PROYECTO DE CICLO DE G.S. DE
Desarrollo de Aplicaciones Web

Campfire Adventure

Carlos Gonzalez Garcia
CURSO 2023/24

Índice

1 Introducción

1.1 Datos del proyecto

Nombre	Apellidos	Título del proyecto	Ciclo	Año	Centro Educativo
Carlos	Glez. Garcia	Campfire Adventure	DAW	2024	IES Virgen del Carmen

1.2 Objetivos

Creación de una página web que sirva como “punto de encuentro” para grupos de rol donde ver estadisticas como tiempo jugado, personas participantes en las sesiones y ficha del personaje.

1.2.1 Objetivos Previstos

- Funciones de administrador para que el “master” (que es la persona encargada de organizar partidas de rol) pueda tener un control sobre sus jugadores.
- Identificacion entre usuarios para redireccion a unas paginas o funcionalidades concretas
- Log-in y un sign-up para poder registrarse y acceder a los datos pertinentes de cada sesión.
- Creacion de estadisticas, registros y funcionalidades de creación, elmininacion o edicion de los mismos para dejar reflejado la información de cada jugador.
- Busqueda de usuarios
- Sistema de tirada de dados

1.2.2 Objetivos No Cumplidos

- Capacidad de elección del juego para variar la tematica visual de la página.
- Creacion de la parte de Jugador donde poder registrar la ficha de juego y una parte de anotaciones donde escribir información determinada del juego.
- Poder diferenciar entre usuario “Master” y “Jugador” para que cada uno vea unas secciones determinadas
- Sistema de tirada de dados

1.3 Idea Del Proyecto

El concepto de Campfire Adventure nace de la necesidad de tener una plataforma online donde alojar las fichas de personajes, y otra información extra necesaria, de las partidas de rol. Así mismo se intenta cubrir la necesidad de no tener que estar transportando, usando y almacenando cantidades significativas de papel, y otros enseres fisicos, de manera innecesaria sin tener en cuenta una perspectiva sostenible.

Con Campfire Adventure podras tener una sesión de rol donde sea, siempre que haya una conexión a internet, ya que tendras acceso a toda la información necesaria de los personajes, historia y unos sistema de tirada de dados.

2 Tecnologías

En base a las tecnologias actuales, y manteniendo una visión de sostenibilidad y facilidad de mantenimiento en el codigo se optó por usar JavaScript de manera integra para la creación tanto del Front-end como del Back-end.

Para la parte del Front-end utilizamos el framework de ***React*** mientras que para la parte del Back-end usaremos ***NextJs***

Asi mismo usaremos ***TypeScript*** para añadir fucionalidades extra a Javascript, un tiapdo estatico de nuestras variables, componentes, etc…

Ademas usaremos la plataforma de ***Vercel***, con la tecnologia que ofrecen, para la gestión de las bases de datos. Esta plataforma nos da opción a escalar nuestra aplicación de otras maneras que en un futuro quizas puedan ser de utilidad.

2.0.1 React

Nos permite granular nuestra pagina en componentes permitiendo, junto con NextJs, la creación de una SPA (Single Page Application), donde iremos modificando el body de nuestro html en función de la navegación del usuario mediante llamadas a APIs de JavaScript de la parte del servidor.

