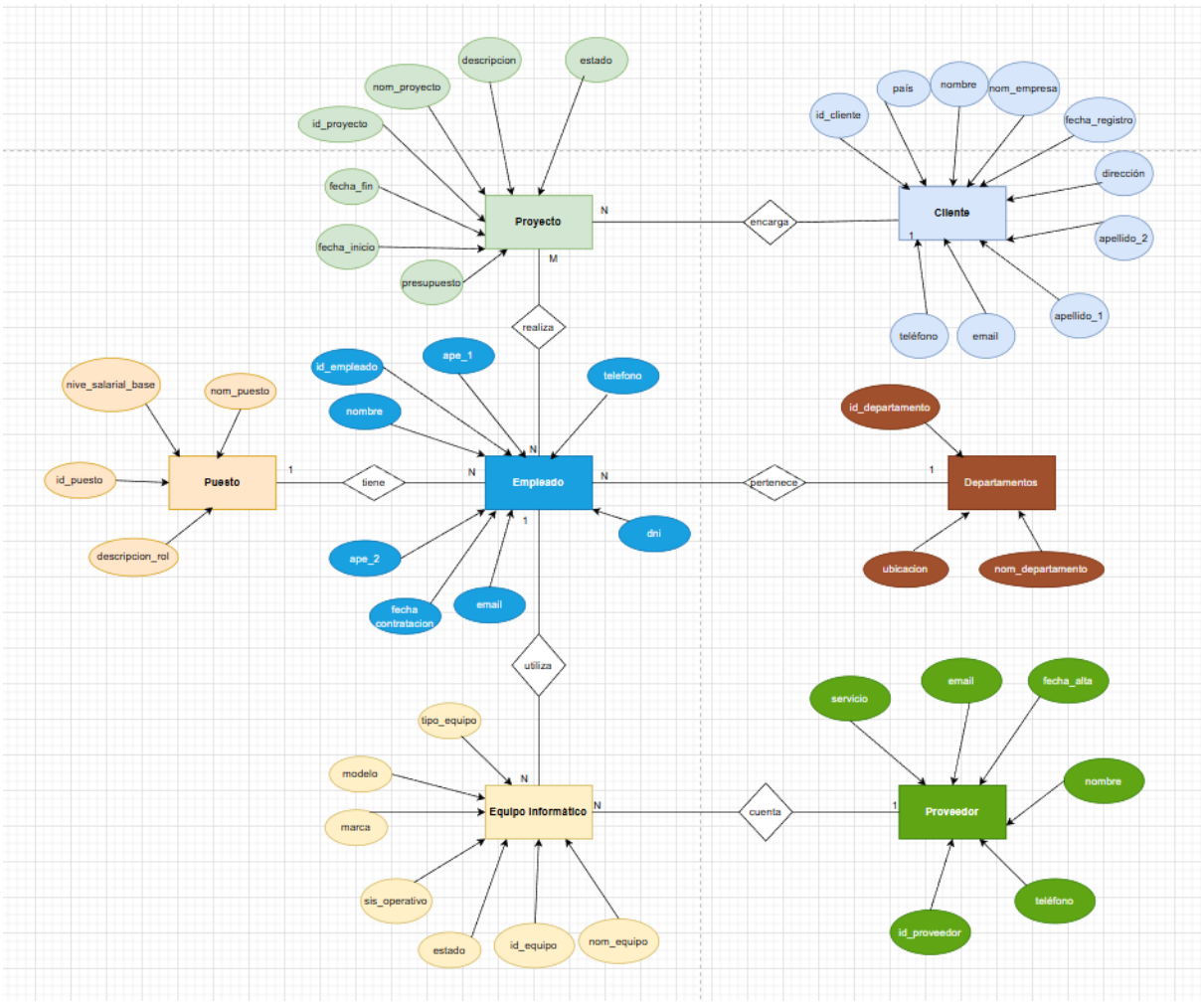
DOCUMENTO EXPLICATIVO DEL PROYECTO

1.- Modelo Entidad Relación:



