contratacionCliente = contratacion;
            break;
        }
    }

    // Si no tiene contrataciones aprobadas, no puede calificar
    if (!contratacionCliente) {
        document.querySelector("#pRevisionPaseador").innerHTML = "No tienes
ningún paseador al cual calificar (Para poder calificar a un paseador debes de
tener una contratación aceptada por alguno de ellos).";
    } else {

        // Obtenemos los datos del paseador que fue contratado usando una función
auxiliar (La de obtenerElementos)
        let paseadorAsignado = obtenerElemento(sistema.paseador,
"numeroIdentificador", contratacionCliente.paseador);

        if (!paseadorAsignado) {
            document.querySelector("#pRevisionPaseador").innerHTML = "Error al
obtener el paseador asignado.";
        } else {

            document.querySelector("#nombreDelPaseadorCalificado").innerHTML = ""
            document.querySelector("#nombreDelPaseadorCalificado").innerHTML = "Tu
Paseador: " + paseadorAsignado.nombre;

            // Aca convertimos las estrellas para que sean estrellitas
            let estrellas = "";
            for (let i = 0; i < estrellasSeleccionadas; i++) {
                estrellas += "★";
            }

            mensaje = `
<br><br>
👤 Cliente: ${clienteLogueado.usuario}<br>
🐶 Paseador calificado: ${paseadorAsignado.nombre}<br>
🗨 Comentario: ${comentario}<br>
★ Calificación: ${estrellas}<br>
<br><br>
`;

            // Agregamos la calificación a la tabla de revisiones
            document.querySelector("#tblRevisionPaseador").innerHTML += mensaje;

```

```
document.querySelector("#pRevisionPaseador").innerHTML = "";