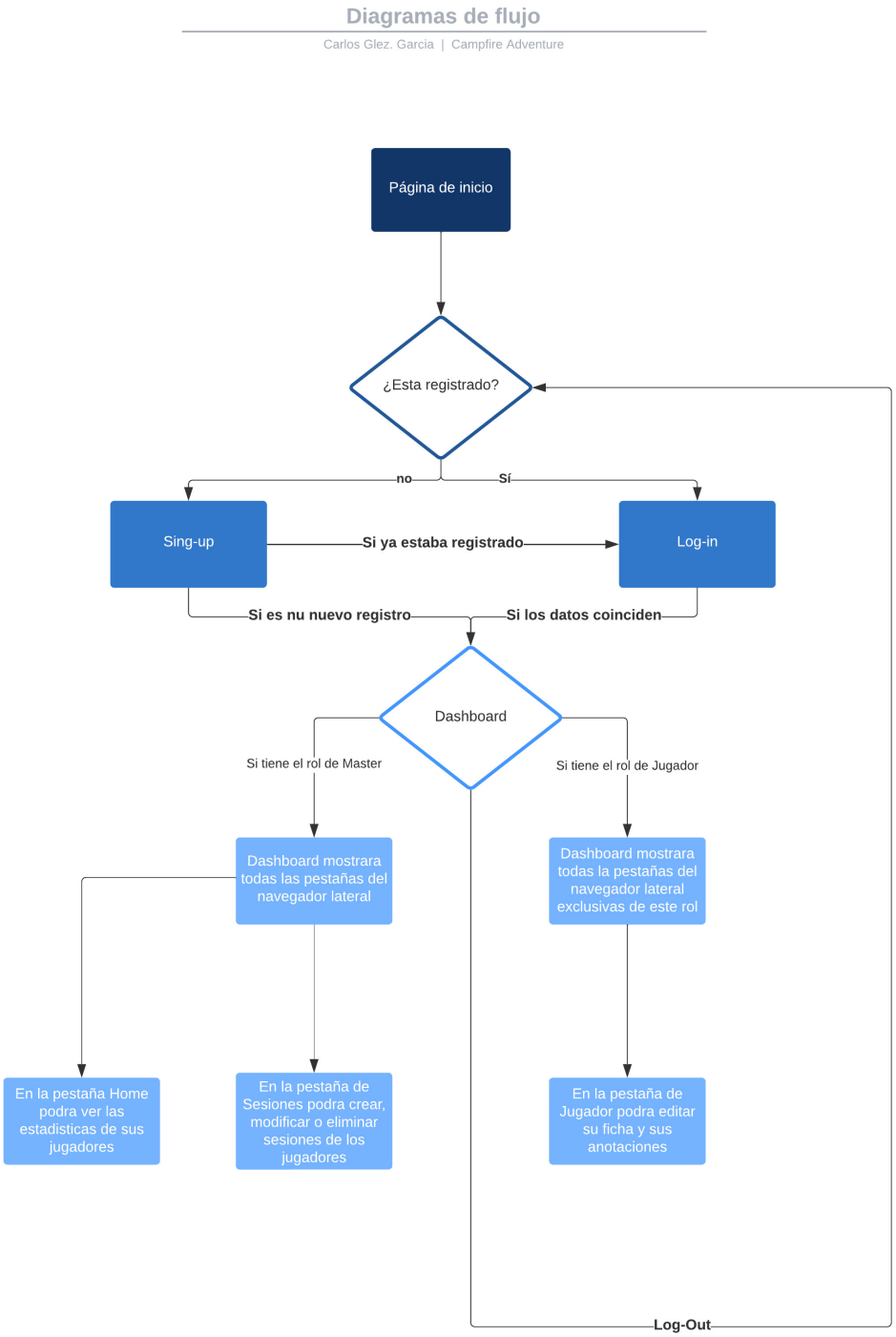
2.0.2 NextJs

Es el framework de JavaScript que perminte crear, y diferenciar, componentes que seran exclusivos del lado del cliente y servidor. Dando la funcionalidad de Back-end a Javascript y que, combinandola con React, nos ayudara a hacer nuestra pagina.

2.0.3 Typescript

Es un lenguaje de programación que nos ayudara ya que agrega funcionalidades extra, y como es complementarioa a Javascript sirve tanto para Front-end como para Back-end.

3 Análisis


Diagramas de flujo
Carlos Glez. Garcia | Campfire Adventure
Página de inicio
¿Esta registrado?
no
Sí
Sing-up
Si ya estaba registrado
Log-in
Si es nu nuevo registro
Si los datos coinciden
Dashboard
Si tiene el rol de Master
Si tiene el rol de Jugador
Dashboard mostrara todas las pestañas del navegador lateral
Dashboard mostrara todas la pestañas del navegador lateral exclusivas de este rol
En la pestaña Home podra ver las estadisticas de sus jugadores
En la pestaña de Sesiones podra crear, modificar o eliminar sesiones de los jugadores
En la pestaña de Jugador podra editar su ficha y sus anotaciones
Log-Out