MODELO RELACIONAL

Cliente (id_cliente, nombre, apellido_1, apellido_2, telefono, direccion, email, fecha_registro, nom_empresa)
Proyecto (id_proyecto, id_encargo, nom_proyecto, descricion, fecha_inicio, fecha_fin, estado, presupuesto)
Departamento (id_departamento, nom_departamento, ubicación)
Puesto (id_puesto, descripcion_rol, nivel_salarial_base, nom_puesto)
Empleado (id_empleado, id_departamento, id_puesto, nombre, apellido_1, apellido_2, DNI, telefono, email, fecha_contratacion)
Proyecto_Empleado (id_proyecto_empleado, id_proyecto, id_empleado)
Proveedor (id_proveedor, telefono, nombre, fecha_alta, servicio, email)
Equipo_informatico (id_equipo, id_proveedor, id_empleado marca, modelo, tipo_equipo, estado, nom_equipo, sis_operativo)

En este modelo representa la estructura de datos esenciales, para la gestión de proyectos, clientes y empleados dentro de una empresa de desarrollo de software.
La entidad cliente, es la persona o empresa que nos solicita o encarga un proyecto, hemos decidido que los atributos importantes son nombre, apellidos

CONSULTAS SQL BBDD.

SELECT + FROM + JOIN + ON + WHERE.
Esta primera consulta nos da información conectada de empleado y departamento, con esto podríamos ver nombre y apellidos de los empleados y a qué departamento pertenece. Con Select mostraremos lo que queremos ver, con From qué tabla sacaremos esta información con Join + on conectaremos ambas tablas y cómo lo haremos y con Where qué condición debe tener.

```
SELECT
e.nombre,
e.ape_1,
d.nombre_departamento
FROM
EMPLEADO e
JOIN
DEPARTAMENTO d ON e.id_departamento =  d.id_departamento;
```

	NOMBRE	APE_1	NOMBRE_DEPARTAMENTO
1	María	López	RRHH
2	Juanita	Ruiz	RRHH
3	Lucia	Mangel	RRHH
4	Pepe	Lopez	I+D
5	Ander	Murcia	I+D
6	Asier	Marc	I+D
7	Marta	Ramirez	Compras
8	Endika	Garcia	Compras
9	Juan	Santana	Compras
10	Jose Luis	Trigo	Comercio
11	Cecilio	Duarte	Comercio
12	Florentino	Sanchez	Finanzas
13	Vinicius	Sanchez	Comercio

En esta consulta podríamos añadir WHERE para buscar un empleado en concreto.

```
SELECT
e.nombre,
e.ape_1,
d.nombre_departamento
FROM
EMPLEADO e
JOIN
DEPARTAMENTO d ON e.id_departamento =  d.id_departamento
WHERE
e.nombre = 'Endika';
```

	NOMBRE	APE_1	NOMBRE_DEPARTAMENTO
1	Endika	Garcia	Compras

Con esta consulta vemos una tabla con el nombre de las columnas editados a como queramos, en este caso “nombre y apellidos”, “nombre de equipo”, y “Modelo”. Vemos estos tres atributos donde el apellido del empleado sea ‘murcia’.

```
SELECT
    e.nombre || ' ' || e.ape_1 AS "Nombre y apellidos",
    ei.nom_equipo AS "Nombre del Equipo",
    ei.modelo AS "Modelo"
FROM
    EMPLEADO e
JOIN
    EQUIPO_INFORMATICO ei ON e.id_empleado = ei.id_empleado
WHERE
    e.ape_1 = 'Murcia';
```

	Nombre y apellidos	Nombre del Equipo	Modelo
1	Ander Murcia	Sobremesa_TierAlto_Prog2	Precision 7865 Tower

Podríamos añadir en where un IN para que recoja mas apellidos y nos muestre su nombre completo, nombre del equipo y el modelo.

```
SELECT
    e.nombre || ' ' || e.ape_1 AS "Nombre y apellidos",
    ei.nom_equipo AS "Nombre del Equipo",
    ei.modelo AS "Modelo"
FROM
    EMPLEADO e
JOIN
    EQUIPO_INFORMATICO ei ON e.id_empleado = ei.id_empleado
WHERE
 e.ape_1 IN ('Murcia', 'Marc', 'Lopez');
```