// Limpiamos el formulario después de enviar la calificación
document.querySelector("#txtCalificarPaseador").value = "";
estrellasSeleccionadas = 0;
for (let i = 1; i <= 5; i++) {
    document.querySelector("#Estrella" + i).value = "☆";
}
}
}
}
```

Universidad ORT Uruguay
Facultad de Ingeniería
Escuela de Tecnología

OBLIGATORIO PROGRAMACIÓN 1
DOCUMENTO DE ANÁLISIS



Manuel Bercianos – 309770



Rodrigo Fernández - 365329

M1B

Profesor: Santiago Fagnoni

Analista en tecnologías de la información.

[26-05-2025]

Índice

Análisis del Sistema 3

- Descripción general del problema a resolver
- Tipos de usuario del sistema
- Listado de funcionalidades

Detalle de Funcionalidades 8

- F01 - Registrarse en el sistema
- F02 - Seleccionar y Contratar paseador
- F03 - Cancelar contratación
- F04 - Visualizar listado de paseadores
- F05 - Calificar paseador
- F06 - Visualizar contrataciones pendientes

- F07 - Visualizar perros asignados
- F08 - Iniciar sesión
- F09 - Cerrar sesión

Casos de Prueba 25

- Casos de prueba por funcionalidad
- Datos de prueba precargados

Información Precargada 50

- Datos de clientes
- Datos de paseadores
- Contrataciones de ejemplo

2. Descripción general del problema a resolver

El sistema desarrollado es una aplicación web para conectar paseadores de perros con clientes que necesitan este servicio. La aplicación maneja dos tipos de usuarios: clientes (dueños de perros) y paseadores (son los prestadores del servicio).

2. Tipos de usuario del sistema

Cliente: Usuarios que contratan los servicios de paseadores para sus perros.

Paseador: Proveedor que realizan los paseos diarios con los perros a su cargo

3. Listado de funcionalidades

Funcionalidades del Cliente:

- **F01 - Registrarse en el sistema:** Permite a un nuevo cliente crear una cuenta en la aplicación.
- **F02 - Seleccionar y Contratar paseador:** Permite al cliente elegir y contratar un paseador de la lista de paseadores disponibles.
- **F03 - Cancelar contratación:** Permite al cliente cancelar una contratación pendiente.
- **F04 - Visualizar listado de paseadores:** Permite al cliente ver un listado de todos los paseadores y la cantidad de cupos que tienen disponibles
- **F05 - Calificar paseador:** Permite al cliente calificar y dejar comentarios sobre el servicio del paseador.

Funcionalidades del Paseador:

- **F06 - Visualizar contrataciones pendientes:** Permite al paseador ver la lista de contrataciones que están pendientes de aprobación.
- **F07 - Visualizar perros asignados:** Permite al paseador ver una lista de todos los perros que tiene asignados, junto con información de cupos.

Funcionalidades Compartidas:

- **F08 - Iniciar sesión:** Permite a un paseador/cliente acceder a la aplicación.
- **F09 - Cerrar sesión:** Permite al paseador/cliente cerrar su sesión en la aplicación.

Funcionalidades Auxiliares:

- **F10 - obtenerElemento(arrBuscar, prop, valor)** Busca un objeto específico en un array basándose en una propiedad y su valor.
- **F11 - verificarBotonesDeshabilitados()** Valida y deshabilita botones de "Procesar Contratación" cuando no se pueden aprobar por falta de cupo o incompatibilidad de tamaños.
- **F12 - validarProcesarContratacion()** Configura el estado inicial del paseador al iniciar sesión, resetea propiedades y descuenta cupos según contrataciones aprobadas.
- **F13 - mostrarBotones(tipoUsuario)** Muestra solo los botones correspondientes al tipo de usuario logueado (inicio, cliente o paseador).
- **F14 - ocultarSecciones()** Oculta todas las secciones antes de mostrar una nueva, preparando la interfaz para el cambio de contenido.
- **F15 - funcionEstrella()** Maneja la selección de estrellas en el sistema de calificación, actualiza visualmente las estrellas seleccionadas.
- **F16 - procesarContratacion()** Procesa las contrataciones pendientes de los clientes. Verifica si el paseador tiene cupo suficiente y si el tamaño del perro es compatible con los perros ya asignados.
- **F17 - mostrarContratacionesDelPaseador()** Muestra en una tabla HTML todas las contrataciones aprobadas asignadas al paseador logueado.
- **F18 - funcionComentar()** Procesa la calificación y comentario del cliente hacia quien fue su paseador asignado.

Detalle de Funcionalidades

En la parte siguiente, se presenta el detalle de cada una de las funcionalidades a resolver en el obligatorio. Las funcionalidades están ordenadas por tipo de usuario.

F01 - Registrarse en el sistema.

- **Acceso: Cliente**
- **Descripción:**
 - Esta funcionalidad permite a un nuevo usuario crear una cuenta en la aplicación como cliente. Al registrarse, el usuario deberá proporcionar la información necesaria para utilizar los servicios, incluyendo sus datos personales y los de su perro como su tamaño

Interfaz de usuario:

The image shows a user registration form titled "REGISTRARSE - CLIENTE". The form is contained within a light gray rounded rectangle. It features four input fields: a text field for "INGRESE SU USUARIO:", a text field for "CONTRASEÑA:", a text field for "NOMBRE DEL PERRO:", and a dropdown menu for "TAMAÑO DE SU PERRO" with "Pequeño" selected. Below the input fields are two buttons: a dark gray button labeled "REGISTRAR CLIENTE" and a blue button labeled "VOLVER".

REGISTRARSE - CLIENTE

INGRESE SU USUARIO:

CONTRASEÑA:

NOMBRE DEL PERRO:

TAMAÑO DE SU PERRO

Pequeño

REGISTRAR CLIENTE

VOLVER

- **Validaciones:**

- Nombre de usuario: Obligatorio
 - Único en el sistema
 - Mensaje de error: "El nombre de usuario ya existe. Por favor, elija otro."
 - Longitud mínima: 5 caracteres (ejemplo)
 - Mensaje de error: "El nombre de usuario debe tener al menos 5 caracteres."
- Contraseña: Obligatoria
- Longitud mínima: 5 caracteres
 - Mensaje de error: "La contraseña debe tener al menos 5 caracteres."
 - Debe contener al menos una letra mayúscula
 - Mensaje de error: "La contraseña debe contener al menos una letra mayúscula."
 - Debe contener al menos una letra minúscula
 - Mensaje de error: "La contraseña debe contener al menos una letra minúscula."
 - Debe contener al menos un número
 - Mensaje de error: "La contraseña debe contener al menos un número."
 - Nombre del perro: Obligatorio
 - Tamaño del perro: Obligatorio
 - Debe ser uno de los valores permitidos, según lo indica del combo desplegable: ("Grande", "Mediano", "Chico")

F02 - Seleccionar y Contratar paseador

- **Acceso:** Cliente
- **Descripción:** Esta parte de la funcionalidad permite al cliente elegir un paseador de la lista de paseadores disponibles para contratar sus servicios, la lista de paseadores mostrada al cliente debe tener en cuenta la disponibilidad del paseador (cupos) y por otro lado, las restricciones de tamaño de los perros (un paseador no puede pasear perros grandes y chicos al mismo tiempo).
 - Lista/Combo desplegable de paseadores:
 - Muestra el nombre de cada paseador disponible.
 - Puede incluir información adicional como la cantidad de perros que pasea actualmente o su calificación (si se implementa).
 - Mensaje informativo (si no hay paseadores disponibles): "No hay paseadores disponibles para el tamaño de su perro en este momento."
- **Interfaz de usuario:**

CONTRATAR PASEADOR

NOMBRE DEL PASEADOR	CUPO DISPONIBLE	GESTION
Jorge	10	CONTRATAR PASEADOR
Rodrigo	10	CONTRATAR PASEADOR
Manuel	15	CONTRATAR PASEADOR
Ana	8	CONTRATAR PASEADOR
Lucia	12	CONTRATAR PASEADOR

VOLVER

- **Validaciones:**
 - La lista de paseadores debe filtrarse automáticamente según:
 - Disponibilidad de cupo: El paseador debe tener cupo suficiente para el tamaño del perro del cliente.

- Tamaño del perro: El paseador no debe tener asignados perros de tamaño incompatible (si el cliente tiene un perro chico, solo se muestran paseadores sin perros grandes asignados, y viceversa).
- Si el cliente intenta realizar una contratación sin seleccionar un paseador:
 - Mensaje de error: "Por favor, seleccione un paseador."
- Verificar que se haya seleccionado un paseador (esto ya se validó en F04, pero se puede volver a verificar).
 - Mensaje de error: "Por favor, seleccione un paseador."
- Verificar que el cliente no tenga ya una contratación pendiente.
 - Mensaje de error: "Ya tiene una contratación pendiente. Por favor, espere a que sea aprobada o rechazada, o cancele la contratación actual."
- Mensajes:
 - Mensaje de éxito: "Contratación realizada. Esperando aprobación del paseador."
 - Mensaje de error: "Ya tiene una contratación pendiente. Por favor, espere a que sea aprobada o rechazada, o cancele la contratación actual."

F03 - Cancelar contratación

- **Acceso:** Cliente.
- **Descripción:** Permite al cliente cancelar una contratación pendiente con un paseador.
 - Botón: "Cancelar Contratación" (Visible sólo si hay una contratación pendiente)
 - Mensaje de confirmación: "Contratación cancelada."
 - Mensaje de error: "No hay contratación pendiente." (Si se intenta cancelar sin tener una)
- **Validaciones:**
 - Verificar si existe una contratación pendiente antes de permitir la cancelación.

- **Interfaz de usuario:**

CANCELAR CONTRATACIONES

PASEADOR PENDIENTE	ACCIÓN
Rodrigo	CANCELAR CONTRATACIÓN

VOLVER A OPCIONES

F04 - Visualizar listado de paseadores

- **Acceso:** Cliente
- **Descripción:** Permite al cliente ver un listado de todos los paseadores registrados en la aplicación, junto con información relevante sobre cada uno, como el cupo disponible que tienen actualmente. Esto ayuda al cliente a tener una visión general de los paseadores disponibles.
 - Listado de Paseadores:
 - Muestra el nombre de cada paseador.
 - Muestra la cantidad de perros que tiene asignados cada paseador.
- **Interfaz de usuario:**

TODOS LOS PASEADORES

NOMBRE DEL PASEADOR	CUPO DISPONIBLE
Jorge	10
Rodrigo	10
Manuel	15
Ana	8
Lucia	12

VOLVER

F5 - Calificar paseador

- **Acceso:** Cliente
- **Descripción:** Permite al cliente asignar una calificación (por ejemplo, de 1 a 5 estrellas) y dejar un comentario opcional sobre el servicio proporcionado por el paseador. Esta retroalimentación ayuda a otros clientes a tomar decisiones y a la administración a mantener un registro de la calidad del servicio.
- - **Formulario de Calificación:**
 - Sistema de 5 estrellas clickeables y área de texto para comentarios.
 - Área de texto opcional para el comentario.
 - Botón "Enviar Calificación".
 - **Mensajes:**
 - Mensaje de éxito: Muestra la calificación enviada con formato estructurado
 - Mensaje de error: "Seleccione una cantidad de estrellas. Escriba un comentario."
 - Mensaje de error: "No tienes ningún paseador al cual calificar (Para poder calificar a un paseador debes de tener una contratación aceptada por alguno de ellos)."
 -
- **Validaciones:**
 - Verificar que se haya seleccionado una calificación.
 - Mensaje de error: "Por favor, selecciona una calificación."
 - La calificación debe estar dentro del rango permitido (por ejemplo, de 1 a 5).
 - Mensaje de error: "La calificación debe estar entre 1 y 5." (Si se implementa la validación del rango en el cliente)."
- **Validaciones:**
 - Verificar que se haya seleccionado una calificación (1-5 estrellas)
 - Verificar que el cliente tenga una contratación aprobada
 - El comentario es opcional pero si no se proporciona junto con las estrellas, se muestra error

Interfaz de usuario:

The image shows a web form titled "CALIFICAR PASEADOR". It includes a label "Tu Paseador: Rodrigo", a "RESEÑA:" label with a text input field, a five-star rating system with five dark buttons each containing a white star, an "ENVIAR CALIFICACIÓN" button, and a "VOLVER A OPCIONES" button. Below these is a summary box containing: "Cliente: cliente1", "Paseador calificado: Rodrigo", "Comentario: ¡Excelente Servicio! Gracias Rodrigo", and "Calificación: ★★★★★".

F6 - Visualizar contrataciones pendientes

Acceso: Paseador.

- **Descripción:** Esta funcionalidad permite al paseador ver una lista de todas las contrataciones que se encuentran en estado “pendiente”. Esto le permite al paseador gestionar las solicitudes de servicio de los clientes.
-
- Listado de contrataciones pendientes.
- Muestra información relevante de cada contratación pendiente como:
 - Nombre del cliente
 - Nombre del perro
 - Tamaño del perro
- Para cada contratación, incluye botón:

Aprobar la contratación

(si ya se llegó al cupo automáticamente el sistema rechaza el resto de las solicitudes, consultas de todas formas)

- Mensaje
 - Éxito
 - Error
- Validaciones

Cupo: Rechazar si no hay cupo suficiente (Grande 4, Mediano 2, Chico 1)

Tamaño: Rechazar perros chicos con grandes asignados y viceversa.

Interfaz de Usuario:

CONTRATACIONES PENDIENTES		
NOMBRE DEL PERRO	USUARIO	ACCIÓN
Fafa	Jose	PROCESAR CONTRATACION
Rex	Roberto	SIN CUPO DISPONIBLE
Tití	Simon	PROCESAR CONTRATACION
Mar	Claudia	PROCESAR CONTRATACION
Colo	Pablo	PROCESAR CONTRATACION
Beto	Josefina	PROCESAR CONTRATACION
Firu	Maite	SIN CUPO DISPONIBLE
Dulce	Diana	PROCESAR CONTRATACION
Teco	Julietta	PROCESAR CONTRATACION
severus	Valeria	PROCESAR CONTRATACION

VOLVER A OPCIONES

F7 - Visualizar perros asignados:

- **Acceso:** Paseador.
- **Descripción:** El paseador podrá ver un listado donde tenga información de los perros asignados como también los cupos que tiene en el momento.
 - Listado de perros asignados:
 - Número.
 - Tamaño.
 - Información General:
 - Cupos disponibles.
 - Cupos máximos.
 - Porcentaje de cupos asignados.

Si el paseador no tiene perros asignados entonces se le mostrará un mensaje que diga (“No hay perros asignados actualmente”).

Interfaz de usuario:

PERROS ASIGNADOS		
NOMBRE DEL PERRO	TAMAÑO DEL PERRO	CLIENTE
Sarha	Grande	Tomas
Thor	Mediano	Mario
VOLVER A OPCIONES		

F08 - Iniciar sesión

- **Acceso:** Paseador/Cliente.
- **Descripción:** Esta funcionalidad permite a los usuarios acceder a la aplicación mediante, (“nombre de usuario” y “contraseña”).
 - Formulario de Inicio de Sesión:
 - Campos:
 - Nombre de usuario (text)
 - Contraseña (password)
 - Botón:
 - "Iniciar Sesión"
 - Mensajes:

- Mensaje de éxito Cliente: "Sesión iniciada correctamente [nombre usuario]"
- Mensaje de éxito Paseador: "Sesión iniciada correctamente [nombre usuario]"
- Mensaje de error: "Por favor, ingrese un usuario y contraseña válidos"
- Mensaje de error: "Por favor, complete ambos campos."

Validaciones:

- Verificar que ambos campos estén completos
 - Verificar credenciales contra la base de datos
 - Redirigir al panel correspondiente según tipo de usuario
- **Interfaz de usuario:**

INICIAR SESIÓN - CLIENTE

INGRESE SU USUARIO:

CONTRASEÑA:

INICIAR SESIÓN

VOLVER

INICIAR SESIÓN - PASEADOR

INGRESE SU USUARIO:

CONTRASEÑA:

INICIAR SESIÓN

VOLVER

F9 - Cerrar sesión

Acceso: Paseador/Cliente.

- **Descripción:** Esta funcionalidad permite al usuario cerrar su sesión actual en la aplicación y volver a la pantalla de inicio de sesión

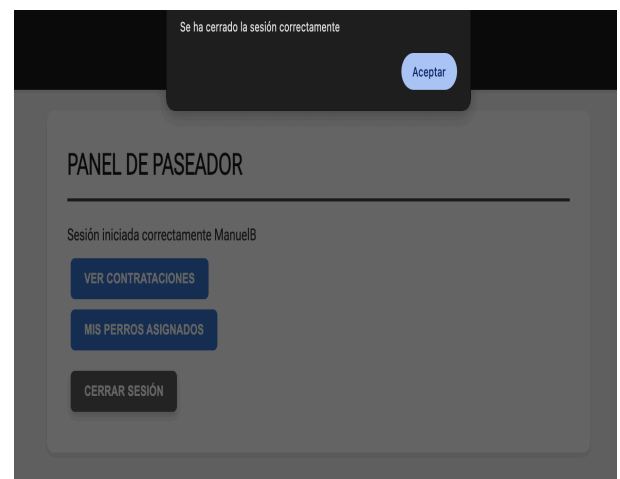
Funcionalidad:

- Limpia todos los campos de entrada

- Resetea las variables de sesión
- Muestra mensaje: "Se ha cerrado la sesión correctamente"
- Redirige a la pantalla de inicio

Botón: "Cerrar Sesión" (Este botón está presente en el panel principal de los dos usuarios)

- **Interfaz de Usuario:**



Funciones auxiliares:

F10 - ObtenerElemento(arrBuscar, prop, valor)

Funcionalidad: Buscar un objeto específico en un array, basándose en alguna propiedad y su valor.

Ejemplo de uso:

```
let cliente = obtenerElemento(sistema.cliente, "usuario", "Marcos");
```

F11 - verificarBotonesDeshabilitados()

Funcionalidad: Validar y deshabilitar botones de “Procesar Contratación” cuando no se pueden aprobar

Qué hace:

- Revisa cada botón de procesamiento en la tabla de contrataciones pendientes
- Calcula si el paseador tiene cupo suficiente para cada tipo de perro
- Verifica las restricciones de compatibilidad (no mezclar perros grandes con chicos)

- Deshabilitar botones y cambia su texto a "Sin cupo disponible" cuando no se puede procesar
- Actualiza la información de cupos disponibles en pantalla

Validaciones que realiza:

- Perros grandes: necesitan 4 cupos y no pueden ir con perros chicos
- Perros medianos: necesitan 2 cupos (sin restricciones)
- Perros chicos: necesitan 1 cupo y no pueden ir con perros grandes

Apoya a: procesarContratacion() y tablaContratacionesPendientes()

F-12. validarProcesarContratacion()

Tipo: Función auxiliar de inicialización

Propósito: Configurar el estado inicial del paseador al iniciar sesión.

Qué hace:

- Resetea las propiedades perroGrande y perroChico del paseador actual
- Recorre todas las contrataciones aprobadas del sistema
- Descuenta cupos de todos los paseadores según sus contrataciones aprobadas
- Marca qué tipos de perros tiene asignados cada paseador
- Actualiza la propiedad cupoDescontado para evitar descuentos duplicados

Cálculo de cupos:

- Perro grande: -4 cupos
- Perro mediano: -2 cupos
- Perro chico: -1 cupo

F-13. MostrarBotones(tipoUsuario)

Tipo: Función auxiliar de interfaz

Propósito: Mostrar solo los botones correspondientes al tipo de usuario logueado.

Qué hace:

- Oculta todos los botones de navegación
- Muestra solo los botones que corresponden al tipo de usuario actual
- Utiliza clases CSS para identificar qué botones mostrar

Tipos de usuario:

- "inicio": Botones de la página principal
- "cliente": Botones del área de cliente
- "paseador": Botones del área de paseador

Apoya a: cambiarSeccion() y las funciones de inicio de sesión.

F14. ocultarSecciones()

Tipo: Función auxiliar de interfaz

Propósito: Ocultar todas las secciones antes de mostrar una nueva.

Qué hace:

- Selecciona todas las secciones con clase .seccion e .inicio
- Establece display: none en todas ellas
- Prepara la interfaz para mostrar la nueva sección

Apoya a: cambiarSeccion() para limpiar la pantalla antes de mostrar nueva contenido.

F15. funcionEstrella() (dentro del bucle de estrellas)

Tipo: Función auxiliar de interfaz

Propósito: Manejar la selección de estrellas en el sistema de calificación.

Qué hace:

- Captura qué estrella fue clickeada
- Actualiza la variable estrellasSeleccionadas
- Cambia visualmente las estrellas (★ para seleccionadas, ☆ para no seleccionadas)
- Actualiza todas las estrellas desde la 1 hasta la seleccionada

Apoya a: funcionComentar() para procesar la calificación del paseador.

Casos de Prueba

Casos de Prueba - F01 (Registrarse en el sistema)

Caso 1: Registro exitoso

- Usuario: "nuevoUsuario"
- Contraseña: "Abc123"
- Nombre perro: "Firulais"
- Tamaño perro: "Grande"
- Resultado esperado: "Cliente registrado correctamente: nuevoUsuario"

Caso 2: Usuario ya existe

- Usuario: "cliente1" (ya existe)
- Resultado esperado: "El usuario ya existe. Por favor, elija otro."

Caso 3: Contraseña inválida

- Usuario: "nuevoUsuario2"
- Contraseña: "abc" (no cumple requisitos)
- Resultado esperado: Mensaje detallando los requisitos no cumplidos

Casos de Prueba - F02/F03 (Seleccionar/Contratar paseador)

Caso 1: Contratación exitosa

- Cliente: "cliente1" (perro Grande)
- Paseador disponible: Jorge (cupó suficiente)
- Resultado esperado: "Solicitud de contratación enviada para Jorge. Estado: Pendiente de aprobación."

Caso 2: Sin cupo disponible

- Cliente con perro Grande, todos los paseadores sin cupo para perros grandes
- Resultado esperado: "No hay paseadores disponibles en este momento."

Caso 3: Contratación duplicada

- Cliente que ya tiene una contratación pendiente
- Resultado esperado: "Ya tienes una contratación pendiente."

Casos de Prueba - F04 (Cancelar contratación)

Caso 1: Cancelación exitosa

- Cliente con contratación pendiente
- Resultado esperado: "Contratación cancelada correctamente."

Caso 2: Sin contrataciones para cancelar

- Cliente sin contrataciones pendientes
- Resultado esperado: Tabla vacía

Casos de Prueba - F06 (Calificar paseador)

Caso 1: Calificación completa

- Cliente: con contratación aprobada
- Estrellas: 5

- Comentario: "Excelente servicio"
- Resultado esperado: Muestra calificación formateada

Caso 2: Sin contratación aprobada

- Cliente sin contrataciones aprobadas
- Resultado esperado: "No tienes ningún paseador al cual calificar..."

Casos de Prueba - F07 (Procesar contrataciones)

Caso 1: Aprobación exitosa

- Paseador: Jorge (cupos disponible)
- Contratación: Perro mediano (requiere 2 cupos)
- Resultado esperado: "La contratación ha sido aprobada. Debes pasar a buscar a [perro]..."

Caso 2: Sin cupo suficiente

- Paseador sin cupo suficiente
- Resultado esperado: Botón deshabilitado "Sin cupo disponible"

Casos de Prueba - F09 (Iniciar sesión)

Caso 1: Login cliente exitoso

- Usuario: "cliente1"
- Contraseña: "Aa123"
- Resultado esperado: "Sesión iniciada correctamente cliente1"

Caso 2: Login paseador exitoso

- Usuario: "JorgeB"
- Contraseña: "Jorge123"
- Resultado esperado: "Sesión iniciada correctamente JorgeB"

Caso 3: Credenciales incorrectas

- Usuario: "usuarioQueNoExiste"

- Contraseña: "contraseñaQueNoExiste"
- Resultado esperado: "Por favor, ingrese un usuario y contraseña válidos"

Código JavaScript:

```
// Clase cliente, se encarga de gestionar el apartado de clientes.
// Esta clase es la que se encarga de gestionar los clientes, sus datos y sus
perros.
// Además, se encarga de gestionar las contrataciones de paseadores.
let logeado = false;
let paseadorActual = null;
let proximoID = 31; // ID que aumenta de uno en uno para nuevos registros
let tipoUsuario = "inicio";

// Hacer que cuando alguien te acepte la contratación, el resto de paseadores
se bloquee, hacer que cuando el perro no sea compatible, el paseador se
elimine de la lista de contratar paseadores

class Cliente {
  constructor(idC, usuarioC, contraC, nombreperroC, sizeperroC) {
    this.id = idC;
    this.usuario = usuarioC;
    this.contra = contraC;
    this.nombreperro = nombreperroC;
    this.sizeperro = sizeperroC;
  }
}

// Clase paseador, se encarga de gestionar el apartado de paseadores.
// Esta clase es la que se encarga de gestionar los paseadores, sus datos y
sus cupos máximos.
// Además, se encarga de gestionar las contrataciones de paseadores.

class Paseador {
  constructor(numeroIP, nombreP, cupoMaxPA, cupoMAXP, usuarioP, contraP) {
    this.numeroIdentificador = numeroIP;
    this.nombre = nombreP;
    this.cupoActual = cupoMaxPA; // Cupo disponible actual
    this.cupoMax = cupoMAXP // Cupo máximo total
    this.usuario = usuarioP;
    this.contra = contraP;
  }
}
```

```

}

class Contratacion {
    constructor(numeroID, clienteCP, paseadorCP, estadoCP, nombreperroC,
sizeperroC) {
        this.numeroIdentificador = numeroID;
        this.cliente = clienteCP;
        this.paseador = paseadorCP;
        this.estado = estadoCP; // Estados: "Pendiente", "Aprobada", "Cancelada"
        this.nombreperro = nombreperroC;
        this.sizeperro = sizeperroC;
        this.cupoDescontado = false; // Para controlar si ya se descontó el cupo
    }
}

class paseadorLogeado {
    constructor(idPL, usuarioPL, contraPL) {
        this.id = idPL;
        this.usuario = usuarioPL;
        this.contra = contraPL;
    }
}

// Clase sistema, se encarga de gestionar el sistema en general.
// Esta clase es la que se encarga de gestionar los paseadores, clientes y
contrataciones.

class Sistema {
    constructor() {
        // Datos de prueba hardcodeados
        this.paseador = [
            new Paseador(1, "Jorge", 10, 10, "JorgeB", "Jorge123"), // El orden es
Nombre Paseador, Cupo Máximo, Usuario Paseador, Contraseña Paseador
            new Paseador(2, "Rodrigo", 10, 10, "RodrigoF", "Rodrigo123"),
            new Paseador(3, "Manuel", 15, 15, "ManuelB", "Manuel123"),
            new Paseador(4, "Ana", 8, 8, "AnaG", "Ana123"),
            new Paseador(5, "Lucia", 12, 12, "LuciaM", "Lucia123"),
        ];

        this.cliente = [
            new Cliente(1, "cliente1", "Aa123", "Fido", "Grande"),
            new Cliente(2, "cliente2", "Bb234", "Luna", "Chico"),
            new Cliente(3, "cliente3", "Cc345", "Max", "Mediano"),
            new Cliente(4, "cliente4", "Dd456", "Rex", "Grande"),
        ];
    }
}

```

```

    new Cliente(5, "cliente5", "Ee567", "Tina", "Chico"),
    new Cliente(6, "cliente6", "Ff678", "Rocky", "Mediano"),
    new Cliente(7, "cliente7", "Gg789", "Bella", "Grande"),
    new Cliente(8, "cliente8", "Hh890", "Coco", "Chico"),
    new Cliente(9, "cliente9", "Ii901", "Bruno", "Chico"),
    new Cliente(10, "cliente10", "Jj012", "Kira", "Chico"),
    new Cliente(11, "Marcos", "Marcos123", "Firulais", "Grande"),
    new Cliente(12, "Laura", "Laura123", "Luna", "Chico"),
    new Cliente(13, "Carlos", "Carlos123", "Snoppy", "Mediano"),
    new Cliente(14, "Ana", "Ana123", "Simba", "Chico"),
    new Cliente(15, "Selma", "Selma123", "Juan", "Grande"),
    new Cliente(16, "Pedro", "Pedro123", "Gusi", "Mediano"),
    new Cliente(17, "Sofia", "Sofia123", "Tobias", "Chico"),
    new Cliente(18, "Miguel", "Miguel123", "Tortuga", "Grande"),
    new Cliente(19, "Elena", "Elena123", "Mesi", "Mediano"),
    new Cliente(20, "Roberto", "Roberto123", "Jamon", "Chico"),
    new Cliente(21, "Lucia", "Lucia123", "Bob", "Grande"),
    new Cliente(22, "Andres", "Andres123", "Lola", "Mediano"),
    new Cliente(23, "Angel", "Angel123", "Kiara", "Chico"),
    new Cliente(24, "Mariana", "Mariana123", "Theo", "Grande"),
    new Cliente(25, "Manuela", "Manuela123", "Sasha", "Mediano"),
    new Cliente(26, "Diego", "Diego123", "Duke", "Grande"),
    new Cliente(27, "Paula", "Paula123", "Cordero", "Chico"),
    new Cliente(28, "Matias", "Matias123", "Maxi", "Mediano"),
    new Cliente(29, "Susana", "Susana123", "Max", "Mediano"),
    new Cliente(30, "Georgina", "Georgina123", "Pia", "Chico"),
];

this.contrataciones = [
    new Contratacion(1, "Jose", 2, "Pendiente", "Fafa", "Mediano"),
    new Contratacion(2, "Roberto", 2, "Pendiente", "Rex", "Grande"),
    new Contratacion(4, "Simon", 2, "Pendiente", "Titi", "Chico"),
    new Contratacion(5, "Claudia", 2, "Pendiente", "Mar", "Mediano"),
    new Contratacion(7, "Pablo", 2, "Pendiente", "Colo", "Chico"),
    new Contratacion(9, "Josefina", 2, "Pendiente", "Beto", "Mediano"),
    new Contratacion(10, "Maite", 2, "Pendiente", "Firu", "Grande"),
    new Contratacion(11, "Celeste", 2, "Pendiente", "Toto", "Chico"),
    new Contratacion(12, "Diana", 2, "Pendiente", "Dulce", "Mediano"),
    new Contratacion(13, "Julieta", 2, "Pendiente", "Teco", "Mediano"),
    new Contratacion(14, "Valeria", 2, "Pendiente", "severus", "Mediano"),

    new Contratacion(13, "Gonzalo", 1, "Aprobada", "Poncho", "Chico"),
    new Contratacion(14, "Santiago", 1, "Aprobada", "Maru", "Mediano"),
    new Contratacion(17, "Tomas", 3, "Aprobada", "Sarha", "Grande"),

```