4 Implementación

4.0.1 Creación de tablas

```
    const { db } = require('@vercel/postgres');
const {
  sheets,
  players,
  hours,
  users,
} = require('../app/lib/placeholder-data.js');//Desde aquí vamos a traer los datos que queremos introducir en nuestra bbdd

const bcrypt = require('bcrypt');

async function seedUsers(client) {
  try {
    await client.sql`CREATE EXTENSION IF NOT EXISTS "uuid-ossp"`;
    // Creamos la tabla "users" en caso de que no exista
    const createTable = await client.sql`
      CREATE TABLE IF NOT EXISTS users (
        id UUID DEFAULT uuid_generate_v4() PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
        email TEXT NOT NULL UNIQUE,
        password TEXT NOT NULL
      );
    `;
    // Insertamos datos en la tabla "users"
    const insertedUsers = await Promise.all(
      users.map(async (user) => {
        const hashedPassword = await sha256(user.password);
        return client.sql`
        INSERT INTO users (id, name, email, password)
        VALUES (${user.id}, ${user.name}, ${user.email}, ${hashedPassword})
        ON CONFLICT (id) DO NOTHING;
      `;
      }),
    );
    return {
      createTable,
      users: insertedUsers,
    };
  } catch (error) {
    console.error('Error seeding users:', error);
    throw error;
  }
}

async function seedSheet(client) {
```

```
  try {
    await client.sql`CREATE EXTENSION IF NOT EXISTS "uuid-ossp"`;

    // Creamos la tabla "sheets" si no existe
    const createTable = await client.sql`
    CREATE TABLE IF NOT EXISTS sheets (
    id UUID DEFAULT uuid_generate_v4() PRIMARY KEY,
    player_id UUID NOT NULL,
    amount INT NOT NULL,
    date DATE NOT NULL
  );
`;

    // Insertamos datos a la tabla "sheets"
    const insertedInvoices = await Promise.all(
      sheets.map(
        (sheet) => client.sql`
        INSERT INTO sheets (player_id, amount, date)
        VALUES (${sheet.player_id}, ${sheet.amount}, ${sheet.date})
        ON CONFLICT (id) DO NOTHING;
      `,
      ),
    );

    return {
      createTable,
      sheets: insertedInvoices,
    };
  } catch (error) {
    console.error('Error seeding sheets:', error);
    throw error;
  }
}

async function seedPlayers(client) {
  try {
    await client.sql`CREATE EXTENSION IF NOT EXISTS "uuid-ossp"`;

    // Creeamos la tabla "players" en caso de que no exista
    const createTable = await client.sql`
      CREATE TABLE IF NOT EXISTS players (
        id UUID DEFAULT uuid_generate_v4() PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
        email VARCHAR(255) NOT NULL,
        image_url VARCHAR(255) NOT NULL,
        status VARCHAR(255) NOT NULL
      );
    `;

    // Insertamos datos en la tabla "players"
    const insertedPlayers = await Promise.all(
```

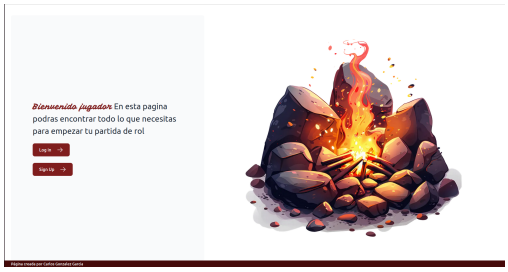
```
96        players.map(
97          (player) => client.sql`
98          INSERT INTO players (id, name, email, image_url, status)
99          VALUES (${player.id}, ${player.name}, ${player.email}, ${player
                .image_url}, ${player.status})
100         ON CONFLICT (id) DO NOTHING;
101       `,
102       ),
103     );
104
105     return {
106       createTable,
107       players: insertedPlayers,
108     };
109   } catch (error) {
110     console.error('Error seeding players:', error);
111     throw error;
112   }
113 }
114
115 async function seedHours(client) {//swap hours for hou
116   try {
117     // Creamos la base de datos de "hours" si no existe
118     const createTable = await client.sql`
119       CREATE TABLE IF NOT EXISTS hours (
120         month VARCHAR(4) NOT NULL UNIQUE,
121         hours INT NOT NULL
122       );
123     `;
124
125     // Introducimos los datos en la tabla "hours"
126     const insertedHours = await Promise.all(
127       hours.map(
128         (hour) => client.sql`
129         INSERT INTO hours (month, hours)
130         VALUES (${hour.month}, ${hour.hours})
131         ON CONFLICT (month) DO NOTHING;
132       `,
133       ),
134     );
135
136     return {
137       createTable,
138       hours: insertedHours,
139     };
140   } catch (error) {
141     console.error('Error seeding hours:', error);
142     throw error;
143   }
144 }
145
```

```
async function main() {
  const client = await db.connect();

  await seedUsers(client);
  await seedPlayers(client);
  await seedSheet(client);
  await seedHours(client);

  await client.end();
}

main().catch((err) => {
  console.error(
    'An error occurred while attempting to seed the database:',
    err,
  );
});
```