	Nombre y apellidos	Nombre del Equipo	Modelo
1	Pepe Lopez	Sobremesa_TierAlto_Prog3	ThinkStation P620
2	Ander Murcia	Sobremesa_TierAlto_Prog2	Precision 7865 Tower
3	Asier Marc	Sobremesa_TierAlto_Prog1	Z2 Tower G8

Otro tipo de consultas podemos hacer una agregación para ver el salario promedio. Aquí nos interesa el salario promedio por departamento, asi sabemos que departamento recibe más salario o menos. Usamos Round para redondear.

```
SELECT
d.nombre_departamento AS Departamento,
 ROUND(AVG(e.salario), 0) AS Salario_Promedio
FROM
    EMPLEADO e
JOIN
    DEPARTAMENTO d ON e.id_departamento = d.id_departamento
GROUP BY
    d.nombre_departamento;
```

	DEPARTAMENTO	SALARIO_PROMEDIO
1	I+D	2950
2	Compras	2433
3	Comercio	2567
4	RRHH	2893
5	Finanzas	4200

```
SELECT SUM(presupuesto) AS Presupuesto_Total_Activo
FROM PROYECTO
WHERE estado = 'En curso';
```

PRESUPUESTO_TOTAL_ACTIVO
25000

```
SELECT e.nombre || ' ' || e.ape_1 AS Empleado_Sin_Equipo
FROM EMPLEADO e
JOIN DEPARTAMENTO d ON e.id_departamento = d.id_departamento
LEFT JOIN EQUIPO_INFORMATICO ei ON e.id_empleado = ei.id_empleado
WHERE ei.id_equipo IS NULL;
```

EMPLEADO_SIN_EQUIPO
Marta Ramirez
Cecilio Duarte
Endika Garcia
Florentino Sanchez
Jose Luis Trigo
Juan Santana

Con esta consulta podemos saber cuántos son los números de empleados en cada departamento.

```
SELECT
    p.nombre_puesto AS Puesto,
    COUNT(e.id_empleado) AS Total_Empleados
FROM
    PUESTO p
LEFT JOIN
    EMPLEADO e ON p.id_puesto = e.id_puesto
GROUP BY
    p.nombre_puesto
ORDER BY
    Total_Empleados DESC;
```

	PUESTO	TOTAL_EMPLEADOS
1	Programador	3
2	Materialista	3
3	Comercial	3
4	Contratista	2
5	Entrevistadora	1
6	Financiero	1

Para consultar qué empleados cobran más que su salario base según el puesto.

```
SELECT
    e.nombre || ' ' || e.ape_1 AS Empleado,
    e.salario AS Salario_Actual,
    p.nivel_salarial_base AS Base_Puesto
FROM
    EMPLEADO e
JOIN
    PUESTO p ON e.id_puesto = p.id_puesto
WHERE
    e.salario > p.nivel_salarial_base;
```

	EMPLEADO	SALARIO_ACTUAL	BASE_PUESTO
1	María López	2950	2800
2	Juanita Ruiz	2900	2400
3	Lucia Mangel	2830	2400
4	Marta Ramirez	2900	2200
5	Vinicius Sanchez	3900	1900

Si queremos ver un proveedor insertado en la base de datos, pero que todavía no nos ha suministrado ningún equipo informático, usaremos esta consulta

```
SELECT
    pr.nombre AS Proveedor_Sin_Suministro,
    pr.fecha_alta
FROM
    PROVEEDOR pr
LEFT JOIN
    EQUIPO_INFORMATICO ei ON pr.id_proveedor = ei.id_proveedor
WHERE
    ei.id_proveedor IS NULL;
```

	PROVEEDOR_SIN_SUMINISTRO	FECHA_ALTA
1	ServiNet	10/09/21

Consulta la cual lista los proyectos “En curso” y nos muestra todos los empleados relacionados al proyecto.

```
SELECT  P.nom_proyecto,  E.nombre,  E.ape_1
FROM  PROYECTO P
JOIN  PROYECTO_EMPLEADO PE ON P.id_proyecto = PE.id_proyecto
JOIN  EMPLEADO E ON PE.id_empleado = E.id_empleado
WHERE  P.estado = 'En curso'
ORDER BY  P.nom_proyecto;
```

Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 8 en 0,006 segundos

	NOM_PROYECTO	NOMBRE	APE_1
1	Bingo Colectivo	Pepe	Lopez
2	Bingo Colectivo	Ander	Murcia
3	Bingo Colectivo	Asier	Marc
4	Bingo Colectivo	Vinicius	Sanchez
5	Bingo Colectivo	Endika	Garcia
6	Bingo Colectivo	Juan	Santana
7	Bingo Colectivo	Florentino	Sanchez
8	Bingo Colectivo	Marta	Ramirez

JAVA/PROGRAMACIÓN

BINGO

Nuestro proyecto consiste en tres códigos fundamentales para crear un juego de red completa para jugar al Bingo:

1. *ServidorBingo.java*: El motor, el bombo y el que controla el juego.
2. *ManejadorCliente.java*: El canal de comunicación y el que valida cada canto.
3. *ClienteBingo.java*: La interfaz gráfica (GUI) y la lógica de marcado del cartón.

Lo que Más Tiempo Nos Llevó

Lo que más nos constó y nos consumió la mayor parte de nuestro tiempo fue garantizar que el juego funcionara de forma segura y justa condiferentes jugadores conectados al mismo tiempo.

- Variables Compartidas: En *ServidorBingo*, la gran dificultad fue controlar los estados globales del juego *(lineaGanada* y *bingoGanado*). Tuvimos que declararlas como volatile para asegurar que cada jugador de *ManejadorCliente* viera el estado más reciente y no se produjeran trampas ni errores de sincronización.

- Validación: En *ManejadorCliente*, el proceso de cantar línea o bingo era secuencial. El jugador enviaba el canto, el sistema paraba el juego, enviaba una pregunta de sostenibilidad al cliente y esperaba la respuesta. Tuvimos que gestionar el *cantoPendiente* y la *respuestaCorrectaPendiente* de forma segura en cada hilo individual para evitar que los mensajes de diferentes jugadores se mezclaran.

Las Mayores Dificultades Técnicas Específicas

- Validación Final: La función *procesarRespuesta()* en *ManejadorCliente* fue algo compleja. Tuvimos que programar una condición en la que el primer jugador en responder correctamente sería el primero en modificar el estado global del servidor. Si dos jugadores respondían a la vez, el sistema tenía que decidir instantáneamente quién ganaba la línea o el bingo.

- Diseño del Cartón: En *ClienteBingo*, aunque el marcado manual era simple, tuvimos que diseñar una clase *BingoCard* con una lógica en *comprobarLineaCompleta().* Decidimos no codificar las 12 líneas una por una, sino usar una lógica de índice inicial y paso para reutilizar el código en filas, columnas y diagonales.

- Números del Bombo: En *ServidorBingo*, el método *sortearNumeroUnico()* tenía que ser 100% fiable. Usamos un *HashSet* para registrar los números sorteados y asegurarnos de que la lógica de selección en el bucle *do-while* fuese a prueba de fallos para que nunca se repitiera un número.

Lo que Más Nos Gustó y nos Motivó

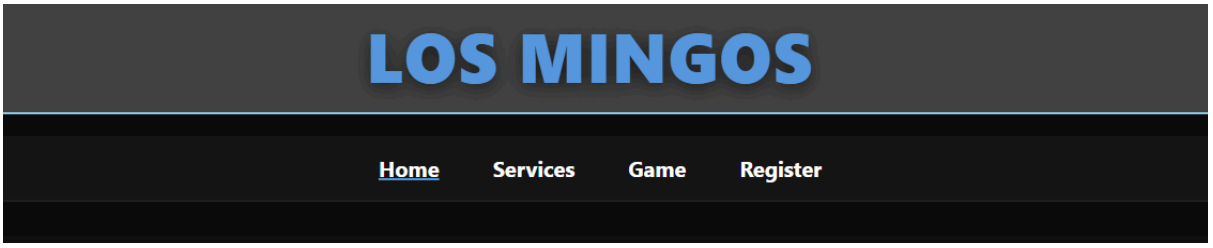
- La integración de las preguntas: Lo más curioso en nuestra opinión fue la implementación de las *PREGUNTAS_SOSTENIBILIDAD* en *ManejadorCliente*. Esto transformó un simple juego de red en un juego interactivo con un fin educativo.