```

        new Contratacion(18, "Mario", 3, "Aprobada", "Thor", "Mediano"),
        new Contratacion(19, "Angela", 4, "Aprobada", "Kia", "Chico"),
        new Contratacion(20, "Andrea", 4, "Aprobada", "Leo", "Mediano"),
        new Contratacion(21, "Lucas", 5, "Aprobada", "Berto", "Grande"),
        new Contratacion(22, "Romina", 5, "Aprobada", "Juju", "Mediano")
    ];
}
}
let sistema = new Sistema();

// Función para obtener un elemento de un array por una propiedad y su valor.
// Esta función recorre un array de objetos y devuelve el primer objeto que
// tenga la propiedad especificada con el valor dado.

function obtenerElemento(arrBuscar, prop, valor) {
    let objDev = null;
    for (let i = 0; i < arrBuscar.length; i++) {
        const obj = arrBuscar[i];
        if (obj[prop] === valor) {
            objDev = obj;
            break;
        }
    }
    return objDev;
}

// Muestra en tabla HTML todas las contrataciones pendientes que puede aceptar
// el paseador logueado
function tablaContratacionesPendientes() {
    let tablaHTML = "";
    if (logueado) {
        for (let i = 0; i < sistema.contrataciones.length; i++) {
            const unObjetoContratacion = sistema.contrataciones[i];

            // Solo mostrar contrataciones con estado "Pendiente"
            if (unObjetoContratacion.estado === "Pendiente") {
                tablaHTML += `<tr>
                    <td>${unObjetoContratacion.nombreperro}</td>
                    <td>${unObjetoContratacion.cliente}</td>
                    <td>${unObjetoContratacion.sizeperro}</td>
                    <td><input type="button" value="Procesar Contratacion"
class="botonProcesar"
data-id="${unObjetoContratacion.numeroIdentificador}"></td>
                </tr>`;
            }
        }
    }
}

```

```

    }
  }
  document.querySelector("#tblContratacionesPendientes").innerHTML =
tablaHTML;

  // Agregar event listeners a los botones de procesar
  let botonesProcesar = document.querySelectorAll(".botonProcesar");
  for (let i = 0; i < botonesProcesar.length; i++) {
    const boton = botonesProcesar[i];
    boton.addEventListener("click", procesarContratacion);
  }
}
}

// Lógica principal para que el paseador acepte o rechace contrataciones según
disponibilidad y restricciones
function procesarContratacion() {
  let idProcesarContratacion = Number(this.getAttribute("data-id"));
  let costoCupo = 0;

  for (let i = 0; i < sistema.contrataciones.length; i++) {
    let cupoActual = paseadorActual.cupoActual;

    if (sistema.contrataciones[i].numeroIdentificador ===
idProcesarContratacion && sistema.contrataciones[i].estado === "Pendiente") {

      // Calcular costo de cupo según tamaño del perro
      if (sistema.contrataciones[i].sizeperro === "Grande") {
        paseadorActual.perroGrande = true; // Marcar que tiene perro grande
        console.log("unPerroGrande : Existe")
        costoCupo = 4;
      } else if (sistema.contrataciones[i].sizeperro === "Mediano") {
        costoCupo = 2;
      } else if (sistema.contrataciones[i].sizeperro === "Chico") {
        paseadorActual.perroChico = true; // Marcar que tiene perro chico
        console.log("unPerroChico: Existe")
        costoCupo = 1;
      }
      console.log("Cupo actual del paseador: " + cupoActual);

      // Verificar si tiene cupo suficiente Y si no hay incompatibilidad de
tamaños
      // Regla: no puede tener perros grandes Y chicos al mismo tiempo

```

```

        if (paseadorActual.cupoActual >= costoCupo &&
(sistema.contrataciones[i].sizeperro === "Mediano" ||
(paseadorActual.perroChico !== paseadorActual.perroGrande))) {
            console.log("Entró al IF, va a aprobar");
            sistema.contrataciones[i].estado = "Aprobada";
            sistema.contrataciones[i].cupoDescontado = true;
            sistema.contrataciones[i].paseador =
paseadorActual.numeroIdentificador;
            paseadorActual.cupoActual -= costoCupo; // Descontar cupo
            alert(`La contratación ha sido aprobada. Debes pasar a buscar a
${sistema.contrataciones[i].nombreperro} ; Cupo:
${paseadorActual.cupoActual}`);
            tablaContratacionesPendientes();
            verificarBotonesDeshabilitados();
        }
    }
}

// Deshabilita botones de contrataciones que el paseador no puede aceptar por
falta de cupo o incompatibilidad
function verificarBotonesDeshabilitados() {
    let botones = document.querySelectorAll(".botonProcesar");
    let cupoActual = paseadorActual.cupoActual;
    let cupoTotal = paseadorActual.cupoMax;

    for (let j = 0; j < botones.length; j++) {
        let botonIdContratacion = Number(botones[j].getAttribute("data-id"));

        for (let k = 0; k < sistema.contrataciones.length; k++) {
            if (sistema.contrataciones[k].numeroIdentificador === botonIdContratacion
&& sistema.contrataciones[k].estado === "Pendiente") {

                // Deshabilitar si no hay cupo suficiente o hay incompatibilidad de
tamaños
                if ((sistema.contrataciones[k].sizeperro === "Grande" && cupoActual <
4) || (paseadorActual.perroChico === true &&
sistema.contrataciones[k].sizeperro === "Grande")) {
                    botones[j].setAttribute("value", "Sin cupo disponible");
                    botones[j].setAttribute("disabled", "disabled");
                }
                if (sistema.contrataciones[k].sizeperro === "Mediano" && cupoActual <
2) {
                    botones[j].setAttribute("value", "Sin cupo disponible");

```

```

        botones[j].setAttribute("disabled", "disabled");
    }

    if ((sistema.contrataciones[k].sizeperro === "Chico" && cupoActual < 1)
|| (paseadorActual.perroGrande === true && sistema.contrataciones[k].sizeperro
=== "Chico")) {
        botones[j].setAttribute("value", "Sin cupo disponible");
        botones[j].setAttribute("disabled", "disabled");
    }

    console.log("Cupo actual del paseador: " + paseadorActual.cupoActual);
    // Actualizar información de cupos en pantalla
    document.querySelector("#pTotalCuposMax").innerHTML = "Total de cupos
disponibles: " + paseadorActual.cupoActual + "/" + cupoTotal;
    document.querySelector("#pPorcentajeCupos").innerHTML = "Porcentaje
ocupado: " + Math.round(((cupoTotal - paseadorActual.cupoActual) / cupoTotal)
* 100) + "%";

    break;
}
}
}
}

// Esta función se encarga de mostrar las contrataciones del paseador logueado
en una tabla HTML.

function mostrarContratacionesDelPaseador() {
    let tablaHTML = "";
    if (logueado && paseadorActual !== null) {
        for (let i = 0; i < sistema.contrataciones.length; i++) {
            const unaContratacion = sistema.contrataciones[i];

            // Mostrar solo contrataciones aprobadas del paseador actual
            if (unaContratacion.estado === "Aprobada" &&
Number(unaContratacion.paseador) === paseadorActual.numeroIdentificador) {
                tablaHTML += `<tr>
                    <td>${unaContratacion.nombreperro}</td>
                    <td>${unaContratacion.sizeperro}</td>
                    <td>${unaContratacion.cliente}
                </tr>`;
            }
        }
    }

    if (tablaHTML === "") {

```

```

        tablaHTML = `<tr><td>No tenés perros asignados aún.</td></tr>`;
    }
    document.querySelector("#tblPerrosAsignados").innerHTML = tablaHTML;
}
}

// Esta función se encarga de cerrar la sesión del cliente logueado, limpiando
los campos de entrada y el aviso.

document.querySelector("#btnCerrarSesionCliente").addEventListener("click",
cerrarSesionCliente);

function cerrarSesionCliente() {
    // Limpiar todos los campos del formulario
    document.querySelector("#txtUsuarioCliente").value = "";
    document.querySelector("#txtContraCliente").value = "";
    document.querySelector("#txtNombrePerroCliente").value = "";
    document.querySelector("#txtSizePerroCliente").value = "";
    document.querySelector("#pAvisosCliente").innerHTML = "";
    clienteLogueado = null;
    alert("Se ha cerrado la sesión correctamente");
    tipoUsuario = "inicio";
    cambiarSeccion("inicio");
}

// Esta función se encarga de cerrar la sesión del paseador logueado,
limpiando los campos de entrada y el aviso.