En las lineas anteriores vemos la creación de las tablas mediante un archivo, el cual usamos mediante un script añadido a nuestro “package.json”

```
Script de creación de tablas
"scripts": {
    "seed": "node -r dotenv/config ./scripts/seed.js"
  }
```

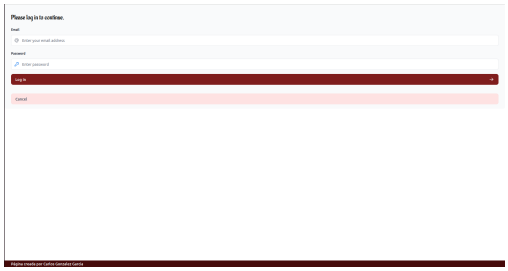
4.0.2 Pagina principal




```
Nuestra pagina princpial, la que ira cambiado su body atraves del
    componente children.

export default function RootLayout({
  children,
}: {
  children: React.ReactNode;
}) {
  return (
    <html lang="en" className=''>
      <body className=' grid-cols h-screen'>
        {children}
```

```
        </body>
        <footer className='space-y-5 flex h-5 fixed items-center inset-x-0 bottom-0 self-start bg-red-950 px-6 py-3 text-sm font-medium text-white'>
          Página creada por Carlos Gonzalez Garcia
        </footer>
      </html>
    );
}
```

4.0.3 Login y validación



```
Se implemento un sistema de validación con la libreria Zod, aquí vemos un esquema generico y dos modificados según la necesidad de crear o modificar una tabla.

/* Creamos un "esquema" de validación de datos que vamos a mandar a la BBDD
 */
const Schema = z.object({
    id: z.string(),
    playerId: z.string(),
    amount: z.coerce.number(),
    status: z.string(),
    date: z.string()
})

//Modificamos el esquema anterior para cada caso de Creación, Update...
const CreateSheetSchema = Schema.omit({
    id: true,
    date: true,
    status: true
})

const UpdateSheets = Schema.omit({
    id: true,
    date: true,
    status: true
});
```

```
Para el uso de cookies usamos el siguiente patrón, que se reutilizara
   para las diversas cookies que queramos usar.
//login
export async function createCookie(formData: FormData) {


    const name = 'Login'
    const value = formData.get('email') as string;

    cookies().set({
        name: name,
        value: value,
        secure: true
    })

}

export async function deleteCookie() {

    const name = 'Login'

    cookies().delete(name)

}

export async function validateCookie() {
    const exist = cookies().has('Login')

    return exist
}
```

```
Aquí esta el tipado de todas las clases que vamos a usar.

export type User = {
  id: string;
  name: string;
  email: string;
  password: string;
};

export type Player = {
  id: string;
  name: string;
  email: string;
  image_url: string;
  status: 'jugador' | 'master';
};

export type Sheet = {
  id: string;
```

```
  player_id: string;
  amount: number;
  date: string;
};

export type Hours = {
  month: string;
  hours: number;
};

export type LatestSheet = {
  id: string;
  name: string;
  image_url: string;
  email: string;
  amount: number;
};

export type LatestSheetRaw = Omit<LatestSheet, 'amount'> & {
  amount: number;
};

export type SheetsTable = {
  id: string;
  customer_id: string;
  name: string;
  email: string;
  image_url: string;
  date: string;
  amount: number;
  status: 'jugador' | 'master';
};

export type PlayersTableType = {
  id: string;
  name: string;
  email: string;
  image_url: string;
  total_Sheets: number;
  total_jugador: number;
  total_master: number;
};

export type FormattedPlayersTable = {
  id: string;
  name: string;
  email: string;
  image_url: string;
  total_Sheets: number;
  total_jugador: string;
  total_master: string;
```

