**TRABAJO FINAL DE GRADO**

**ALPHADSGAME: UN JUGADOR ARTIFICIAL INTELIGENTE**

Por: Enrique Albaladejo Herrero

DNI: 53716332Q

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Año 2023

Ingeniería informática

*A mis padres, sin vosotros no habría llegado tan lejos*

*Gracias por todo*

Índice de contenido

**No se encontraron entradas de tabla de contenido.**

Índice de Imágenes

**No se encuentran elementos de tabla de ilustraciones.**

# Resumen

Este trabajo tiene como objetivo el desarrollo de un agente inteligente del juego ALPHADSGAME, así como una interfaz gráfica. Para ello, se ha decidido crear la parte gráfica del juego siguiendo la arquitectura cliente – servidor.

Para la parte del frontend he decidido utilizar [ReactJS](https://react.dev/), un framework de desarrollo web centrado en el desarrollo de interfaces por componentes. Gracias a esto, he conseguido modularizar distintos elementos de la interfaz, logrando mantener un código mucho más limpio y legible.

Por último, los datos se almacenarán en una base de datos de tipo relacional usando [MySQL](https://www.mysql.com/).

# Introducción

Dada la popularización de los videojuegos en la actualidad, estos se han convertido en un elemento fundamental de entretenimiento en la vida de muchos niños. Gracias al gran atractivo que estos generan, han ido surgiendo juegos que tienen como objetivo enseñar alguna habilidad. Un claro ejemplo es el “[Minecraft Education](https://education.minecraft.net/es-es)”, diseñado para ayudar a los estudiantes a explorar temas como historia, matemáticas y salud mental entre otros.

Con este mismo propósito, AlphaDSGame pretende familiarizar a los jugadores con el proceso de desarrollo de aplicaciones, los elementos necesarios para buscar el funcionamiento deseado de estas y el cumplimiento de los requisitos planteados al principio de su desarrollo.

## AlphaDSGame

A continuación, se detallará el funcionamiento del juego incluyendo sus reglas, elementos y las relaciones entre estos dentro de una partida.

AlphaDSGame es un juego de 4 jugadores donde cada uno tendrá que completar una misión con el objetivo de comunicar su circuito con el del siguiente jugador. Para ello, se disponen de una serie de elementos.

### Elementos

#### ­­Los componentes

Son las piezas que se colocan en el tablero con la finalidad de conectarse entre sí para formar un circuito que conecte con el tablero del siguiente jugador y así poder enviar información por el canal construido. Cada una de estas fichas tendrá un número de conexiones distintas que le permitirán conectarse con otras fichas. Además, al colocarlas se podrán rotar para que el jugado pueda elegir con qué posición colocarlas en el tablero.

Cabe destacar que cada componente tiene un valor de puntos asociado que variara en función del tipo, calidad y variante

Cada uno de estos componentes tiene un valor de puntos asociado que variara en función del tipo, calidad y variante. Los tipos en los que estarán clasificados son los normales y los especiales.

Los componentes normales son aquellos que pueden ser obtenidos por todos los jugadores. Varían según su número de conexiones y su material, este último indicará la calidad del componente.

* Componentes normales según número de conexiones: hay componentes de dos conexiones, de tres y de cuatro.

Imagen que contiene dibujo, señal

Descripción generada automáticamente

* Componentes según calidad: como se ha mencionado anteriormente, la calidad de un componente depende de su material. Los hay normales, de plata y de oro, calidad baja, media y alta respectivamente.

Icono

Descripción generada automáticamente

Los componentes especiales son aquellos que tienen el color de algún jugador, hay tres por cada jugador, tienen 4 conexiones, una en cada lado y tienen un título y descripción específicos. Habría que destacar que cada componente pertenece a un jugador y por lo tanto solo él podría comprar este componente y utilizarlo.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

#### El tablero

Todos los jugadores comenzaran la partida con un tablero propio en el cual habrá un solo componente inicial que viene por defecto. En el trascurso del juego, cada jugador tendrá que crear un circuito con los componentes que vaya consiguiendo en el mercado.

#### Los riesgos

Los riesgos son aquellos elementos peligrosos que aparecerán a lo largo de la partida en el mercado, se diferencian de los componentes porque son blancos, y bajo ellos aparece un botón de resolver, cuando aparece algún riesgo salta un modal que avisa al jugador de que debe resolver el riesgo lo antes posible, ya que, hasta que no se resuelven todos los riesgos el juego queda suspendido y ningún jugador podrá comprar ningún componente, el número de riesgos es limitado ya que hay 4 riesgos en el juego. Por último, cabe destacar que para resolver los riesgos se necesitan cartas de un subtipo específicas, las cuales se notificaran si mantenemos el cursor sobre el número de elementos necesarios para resolver este riesgo.