document.querySelector("#btnCerrarSesionPaseador").addEventListener("click",
cerrarSesionPaseador);

function cerrarSesionPaseador() {
    logeado = false;
    paseadorActual = null;
    // Limpiar campos del formulario
    document.querySelector("#txtUsuarioPaseador").value = "";
    document.querySelector("#txtContraPaseador").value = "";
    alert("Se ha cerrado la sesión correctamente");
    tipoUsuario = "inicio";
    cambiarSeccion("inicio");
}

let clienteLogueado = null;

```

```

// Esta función se encarga de iniciar sesión como cliente, verificando el
usuario y la contraseña ingresados.

document.querySelector("#btnIniciarSesionCliente").addEventListener("click",
iniciarSesionCliente);

function iniciarSesionCliente() {
    let usuarioCliente = document.querySelector("#txtUsuarioCliente").value;
    let contraCliente = document.querySelector("#txtContraCliente").value;
    let cliente = obtenerElemento(sistema.cliente, "usuario", usuarioCliente); //
    Buscar cliente por usuario

    if (!usuarioCliente || !contraCliente) {
        document.querySelector("#pAvisosCliente").innerHTML = "Por favor, complete
    ambos campos.";
        return;
    }

    // Verificar credenciales
    if (cliente && cliente.usuario === usuarioCliente && cliente.contra ===
    contraCliente) {
        document.querySelector("#pSesionIniciadaCliente").innerHTML =
            `Sesión iniciada correctamente ${cliente.usuario}`;
        clienteLogueado = cliente;
        tipoUsuario = "cliente";
        cambiarSeccion("seccionClienteLogueado");
    } else {
        document.querySelector("#pAvisosCliente").innerHTML = "Por favor, ingrese
    un usuario y contraseña válidos";
    }
}

// Registra nuevos clientes validando que no existan y que la contraseña
cumpla requisitos
document.querySelector("#btnRegistrarCliente").addEventListener("click",
registrarCliente);

function registrarCliente() {
    let usuario = document.querySelector("#txtUsuarioClienteReg").value;
    let nombrePerro = document.querySelector("#txtNombrePerroCliente").value;
    let sizePerro = document.querySelector("#txtSizePerroCliente").value;
    let contra;
    let usuarioYaExiste = false;

```

```

// Verificar si el usuario ya existe
for (let i = 0; i < sistema.cliente.length; i++) {
    if (sistema.cliente[i].usuario.toLowerCase() === usuario.toLowerCase()) {
        usuarioYaExiste = true;
        break;
    }
}

if (usuarioYaExiste) {
    document.querySelector("#pAvisosClienteReg").innerHTML = "El usuario ya
existe. Por favor, elija otro.";

} else {
    // Variables para validar la contraseña
    let minimoDeCaracteres = "No cumple";
    let condicionDeMayuscula = "No cumple";
    let condicionDeMinuscula = "No cumple";
    let condicionDeNumero = "No cumple";
    let condicionDeAusenciaDeEspacios = "Cumple";
    contra = "";
    document.querySelector("#pAvisosClienteReg").innerHTML = "";
    contra = document.querySelector("#txtContraClienteReg").value;

    // Validar longitud mínima de la contraseña
    if (contra.length >= 5) {
        minimoDeCaracteres = "Cumple";
    }

    // Verificar espacios
    for (let i = 0; i < contra.length; i++) {
        if (contra.charAt(i) === " ") {
            condicionDeAusenciaDeEspacios = "No cumple";
        }
    }

    // Validar mayúsculas, minúsculas y números
    for (let i = 0; i < contra.length; i++) {
        let caracter = contra.charAt(i);

        if (!isNaN(caracter) && i !== 0) { // Número que no esté en primera
posición
            condicionDeNumero = "Cumple";
        }

        if (caracter === caracter.toUpperCase() && caracter !==
caracter.toLowerCase()) {

```

```

        condicionDeMayuscula = "Cumple";
    }

    if (caracter === caracter.toLowerCase() && caracter !==
caracter.toUpperCase()) {
        condicionDeMinuscula = "Cumple";
    }
}
}

// Verificar si todas las condiciones se cumplen
if (minimoDeCaracteres === "Cumple" && condicionDeMayuscula === "Cumple" &&
condicionDeMinuscula === "Cumple" && condicionDeAusenciaDeEspacios ===
"Cumple") {
    if (usuario && contra && nombrePerro && sizePerro) {
        let nuevoCliente = new Cliente(proximoID++, usuario, contra,
nombrePerro, sizePerro);
        sistema.cliente.push(nuevoCliente);
        alert("Cliente registrado correctamente: " + nuevoCliente.usuario);
        console.log(nuevoCliente.usuario)
        // Limpiar formulario
        document.querySelector("#txtUsuarioClienteReg").value = "";
        document.querySelector("#txtContraClienteReg").value = "";
        document.querySelector("#txtNombrePerroCliente").value = "";
        document.querySelector("#txtSizePerroCliente").value = "";
    }
    document.querySelector("#pAvisosClienteReg").innerHTML =
        "Su contraseña es apta para nuestro sistema y fue validada
correctamente." +
        "<br><br>Su contraseña fue: " + contra;
} else {
    // Mostrar qué condiciones no se cumplen
    document.querySelector("#pAvisosClienteReg").innerHTML =
        "Su contraseña aún no es apta para nuestro sistema, revise los
siguientes puntos: <br><br>" +
        "Minimo de 5 caracteres: " + minimoDeCaracteres + "<br>" +
        "Tener al menos una mayúscula (Usuario y Contraseña): " +
condicionDeMayuscula + "<br>" +
        "Tener al menos una minúscula (Usuario y Contraseña): " +
condicionDeMinuscula + "<br>" +
        "No tener espacios: " + condicionDeAusenciaDeEspacios + "<br>" +
        "Tener un número en algún lugar que no sea el primero: " +
condicionDeNumero + "<br><br>" +
        "Su contraseña fue: " + contra;
}

```



```

    }

    if (!usuario || !contra || !nombrePerro || !sizePerro) {
        document.querySelector("#pAvisosCliente").innerHTML =
            "Por favor, complete todos los campos.";
    }
}
}

// Esta función se encarga de iniciar sesión como paseador, verificando el
usuario y la contraseña ingresados.

document.querySelector("#btnIniciarSesionPaseador").addEventListener("click",
iniciarSesionPaseador);

function iniciarSesionPaseador() {
    let usuarioPaseador = document.querySelector("#txtUsuarioPaseador").value;
    let contraPaseador = document.querySelector("#txtContraPaseador").value;
    let paseador = obtenerElemento(sistema.paseador, "usuario", usuarioPaseador);
    // Buscar paseador por usuario

    if (paseador && usuarioPaseador === paseador.usuario && contraPaseador ===
paseador.contra) {
        logeado = true;
        paseadorActual = paseador;
        // Inicializar variables de tipos de perro
        paseadorActual.perroGrande = false;
        paseadorActual.perroChico = false;
        tipoUsuario = "paseador";
        cambiarSeccion("seccionPaseadorLogueado");

        document.querySelector("#pSesionIniciadaPaseador").innerHTML = "Sesión
iniciada correctamente " + paseador.usuario;
        validarProcesarContratacion(); // Actualizar cupos según contrataciones
existentes
    } else {
        logeado = false;
        paseadorActual = null;
        document.querySelector("#pAvisosPaseador").innerHTML = "Por favor, ingrese
un usuario y contraseña válidos";
    }
}
}

```

```

// Función compleja que actualiza los cupos de todos los paseadores según las
contrataciones aprobadas
// También marca si el paseador logueado ya tiene perros grandes o chicos
function validarProcesarContratacion() {

    paseadorActual.perroGrande = false;
    paseadorActual.perroChico = false;

    for (let i = 0; i < sistema.contrataciones.length; i++) {
        const ObjCont = sistema.contrataciones[i];
        if (ObjCont.estado === "Aprobada") {
            const ObjPas = obtenerElemento(sistema.paseador, "numeroIdentificador",
ObjCont.paseador);
            if (ObjPas) {
                // Descontar cupo según tamaño del perro
                if (ObjCont.sizeperro === "Grande") {
                    if (!ObjCont.cupoDescontado) {
                        ObjPas.cupoActual -= 4;
                    }
                    if (ObjPas === paseadorActual) {
                        paseadorActual.perroGrande = true;
                    }
                } else if (ObjCont.sizeperro === "Mediano") {
                    if (!ObjCont.cupoDescontado) {
                        ObjPas.cupoActual -= 2;
                    }
                } else if (ObjCont.sizeperro === "Chico") {
                    if (!ObjCont.cupoDescontado) {
                        ObjPas.cupoActual -= 1;
                    }
                }

                if (ObjPas === paseadorActual) {
                    paseadorActual.perroChico = true;
                }
            }
            ObjCont.cupoDescontado = true;
        }
    }
}

let botones = document.querySelectorAll(".boton");

```

```

for (let i = 0; i < botones.length; i++) {
  const botonHTML = botones[i];
  botonHTML.addEventListener("click", mostrarSeccion);
}