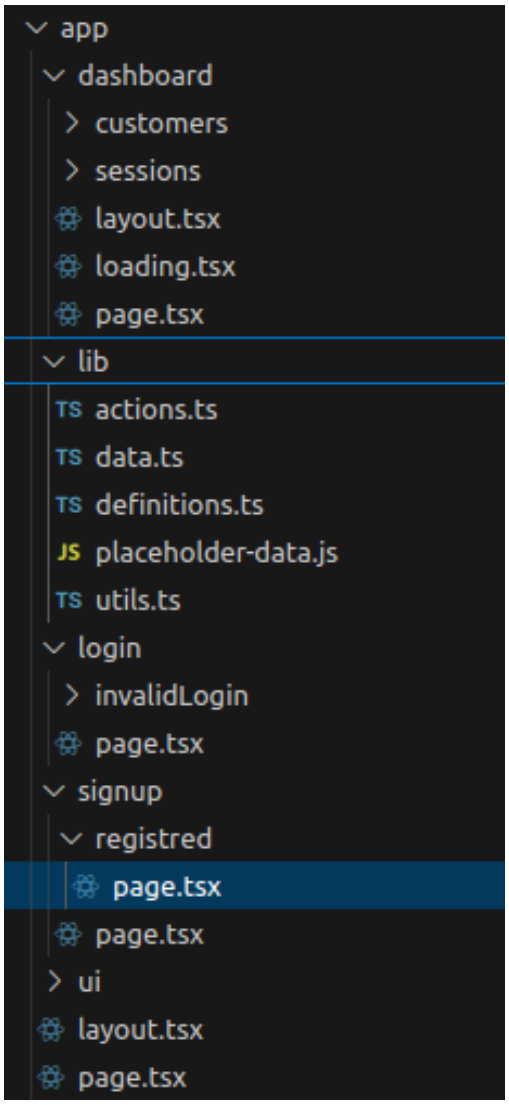
```
};

export type PlayerField = {
  id: string;
  name: string;
};

export type SheetForm = {
  id: string;
  player_id: string;
  amount: number;
};
```

4.0.4 Funcionalidades

```
En el archivo utils.ts tenemos estas funciones que usamos para algunas
    de las funcionalidades del proyecto, como la generación de tablas

export const formatHour = (amount: number) => {
  return (
    (Math.round(amount / 24))
  );
};

export const formatDateToLocal = (
  dateStr: string,
  locale: string = 'en-US',
) => {
  const date = new Date(dateStr);
  const options: Intl.DateTimeFormatOptions = {
    day: 'numeric',
    month: 'short',
    year: 'numeric',
  };
  const formatter = new Intl.DateTimeFormat(locale, options);
  return formatter.format(date);
};

 export const generateYAxis = (hours: Hours[]) => {
  const yAxisLabels = [];
  const highestRecord = Math.max(...hours.map((month) => month.hours));
  const topLabel = Math.ceil(highestRecord / 1000) * 1000;

  for (let i = topLabel; i >= 0; i -= 1000) {
    yAxisLabels.push(`${i / 1000}h`);
  }

  return { yAxisLabels, topLabel };
};
```