- Comunicación entre jugadores: Nos gustó la sencillez del método *ServidorBingo.enviarMensajeATodos().* Después de toda la complejidad, con una sola línea de código podíamos comunicar cualquier cosa que ocurriese en el juego (como un número sorteado o un ganador) a todos los clientes simultáneamente.

El proyecto en general, fue un ejercicio complejo de programación, obligándonos a pensar cómo crear diferentes proyectos y enlazarlos entre ellos.

PÁGINA WEB EN INGLÉS

Nuestra página web consta de cuatro apartados principales que se pueden encontrar fácilmente en una barra debajo del título. Estos apartados tienen texto, links, imágenes, etc.



LO QUE MÁS TIEMPO NOS LLEVÓ HACER

La página tenía que ser en inglés, pero también queríamos ofrecer la posibilidad de que la página pudiese estar en Español, algo que nos costó bastante tiempo hacer:

- Al principio optamos por poner un botón en el que al hacer clic en él se cambiaba el idioma al Español o Inglés. Esto lo hicimos posible editando el html y poniendo el texto en dos idiomas diferentes en cada texto que quisiéramos traducir. Al ponerlo en práctica después de mucho esfuerzo, nos dimos cuenta de que había un error bastante notable, y es que cuando cambiabas de apartado se te ponía siempre en inglés, algo que no era práctico ya que queríamos que la página fuese cómoda y fácil de usar.

- Es por eso que dimos marcha atrás y borramos todos los textos que tuviesen doble idioma, y dejamos la página en Inglés. Más tarde, tuvimos la idea de hacer dos botones, uno para el idioma de Inglés y otro para el idioma de Español, lo que fué todo un éxito, ya que, gracias a esta idea, pudimos conseguir que nuestra página tuviese dos idiomas diferentes, y lo más importante, que estuviesen bien conectados.

- Ahora, al cambiar el idioma al Español y cambiar de apartado sigue apareciendo en Español. Ya que esos dos botones están conectados a dos diferentes html en diferentes idiomas, pero un mismo css. Es por eso que, teniendo la misma estructura, lo único que cambia es el idioma.

LAS MAYORES DIFICULTADES TÉCNICAS

Una de las cosas que más difícil nos resultó hacer fue el “back-to-top”.

Esto es un botón, que, al hacerle clic, lleva al usuario a la parte de arriba de la página, algo nuevo que no hemos visto. Lo que más nos costó configurar fué el css, ya que también queríamos que solo apareciese cuando el usuario estuviese en la parte de abajo y tuviese abierto un desplegable de los seis que pusimos. Algo que nos costó ver varios videos y, tras entenderlo, lo llevamos a cabo exitosamente.

LO QUE MÁS NOS GUSTÓ

Hay muchas partes del proyecto de la creación de nuestra página que nos han gustado, por ejemplo:

- **Crear el CSS general** → nosotros queríamos tener, por así decirlo, una parte del CSS que fuese general, esto quiere decir que todas los diferentes html que tenemos siguen un mismo patrón de diseño. Esto lo hemos hecho para que la página de la empresa se vea mucho más profesional y ordenada, todo el diseño tiene relación y sentido. Eso sí, cada html tiene un main diferente, esto por ejemplo si cambia dependiendo del CSS en el que estés.

- **Traducir los HTML** → traducir los html se nos hizo pesado, pero también tuvo su punto de diversión, ya que la mayoría de veces intentamos traducir del español al inglés de memoria y a veces nos confundimos y deciamos cosas que estaban mal, algo que nos causaba gracia.

- **Crear transiciones** → crear transiciones también nos resultó divertido, es algo que es relativamente sencillo de hacer (siempre y cuando te acuerdes del código), y encima se crean animaciones visuales que hacen que la página mejore mucho.