// Esta función se encarga de mostrar la sección correspondiente al botón que
se ha pulsado.

function mostrarSeccion() {
  let idBoton = this.getAttribute("id");
  let idSeccion = idBoton.charAt(3).toLowerCase() + idBoton.substring(4); //
  Convertir ID del botón a ID de sección
  cambiarSeccion(idSeccion);
}

// Esta función se encarga de mostrar los botones correspondientes al tipo de
usuario que ha iniciado sesión.

function mostrarBotones(tipoUsuario) {
  // Ocultar todos los botones primero
  let botones = document.querySelectorAll(".boton");
  for (let i = 0; i < botones.length; i++) {
    const botonHTML = botones[i];
    botonHTML.style.display = "none";
  }

  // Mostrar solo los botones del tipo de usuario actual
  let botonesMostrar = document.querySelectorAll("." + tipoUsuario);
  for (let i = 0; i < botonesMostrar.length; i++) {
    const botonHTML = botonesMostrar[i];
    botonHTML.style.display = "block";
  }
}

// Muestra paseadores disponibles que tienen cupo suficiente para el perro del
cliente logueado
function tablaPaseadoresDisponibles() {
  let tablaHTML = "";

  for (let i = 0; i < sistema.paseador.length; i++) {
    const unPaseador = sistema.paseador[i];
    let costoCupo = 0

    if (clienteLogueado.sizeperro === "Grande") {

```

```

    costoCupo = 4;
  } else if (clienteLogueado.sizeperro === "Mediano") {
    costoCupo = 2;
  } else if (clienteLogueado.sizeperro === "Chico") {
    costoCupo = 1;
  }

  // Solo mostrar paseadores que tengan cupo disponible
  if (unPaseador.cupoActual >= costoCupo) {
    console.log(costoCupo)
    tablaHTML += `<tr>
      <td>${unPaseador.nombre}</td>
      <td>${unPaseador.cupoActual}</td>
      <td><input type="button" value="Contratar Paseador"
class="botonContratar" data-id="${unPaseador.numeroIdentificador}"></td>
    </tr>`;
  }
}

if (tablaHTML === "") {
  tablaHTML = `<tr><td>No hay paseadores disponibles en este
momento.</td></tr>`;
}

document.querySelector("#tblPaseadoresDisponibles").innerHTML = tablaHTML;

let botonesContratar = document.querySelectorAll(".botonContratar");
for (let i = 0; i < botonesContratar.length; i++) {
  const boton = botonesContratar[i];
  boton.addEventListener("click", contratarPaseador);
}

let botonesCancelar = document.querySelectorAll(".botonCancelar");
for (let i = 0; i < botonesCancelar.length; i++) {
  const botonC = botonesCancelar[i];
  botonC.addEventListener("click", cancelarContratacionDelPaseador)
  if (Number(botonC.getAttribute("data-id")) !== idPaseadorSeleccionado) {
    botonC.hidden = true;
  }
}
}
}

//Muestra todos los paseadores.

```

```

function tablaPaseadoresTotales() {
  let tablaHTML = "";

  for (let i = 0; i < sistema.paseador.length; i++) {
    const unPaseador = sistema.paseador[i];
    tablaHTML += `<tr>
      <td>${unPaseador.nombre}</td>
      <td>${unPaseador.cupoActual}</td>
    </tr>`;
  }

  if (tablaHTML === "") {
    tablaHTML = `<tr><td>No hay paseadores.</td></tr>`;
  }

  document.querySelector("#tblPaseadoresTotales").innerHTML = tablaHTML;
}

//Es la encargada de hacer el proceso de contratar a un paseador si cumple con
los requerimientos.

function contratarPaseador() {
  let idPaseadorSeleccionado = Number(this.getAttribute("data-id"));
  let costoCupo = 0;
  let existeContratacion = false

  for (let i = 0; i < sistema.contrataciones.length; i++) {
    let contratacion = sistema.contrataciones[i];
    if (contratacion.cliente === clienteLogueado.usuario &&
(contratacion.estado === "Pendiente" || contratacion.estado === "Aprobada")) {
      existeContratacion = true;
    }
  }

  if (existeContratacion) {
    alert("Ya tienes una contratacion pendiente.");
  }

  if (clienteLogueado.sizeperro === "Grande") {
    costoCupo = 4;
  } else if (clienteLogueado.sizeperro === "Mediano") {
    costoCupo = 2;
  } else if (clienteLogueado.sizeperro === "Chico") {
    costoCupo = 1;
  }
}

```

```

    let paseadorSeleccionado = obtenerElemento(sistema.paseador,
"numeroIdentificador", idPaseadorSeleccionado);

    if (paseadorSeleccionado && paseadorSeleccionado.cupoActual >= costoCupo &&
!existeContratacion) {
        let nuevaContratacion = new Contratacion(proximoID++,
clienteLogueado.usuario, idPaseadorSeleccionado, "Pendiente",
clienteLogueado.nombreperro, clienteLogueado.sizeperro);
        sistema.contrataciones.push(nuevaContratacion);

        document.querySelector("#pAvisosCliente").innerHTML = `Solicitud de
contratación enviada para ${paseadorSeleccionado.nombre}. Estado: Pendiente de
aprobación.`;

        tablaPaseadoresDisponibles();
    } else {
        document.querySelector("#pAvisosCliente").innerHTML =
            `No se pudo contratar al paseador. Cupo insuficiente para perros de
tamaño ${clienteLogueado.sizeperro}.`;
    }
}

// Tabla donde esta la interfaz para poder cancelar una contratacion, que solo
va a aparecer si esta en pendiente.
function tablaCancelarContratacion() {

    if (clienteLogueado) {
        let tablaHTML = "";

        for (let i = 0; i < sistema.contrataciones.length; i++) {
            const unObjetoContratacion = sistema.contrataciones[i];
            let paseadorObj = obtenerElemento(sistema.paseador,
"numeroIdentificador", unObjetoContratacion.paseador);
            let unPaseador;

            if (paseadorObj) {
                unPaseador = paseadorObj.nombre;
            }

            if (unObjetoContratacion.cliente === clienteLogueado.usuario &&
unObjetoContratacion.estado === "Pendiente") {
                tablaHTML += `<tr>
                <td>${unPaseador}</td>

```

```

        <td><input type="button" value="Cancelar Contratación"
class="botonCancelar"
data-idCancelar="${unObjetoContratacion.numeroIdentificador}"></td>
    </tr>`;
    }
    document.querySelector("#tblCancelarContratacion").innerHTML = tablaHTML;

    let botonesCancelar = document.querySelectorAll(".botonCancelar");
    for (let i = 0; i < botonesCancelar.length; i++) {
        botonesCancelar[i].addEventListener("click", cancelarContratacion);
    }
}
}
}

// Es la encargada de que el cliente puede cancelar su contratacion si no la
Acepto el paseador.
function cancelarContratacion() {
    let idCancelarContratacion = Number(this.getAttribute("data-idCancelar"));

    for (let i = 0; i < sistema.contrataciones.length; i++) {
        const contratacion = sistema.contrataciones[i];
        if (contratacion.numeroIdentificador === idCancelarContratacion &&
contratacion.estado === "Pendiente") {
            contratacion.estado = "Cancelada";
            document.querySelector("#pAvisosCliente").innerHTML = "Contratación
cancelada correctamente.";
            break;
        }
    }
    tablaCancelarContratacion();
}
mostrarBotones("inicio");

// Esta función se encarga de cambiar la sección visible en la interfaz,
ocultando las demás secciones y mostrando la nueva sección.