#### El mercado

El mercado es de donde salen los diversos componentes y riesgos que se necesitan comprar o resolver. Para ello se necesitarán unos elementos que hagan que se puedan comprar los diversos componentes, estos son las cartas. Bajo cada componente aparecerá una cajita con números, estos números representan la cantidad del tipo de cartas necesarias para poder comprar este componente especifico. En el caso del riesgo se indica el subtipo de cartas, así como la cantidad de ellas necesarias para poder resolver el riesgo. Cabe destacar que en ambos elementos aparece una sombra roja cuando no es posible comprar o resolver el objeto en cuestión.

#### Las cartas

Las cartas son los elementos que hacen posible la compra y resolución de los componentes y riesgos que existen en la aplicación. Hay tres tipos de cartas, las cartas de tipo “domain” (estas se dividen a su vez en “protected data”, “database”, y “team spirit”), tipo “computer science” (estas se dividen en “open source”, “antivirus” y “new technology”) y tipo “mathematics” (estas se dividen en “simple model”, “fast model” y “right model”). Cada vez que cada jugador haya jugado un turno se barajaran y repartirán cartas hasta que cada jugador se quede con 4.

Estos serian todos los elementos que forman parte del juego.

### Como se juega

En esta sección se explicará cómo se juega el juego que se va a implementar en físico.

Para comenzar son necesarios 4 jugadores, cada uno este encargado de una parte distinta en el desarrollo de la aplicación, estas partes se distinguen debido al color característico que tienen. Al comienzo de la partida, se reparten a cada jugador 4 cartas aleatoriamente y se ponen el resto encima de la mesa boca abajo. A continuación, se pone una tarjeta de mercado en el centro de la mesa y encima de los huecos se ponen 4 componentes que tienen costes distintos según la tarjeta, a continuación, si ha salido algún riesgo, se paraliza el juego y se intenta resolver el juego, mientras no se resuelva el juego, no se podrá continuar y lo único que podrán hacer los jugadores será descartarse e intercambiar cartas con el fin de solucionar el riesgo. Una vez se resuelve el riesgo se le da al jugador que lo resuelva unos puntos aleatorios. Si no hay riesgo, los jugadores tienen libertad para intercambiar cartas y comprar los componentes del mercado, no obstante, eso se deberá hacer en orden. Una vez comprados los componentes deberán ser ubicados en el circuito del jugador de forma que conecte con el resto de los componentes. Al terminar todos los jugadores su turno se volverá a repartir cartas y se colocaran nuevos componentes en el mercado hasta que haya 4 componentes normales. Además, la información que avanza por el circuito avanzará un componente en dirección al próximo jugador, es importante destacar que el jugador que disponga de este dato en su circuito deberá hacer lo que el grupo de jugadores decidan, perdiendo así la posibilidad de decidir que hacer. Estas acciones deben de ser las que hagan que el dato salga lo antes posible del circuito del jugador.

# Propuesta

En primera instancia estuve barajando diversas tecnologías en las cuales implementar el juego, pero, tras un tiempo me decidí por implementar el juego siguiendo la estructura de cliente servidor en la cual hay dos procesos corriendo, por una parte, está el frontend que se encarga de la parte gráfica, la del cliente y por otra la parte del servidor que se encarga del procesado de datos, además el backend está conectado a una base de datos donde se guardarán los datos de cada juego.

## Tecnologías frontend

Para el frontend he decidido utilizar ReactJS ya que al investigar un poco vi que era una manera bastante simple y limpia de crear interfaces visuales, que además comparte bastantes similitudes con React Native, otro framework de desarrollo de interfaces en el cual se pueden exportar a distintos tipos de plataforma. No obstante, dado mi conocimiento de HTML, CSS y JS opte por utilizar ReactJS ya que muchas de las bases ya las conocía y me servían para crear la interfaz de una manera más cómoda respecto a React Native.

Pero antes de nada voy a introducir un poco React y cómo funciona.

### ReactJS

Tal y como he explicado antes React es un framework que te permite hacer de forma sencilla interfaces para web. Este framework es un framework para JS y es capaz de modularizar los distintos elementos de la web en Componentes que luego se podrán utilizar en distintas partes de la interfaz aprovechando tanto el código como los estilos y los elementos HTML.

#### Componente

Un componente es un fragmento de la interfaz que se ha modularizado, siendo un ejemplo, un botón con un estilo o una funcionalidad más compleja. Estos componentes se pueden escribir de varias maneras, están los componentes de tipo clase y los componentes de tipo funcional, en esta aplicación he decidido utilizar componentes de tipo funcional. Estos se caracterizan en que a la hora de escribirlos

## Tecnologías backend

### Flask

### Python

## Tecnologías adicionales

### MySQL

### GitHub

## Proceso de desarrollo

### Frontend

#### Flujos

### Backend

#### Lógica de aplicación

#### Lógica de agente

### Etapas