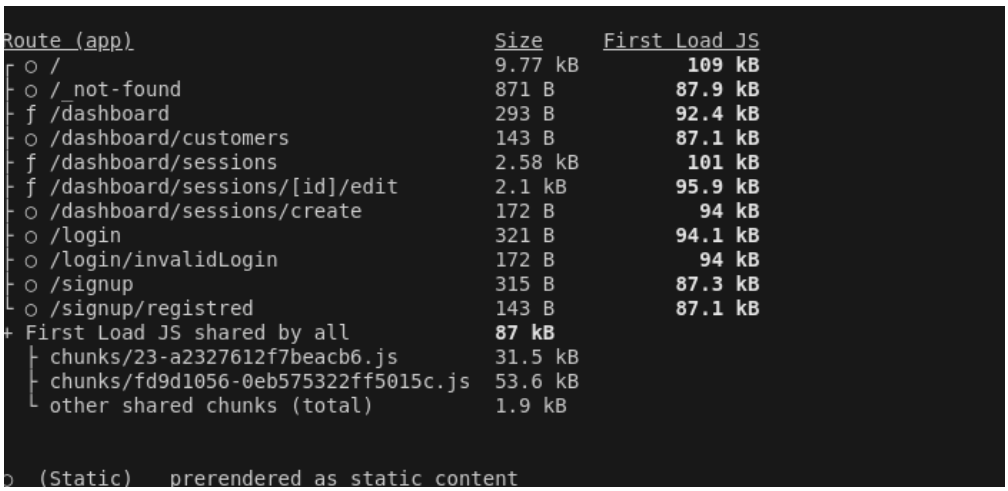
```
34
35 export const generatePagination = (currentPage: number, totalPages:
       number) => {
36
37   if (totalPages <= 7) {
38     return Array.from({ length: totalPages }, (_, i) => i + 1);
39   }
40
41   if (currentPage <= 3) {
42     return [1, 2, 3, '...', totalPages - 1, totalPages];
43   }
44
45   if (currentPage >= totalPages - 2) {
46     return [1, 2, '...', totalPages - 2, totalPages - 1, totalPages];
47   }
48
49   return [
50     1,
51     '...',
52     currentPage - 1,
53     currentPage,
54     currentPage + 1,
55     '...',
56     totalPages,
57   ];
58 };
```

4.0.5 Estructura de la App



Aquí podemos ver la estructura de nuestra pagina. Con NextJs no necesitamos un archivo de rutas como tal ya que por cada carpeta que contiene un archivo “page.tsx” el framework sabe que es una ruta a seguir.

De manear interna Nextjs entiende que la estructura de nuestro proyecto es así:



Como podemos ver se omiten las carpetas de lib y ui, donde hay recursos funcionales de la app, quedando solo las carpetas (rutas) donde hay un page.tsx

Si vien es cierto que tener que nombrar un archivo de la misma manera puede ser lioso a simple vista dentro del mismo podremos llamar a la función principal de manera diferenciadora como podemos ver en el codigo siguiente

```
1 export default function PageLoging() {
2     return (
3     <LoginForm/>
4     );
5 }
```

5 Conclusiones

- Durante el proceso de creación de mi app he podido ver los vacios de conocimientos respecto a lo que despligue de aplicaciones se refiere. He aprendido a entender, y usar mejor, los package.json.
- Así mismo he aprendido de cero los fundamentos basicos de los frameworks de ***React*** y ***NextJs*** así como he empezado a sumergirme en la versatilidad de los mismos
- He perfilado conocimientos del uso de las bases de datos y descubierto modelos que estudiar, y añadirlo a mis conocimientos, como el ***ORM***
- En definitiva: La creación de un app mia, junto con las practicas realizadas en empresas, me han mostrado el basto mundo que es la programación y ciertos caminos a tomar que creo que me gustarian, y en los que desarrollarme profesionalmente

6 Bibliografía

Apunta aquí cada Web, tutorial, vídeo que veas y qué has aprendido con él:

Tutorial de React desde cero: https://www.youtube.com/watch?v=7iobxzd_2wY&list=PLUofhDIg_38q4D0xNWp7FEHOTcZhjWJ29.

Documentación de React: https://legacy.reactjs.org/docs/getting-started.html.

Tutorial de Nextjs: https://www.youtube.com/watch?v=ZjAqacIC_3c&list=PLC3y8-rFHvwjOKd6gdf4QtV1uYNiQnruI&ab_channel=Codevolution.

Otros puntos de vista de como afrontar una duda de codigo expuestos por diversos usuaros del conocido foro StackOverflow: https://stackoverflow.com/,

Documentación de Next.js: https://nextjs.org/docs.

Documentación de Tailwind: https://tailwindcss.com