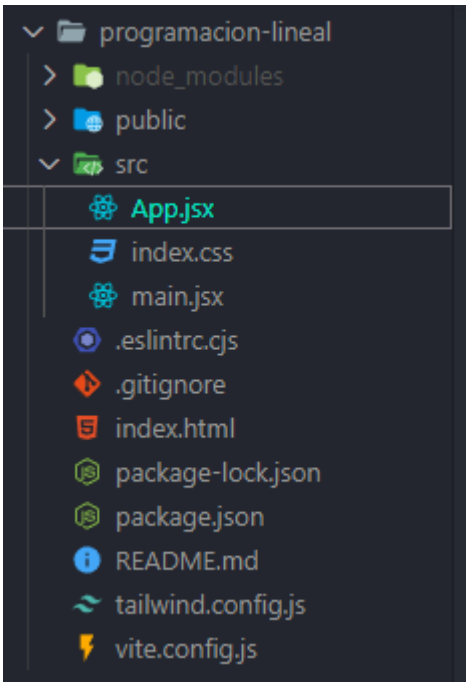
Documentación “Programación Lineal”

El presente documento representa la documentación del simulador de cartas, se eligió javascript como lenguaje de programación pues este fue es el lenguaje el cual todos los integrantes estamos familiarizados

El proyecto consiste en una aplicación de react, la misma cuenta con un único componente el cual contiene todo lo referente al problema, para ejecutar la aplicación se utiliza el comando “npm run dev” en la terminal

En el directorio podemos encontrar la siguiente estructura de archivos:



La parte más fundamental de la aplicación se encuentra en el componente App.jsx.

```jsx
import { useState } from "react";
import solver from "javascript-lp-solver/src/solver.js";

function App() {
    const [constraints, setConstraints] = useState({});
    const [variables, setVariables] = useState({});
    const [optimize, setOptimize] = useState("");
    const [results, setResults] = useState(null);

    const handleAddConstraint = () => {
        const constraintName = prompt(
            "Ingrese el nombre de la variable a restringir:"
        );
        const constraintValue = prompt("Ingrese el valor de la restricción:");
        setConstraints((prevConstraints) => ({
            ...prevConstraints,
            [constraintName]: { max: Number(constraintValue) },
        }));
    };

```

En estas primeras 20 líneas del código encontramos los imports necesarios para el correcto funcionamiento de la aplicación, encontramos también el inicio del componente principal, los múltiples estados para almacenar las restricciones, variables, variable a optimizar, los resultados de la optimización y por último la función encargada de manejar el evento de agregar una restricción, esta función le solicita al usuario el nombre y valor de la restricción y la agrega al estado de restricciones.

```
20
21  const handleAddVariable = () => {
22      const variableName = prompt("Ingrese el nombre del producto:");
23      const variableValues = {};
24      const variableValueCount = prompt(
25          "Ingrese la cantidad de valores que tendrá la variable:"
26      );
27
28      for (let i = 0; i < variableValueCount; i++) {
29          const valueName = prompt(
30              `Ingrese el nombre de la variable ${i + 1}:`
31          );
32          const value = prompt(`Ingrese el valor de la variable ${i + 1}:`);
33          variableValues[valueName] = Number(value);
34      }
35
36      setVariables((prevVariables) => ({
37          ...prevVariables,
38          [variableName]: variableValues,
39      }));
40  };
41
```

Posteriormente tenemos la función encargada de manejar el evento de agregar una variable, esta solicita al usuario el nombre, los valores y sus correspondientes valores de la variable y la agrega al estado de variables.

```
42      const handleSolve = () => {
43          const model = {
44              optimize,
45              opType: "min",
46              constraints,
47              variables,
48          };
49
50          const solverResults = solver.Solve(model);
51          setResults(solverResults);
52      };
```

Y por último encontramos la función encargada de manejar el evento de resolver el problema de optimización, esta crea el modelo de optimización con los datos de variables, restricciones y la variable a optimizar. Y utiliza el solver para resolver el modelo y actualiza los resultados en el estado. Posteriormente se encuentra el renderizado del componente el cual usa estilos de tailwind.