****

**S.I.G.P.D.**

Programación Full Stack

JoJeRo

| ROL | C.I | APELLIDO | NOMBRE | E-MAIL |
| --- | --- | --- | --- | --- |
| Coordinador | 5.120.902-3 | Orcasberro | Jefferson | jeffersonorcasberrolopez@gmail.com |
| Subcoordinador | 4.910.640-5 | Falero | Joaquín | joaquinfalero01@gmail.com |
| Integrante 1 | 5.257.348-1 | Eiriz | Rodrigo | rodrigoeirizantunez@hotmail.com |
| Integrante 2 | 6.638.292-7 | Mikhailov | Iaromir | yarwq05@gmail.com |

**Docente: Méndez, Diego**



**I.T.I 3MF**

## 

## **Justificación Tecnológica**

### **Introducción**

El presente documento justifica la elección de tecnologías utilizadas en el desarrollo del proyecto **Draftosaurus Digital**, un videojuego web de tablero adaptado de su versión física. La selección tecnológica se realizó con base en criterios como compatibilidad, escalabilidad, agilidad en el desarrollo, experiencia de usuario, trabajo colaborativo y mantenimiento del código.

### **Tecnologías seleccionadas**

#### **HTML**

Se utiliza **HTML5** como lenguaje base para la estructuración del contenido del juego.

**Ventajas:**

* Permite jerarquizar correctamente los elementos del juego como: tablero, cartas, botones y menús.
* Proporciona **estructuración semántica**, facilitando el acceso al contenido tanto para desarrolladores como para tecnologías asistivas.
* **Compatibilidad amplia** con todos los navegadores modernos.
* Integración fluida con CSS, JavaScript y frameworks como Bootstrap.

#### **CSS**

CSS es utilizado para definir la **estética y experiencia visual** del juego, permitiendo diferenciar zonas temáticas y establecer estilos consistentes.

**Ventajas:**

* **Personalización total** de colores, tamaños, animaciones y estilos.
* **Diseño responsive**, adaptado a dispositivos móviles y de escritorio.
* Separación clara entre estructura (HTML) y presentación, lo que mejora la mantenibilidad.

**Bootstrap**

Bootstrap fue seleccionado como **framework frontend complementario**, debido a su eficiencia y versatilidad en el desarrollo de interfaces rápidas.

**Ventajas:**

* Ofrece componentes ya diseñados (botones, menús, alertas, modales), útiles para funciones como "tirar dado", mostrar puntuaciones o cambiar de ronda.
* Sistema de grillas responsive que se adapta fácilmente al diseño del tablero digital.
* Permite mantener **consistencia visual** en todas las secciones del juego.
* Facilita un **desarrollo ágil** gracias a su documentación clara y amplia comunidad.

**Desventajas:**

* Puede generar interfaces genéricas si no se personaliza adecuadamente.
* La personalización avanzada a veces requiere sobrescribir estilos preexistentes.

### **Herramientas de control de versiones y colaboración**

#### **Git y GitHub**

Para el control de versiones del código se eligió **Git** en conjunto con la plataforma **GitHub**.

**Ventajas:**

* Permite realizar seguimientos detallados de los cambios en el código.
* Posibilita la colaboración entre varios integrantes del equipo mediante ramas (*branches*) y solicitudes de extracción (*pull requests*).
* Ofrece control sobre versiones anteriores del proyecto, facilitando correcciones sin pérdida de trabajo.
* Integración con otras herramientas como Trello, Discord y GitHub Pages.

#### **Trello**

Se utilizó **Trello** como herramienta de organización y seguimiento de tareas, permitiendo gestionar cronogramas, responsabilidades y avances del equipo de forma visual y colaborativa.

### **Relación con los requerimientos del juego**

Cada tecnología fue seleccionada teniendo en cuenta las **necesidades específicas del videojuego Draftosaurus**, incluyendo:

| **Requerimiento** | **Tecnología relacionada** | **Justificación** |
| --- | --- | --- |
| Interfaz dinámica para colocar dinosaurios | HTML + CSS + Bootstrap | Permite crear elementos interactivos con estilo coherente |
| Diseño responsive para jugar desde distintos dispositivos | CSS + Bootstrap | Facilita adaptación sin duplicar esfuerzos |
| Interacción por botones (“tirar dado”, “pasar turno”) | HTML + Bootstrap | Componente de botón estilizado y funcional |
| Control de versiones y trabajo en grupo | Git + GitHub + Trello | Registro de cambios, colaboración y seguimiento del desarrollo |

### **Conclusión**

La combinación de **HTML, CSS y Bootstrap** permite desarrollar de forma eficiente y estructurada la interfaz del videojuego, mientras que **Git, GitHub y Trello** garantizan un trabajo colaborativo, controlado y organizado. Estas tecnologías cumplen con los requerimientos técnicos y funcionales del proyecto, optimizando el desarrollo y garantizando una experiencia de usuario consistente y accesible.