function cambiarSeccion(seccionNueva) {
    ocultarSecciones();
    const seccion = document.querySelector("#" + seccionNueva);
    seccion.style.display = "block";

    switch (seccionNueva) {
        case "inicio":

```

```

        mostrarBotones("inicio");
        break;
    case "cliente":
    case "seccionCliente":
    case "seccionClienteIniciarSesion":
    case "seccionClienteRegistrarse":
    case "seccionClienteLogueado":
    case "seccionCalificarPaseador":
        mostrarBotones("cliente");
        break;
    case "seccionContratarPaseador":
        mostrarBotones("cliente");
        tablaPaseadoresDisponibles();
        break;
    case "seccionTodoPaseadores":
        tablaPaseadoresTotales();
        break;
    case "seccionCancelarContratacion":
        mostrarBotones("cliente");
        tablaCancelarContratacion();
        break;
    case "paseador":
    case "seccionPaseador":
    case "seccionPaseadorIniciarSesion":
    case "seccionPaseadorLogueado":
        mostrarBotones("paseador");
        break;
    case "seccionContrataciones":
        mostrarBotones("paseador");
        tablaContratacionesPendientes();
        verificarBotonesDeshabilitados();
        break;
    case "seccionPerrosAsignados":
        mostrarBotones("paseador");
        mostrarContratacionesDelPaseador();
        break;
    default:
        break;
}
}

// Esta función se encarga de ocultar todas las secciones de la interfaz,
excepto la sección de inicio.
function ocultarSecciones() {
    let secciones = document.querySelectorAll(".seccion, .inicio");

```



```

for (let i = 0; i < secciones.length; i++) {
    const seccionHTML = secciones[i];
    seccionHTML.style.display = "none";
}
}

// Variable global para saber cuántas estrellas eligió el usuario
let estrellasSeleccionadas = 0;

// Configuramos el sistema de estrellas interactivo de (1 a 5 estrellas)
for (let i = 1; i <= 5; i++) {
    let estrella = document.querySelector("#Estrella" + i);

    estrella.addEventListener("click", funcionEstrella);

    // Función que maneja el click en cada estrella
    function funcionEstrella() {
        estrellasSeleccionadas = i; // Guardamos cuántas estrellas seleccionó
        for (let j = 1; j <= 5; j++) {
            let est = document.querySelector("#Estrella" + j);
            est.value = j <= i ? "★" : "☆";
        }
    }
}

// Conectamos el botón de calificar con su función
document.querySelector("#btnCalificarPaseador").addEventListener("click",
funcionComentar);

// Función principal que procesa la calificación del paseador
function funcionComentar() {
    let comentario = document.querySelector("#txtCalificarPaseador").value;
    let mensaje = "";

    // Chequea que haya estrellas y comentario
    if (estrellasSeleccionadas === 0 && comentario === "") {
        document.querySelector("#pRevisionPaseador").innerHTML = `Seleccione una
cantidad de estrellas. <br> Escriba un comentario.`;
    } else {

        // Buscamos si el cliente logueado tiene alguna contratación aprobada
        let contratacionCliente = null;
        for (let i = 0; i < sistema.contrataciones.length; i++) {
            const contratacion = sistema.contrataciones[i];

```

```

        if (contratacion.cliente === clienteLogueado.usuario &&
contratacion.estado === "Aprobada") {
            contratacionCliente = contratacion;
            break;
        }
    }

    // Si no tiene contrataciones aprobadas, no puede calificar
    if (!contratacionCliente) {
        document.querySelector("#pRevisionPaseador").innerHTML = "No tienes
ningún paseador al cual calificar (Para poder calificar a un paseador debes de
tener una contratación aceptada por alguno de ellos).";
    } else {

        // Obtenemos los datos del paseador que fue contratado usando una función
auxiliar (La de obtenerElementos)
        let paseadorAsignado = obtenerElemento(sistema.paseador,
"numeroIdentificador", contratacionCliente.paseador);

        if (!paseadorAsignado) {
            document.querySelector("#pRevisionPaseador").innerHTML = "Error al
obtener el paseador asignado.";
        } else {

            document.querySelector("#nombreDelPaseadorCalificado").innerHTML = ""
            document.querySelector("#nombreDelPaseadorCalificado").innerHTML = "Tu
Paseador: " + paseadorAsignado.nombre;

            // Aca convertimos las estrellas para que sean estrellitas
            let estrellas = "";
            for (let i = 0; i < estrellasSeleccionadas; i++) {
                estrellas += "★";
            }

            mensaje = `
<br><br>
👤 Cliente: ${clienteLogueado.usuario}<br>
🐶 Paseador calificado: ${paseadorAsignado.nombre}<br>
🗨 Comentario: ${comentario}<br>
★ Calificación: ${estrellas}<br>
<br><br>
`;

            // Agregamos la calificación a la tabla de revisiones
            document.querySelector("#tblRevisionPaseador").innerHTML += mensaje;

```

```

document.querySelector("#pRevisionPaseador").innerHTML = "";

// Limpiamos el formulario después de enviar la calificación
document.querySelector("#txtCalificarPaseador").value = "";
estrellasSeleccionadas = 0;
for (let i = 1; i <= 5; i++) {
    document.querySelector("#Estrella" + i).value = "☆";
}
}
}
}
}
}
}
}

```

Casos de prueba:

PASEADORES

Jorge (ID: 1)

- Usuario: JorgeB
- Contraseña: Jorge123
- Cupo máximo: 10

Rodrigo (ID: 2)

- Usuario: RodrigoF
- Contraseña: Rodrigo123
- Cupo máximo: 10

Manuel (ID: 3)

- Usuario: ManuelB
- Contraseña: Manuel123
- Cupo máximo: 15

Ana (ID: 4)

- Usuario: AnaG
- Contraseña: Ana123
- Cupo máximo: 8

Lucia (ID: 5)

- Usuario: LuciaM
- Contraseña: Lucia123
- Cupo máximo: 12

CLIENTES

Marcos (ID: 11)

- Usuario: Marcos
- Contraseña: Marcos123
- Perro: Firulais (Grande)

Laura (ID: 12)

- Usuario: Laura
- Contraseña: Laura123
- Perro: Luna (Chico)

Carlos (ID: 13)

- Usuario: Carlos
- Contraseña: Carlos123
- Perro: Snoppy (Mediano)

Ana (ID: 14)

- Usuario: Ana
- Contraseña: Ana123
- Perro: Simba (Chico)

Selma (ID: 15)

- Usuario: Selma
- Contraseña: Selma123
- Perro: Juan (Grande)

Pedro (ID: 16)

- Usuario: Pedro
- Contraseña: Pedro123
- Perro: Gusi (Mediano)

Sofia (ID: 17)

- Usuario: Sofia

- Contraseña: Sofia123
- Perro: Tobias (Chico)

Miguel (ID: 18)

- Usuario: Miguel
- Contraseña: Miguel123
- Perro: Tortuga (Grande)

CONTRATACIONES PENDIENTES

Jose (ID: 1)

- Paseador: Rodrigo (ID: 2)
- Perro: Fafa (Mediano)
- Estado: Pendiente

Roberto (ID: 2)

- Paseador: Rodrigo (ID: 2)
- Perro: Rex (Grande)
- Estado: Pendiente

Simon (ID: 4)

- Paseador: Rodrigo (ID: 2)
- Perro: Titi (Chico)
- Estado: Pendiente

Claudia (ID: 5)

- Paseador: Rodrigo (ID: 2)
- Perro: Mar (Mediano)
- Estado: Pendiente

Pablo (ID: 7)

- Paseador: Rodrigo (ID: 2)
- Perro: Colo (Chico)
- Estado: Pendiente

CONTRATACIONES APROBADAS

Gonzalo (ID: 13)

- Paseador: Jorge (ID: 1)
- Perro: Poncho (Chico)
- Estado: Aprobada

Santiago (ID: 14)

- Paseador: Jorge (ID: 1)
- Perro: Maru (Mediano)
- Estado: Aprobada

Tomas (ID: 17)

- Paseador: Manuel (ID: 3)
- Perro: Sarha (Grande)
- Estado: Aprobada

Mario (ID: 18)

- Paseador: Manuel (ID: 3)
- Perro: Thor (Mediano)
- Estado: Aprobada

Angela (ID: 19)

- Paseador: Ana (ID: 4)
- Perro: Kia (Chico)
- Estado: Aprobada