Documentación de lol-simulator

- 1. Descripción del Proyecto
- 2. Tecnologías Utilizadas
- 3. Herramientas de Desarrollo
- 4. Scripts
- 5. React + TypeScript + Vite
    - 5.1. Expanding the ESLint configuration

1. Descripción del Proyecto

lol-simulator es una aplicación web que simula el comportamiento de un juego en línea multijugador llamado "League of Legends" (LoL). La aplicación está diseñada para proporcionar una experiencia interactiva donde los usuarios pueden explorar diferentes aspectos del juego, como la selección de personajes, las mecánicas de juego y las estrategias de equipo.

2. Tecnologías Utilizadas

- **React**: Se utiliza como biblioteca principal para construir la interfaz de usuario de la aplicación, permitiendo una experiencia de usuario dinámica y receptiva.
- **React Router Dom**: Esta biblioteca se utiliza para gestionar las rutas de la aplicación y permitir la navegación entre diferentes vistas.
- **Styled Components**: Se utiliza para aplicar estilos CSS de manera modular y mantenible a los componentes de React.
- **Framer Motion**: Proporciona herramientas para crear animaciones fluidas y atractivas en la interfaz de usuario de la aplicación.
- **Axios**: Biblioteca para realizar solicitudes HTTP desde el cliente hacia el servidor, utilizada para obtener datos del juego o interactuar con una API externa.
- **LocalForage**: Proporciona una interfaz sencilla para almacenar datos en el navegador del usuario de manera persistente.
- **Match-sorter**: Utilizado para la clasificación y filtrado eficiente de datos, como la búsqueda de personajes dentro del juego.
- **Sort-by**: Biblioteca para ordenar colecciones de datos basadas en propiedades específicas.

3. Herramientas de Desarrollo

- **TypeScript**: Se utiliza para agregar tipado estático al código JavaScript, lo que mejora la robustez y la escalabilidad del proyecto.
- **ESLint**: Herramienta de linting para mantener un código limpio y consistente, con reglas específicas para TypeScript y React.
- **Vite**: Herramienta de construcción rápida para el desarrollo de aplicaciones web modernas, compatible con React y TypeScript.
- **@vitejs/plugin-react**: Plugin para Vite que facilita el desarrollo de aplicaciones React con Vite.

4. Scripts

- **dev**: Inicia un servidor de desarrollo utilizando Vite.
- **build**: Compila el código TypeScript y crea una versión optimizada para producción de la aplicación utilizando Vite.
- **lint**: Ejecuta ESLint para detectar y corregir problemas de estilo y errores en el código.
- **preview**: Inicia un servidor de vista previa utilizando Vite.

Esta documentación proporciona una descripción general de lol-simulator, incluyendo las tecnologías utilizadas, las herramientas de desarrollo y los scripts disponibles para la construcción y el mantenimiento del proyecto.

5. React + TypeScript + Vite

This template provides a minimal setup to get React working in Vite with HMR and some ESLint rules.

Currently, two official plugins are available:

- @vitejs/plugin-react uses Babel for Fast Refresh
- @vitejs/plugin-react-swc uses SWC for Fast Refresh

5.1. Expanding the ESLint configuration

If you are developing a production application, we recommend updating the configuration to enable type aware lint rules:

- Configure the top-level `parserOptions` property like this:

```
export default {
    // other rules...
    parserOptions: {
        ecmaVersion: 'latest',
        sourceType: 'module',
        project: ['./tsconfig.json', './tsconfig.node.json'],
        tsconfigRootDir: __dirname,
    },
};
```

- Replace `plugin:@typescript-eslint/recommended` to `plugin:@typescript-eslint/recommended-type-checked` or `plugin:@typescript-eslint/strict-type-checked`
- Optionally add `plugin:@typescript-eslint/stylistic-type-checked`
- Install eslint-plugin-react and add `plugin:react/recommended` & `plugin:react/jsx-runtime` to the `extends` list