**Prototipo de Interfaz de Usuario**

**Maquetación**

El prototipo visual fue realizado en Figma, representando las principales vistas del sistema:

* **Pantalla de inicio**
* **Bienvenida al jugador**
* **Presentación del juego**
* **Login y Registro**
* **Tablero de juego** (con zonas de colocación)
* **Resultados y finalización de partida**

El diseño mantiene una estética **cyberpunk/neón**, con tipografía ***Outfit*** y colores brillantes sobre fondo oscuro. La estructura visual prioriza **usabilidad y claridad** para jugadores de todas las edades.

**Prototipado completo (dado, piezas, botones, tipografía, etc)**

**Pantallas e interfaz** 

**Tablero**

**HTML con semántica correcta**

Se utilizó HTML5 con etiquetas semánticas para estructurar el contenido, mejorando accesibilidad y legibilidad:

**<header>** <!-- menú superior --> **</header>**

**<nav>** <!-- barra de navegación --> **</nav>**

**<main>** <!-- contenido principal del juego --> **</main>**

**<section>** <!-- áreas del tablero, registro, login --> **</section>**

**<footer>** <!-- créditos del proyecto --> **</footer>**

Esta estructuración facilita futuras integraciones con frameworks o scripts.

**CSS con enfoque en usabilidad responsive**

El diseño de estilos CSS se centró en:

* Contraste alto (neón sobre fondo oscuro)
* Botones grandes y claros
* Layout adaptable a pantallas pequeñas
* Separación visual por secciones

Se usaron unidades relativas (em, %, vh, vw) para favorecer la adaptación entre resoluciones.

**Uso de Bootstrap (opcional)**

Bootstrap fue usado parcialmente para:

* Aplicar una grilla responsive (.container, .row, .col)
* Dar estilos rápidos a botones y formularios
* Agregar márgenes y espaciado sin escribir clases desde cero

**Navegación entre vistas (Prototipo + Código)**

Se implementó navegación básica en HTML con múltiples páginas enlazadas entre sí:

#### Estructura de vistas:

* **index.html** → Inicio
* **bienvenida.html** → Bienvenida al usuario
* **presentacion.html** → Introducción al juego
* **login.html** → Formulario de acceso
* **registro.html** → Alta de nuevo usuario
* **tablero.html** → Tablero de juego
* **resultados.html** → Resultados de la partida

Cada página incluye botones o enlaces que permiten avanzar o retroceder, simulando la lógica de flujo del juego.

**Diseño Responsivo**

**Enfoque Mobile First**

El diseño se pensó para dispositivos móviles primero (**max-width: 576px**), y luego se expandió a pantallas mayores.

@media (max-width: 576px) {

body {

font-size: 14px;

}

.dino-card {

flex-direction: column;

}

}

Se asegura una experiencia cómoda en resoluciones **320px a 576px**, con botones accesibles y textos legibles.

**Frameworks complementarios**

No se utilizaron otros frameworks como Tailwind para evitar agregar complejidad innecesaria en esta primera fase.

**Configuración del entorno de desarrollo**

**Herramientas utilizadas**

* IDE: Visual Studio Code
* Extensiones útiles:  
  + Live Server
  + HTML CSS Support
  + Prettier
* Sistema operativo: Windows 10

Nota: la instalación del entorno será detallada en la segunda entrega según el profesor.

**Control de Versiones**

**Repositorio Git**

Se creó un repositorio común en GitHub para alojar:

* Código fuente (HTML, CSS)
* Archivos de diseño y documentación
* Recursos gráficos (dinosaurios, tablero, dado)

🔗 Link: *(*[*https://github.com/yarwq/Proyecto\_Final*](https://github.com/yarwq/Proyecto_Final)*)*

**Justificación del uso de GitHub**

GitHub fue elegido por su:

* Accesibilidad desde navegador
* Integración con VS Code
* Facilidad para gestionar ramas y cambios
* Soporte para equipos pequeños con historial compartido

Se tomó como referencia el siguiente video:  
<https://www.youtube.com/watch?v=VdGzPZ31ts8>

**Convenciones de commits**

Se establecieron nombres claros y descriptivos para cada commit, por ejemplo:

* **feat**: diseño de la pantalla de bienvenida
* **fix**: error en maquetación responsive
* **docs**: se agrega justificación tecnológica

**Contenido del repositorio**

**/docs**: documentación técnica

**/assets**: imágenes y diseños

**/src**: código HTML y CSS

**README.md**: descripción del proyecto, autores e instrucciones