



**Prueba Técnica**  
Desarrollador de Software Junior

Rev3.02 – 2023  
jy/JPM

Prueba técnica DT - Frontend

PAOLA PACHECO  
MORENO

CL:3209887626

DataTraffic (Carryt.co).

BOGOTÁ- 2023

**Objetivo:**

El objetivo de esta evaluación es poner a prueba las competencias, fortalezas y debilidades técnicas de l@s postulantes al rol de desarrollador(a) frontend o fullstack JS.

**Alcance:**

Queremos saber si tienes las capacidades técnicas que se requerirán en un rol como desarrollador@ en Datattraffic. El porcentaje de éxito de contratación de candidat@s que hacen la prueba técnica es del 50%.

Consideramos que alguien con experiencia puede ejecutar la prueba en dos horas de trabajo, después de haberla leído, pero sabemos que las personas aprenden y retienen conocimientos de maneras diferentes, por lo que es posible que tome más o menos tiempo.

En <https://gitlab.com/jjyopez/rick-y-morty-api> hay un código, con una aplicación desarrollada en Javascript y Node.js

La aplicación ya es funcional y se puede descargar y usar. Esa aplicación consulta un API remoto que tiene la información de personajes, episodios y ubicaciones.

La información de ese API remoto se ha almacenado localmente en una base de datos del sqlite, disponible en `server/data/rickandmorty_v1.db`.

Las tres entidades ofrecidas por el API remoto son:

"characters": "<https://rickandmortyapi.com/api/character>",

"locations" : "<https://rickandmortyapi.com/api/location>",

"episodes" : <https://rickandmortyapi.com/api/episode>

**Tarea:**

Construir una página adicional en ReactJS, con un mapa de calor, dónde se vean en diferentes colores la

cantidad de personajes que salen en cada episodio. Modele las filas como temporadas, las columnas como

episodios y en el contenido el dato con la cantidad de personajes que aparecen.

Bono extra: Realizar el cambio general de la apariencia del software. A mayor cambio, mayor valor adicional.

Forma de entrega: Crear un repositorio público en bitbucket/github/gitlab con la aplicación modificada. Incluir instrucciones para ejecución en el Readme.md; además, redactar un documento con el resumen del proceso de análisis, entendimiento, desarrollo y complicaciones encontradas en la resolución del reto..}

## DESARROLLO

Descripción de los requerimientos funcionales:

**NOMBRE DEL SOFTWARE:** El informante.edu

A continuación, se muestra la App.js quien hace el llamado de cada uno de los componentes: Characters, Location y Episodios.

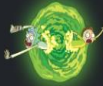
```
App.js / App
import React from 'react'
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom'

import Home from './Pages/Home'
import Navbar from './components/Navbar/Navbar'
import Episodes from './Pages/Episodes'
import Location from './Pages/Location'
import CardDetails from './components/Card/CardDetails'
/**
 * It returns Router
 * @return {void}.
 */
function App() {
  return (
    <Router>
      <div className="App">
        <Navbar />
      </div>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/:id" element={<CardDetails />} />

        <Route path="/episodes" element={<Episodes />} />
        <Route path="/episodes/:id" element={<CardDetails />} />

        <Route path="/location" element={<Location />} />
        <Route path="/location/:id" element={<CardDetails />} />
      </Routes>
    </Router>
  )
}


export default App
```




# El Rick & Morty

[Episodio](#)
[Caracteres](#)
[Ubicación](#)


## Caracteres



**Rick Sánchez**  
Última ubicación  
Ciudadela de Ricks



**Morty Smith**  
Última ubicación  
Ciudadela de Ricks



**Verano Smith**  
Última ubicación  
Tierra (dimensión de reemplazo)


Filtros

[Todos los filtros](#)


Estado ▼

Especie ▼


Género ▼




**Amish Cyborg**  
Última ubicación  
Tierra (dimensión de reemplazo)




**Annie**  
Última ubicación  
Parque de Anatomía



**Antena Morty**  
Última ubicación  
Ciudadela de Ricks

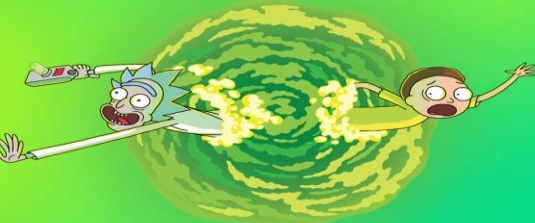


**Antena Rick**  
Última ubicación  
desconocido



**Hormigas en mis ojos Johnson**  
Última ubicación  
Interdimensional Cable

[Anterior](#)
[1](#)
[2](#)
[...](#)
[41](#)
[42](#)
[Próximo](#)



te es un proyecto que manipula la API de Rick y Morty. Es una API pública, y este proyecto no tiene fines de lucro.

Desenvolvedor : Paola Pacheco Moreno

## 1. El componente Charcters y/o Home, muestra todos los personajes almacenados en la api:

"characters": "https://rickandmortyapi.com/api/character",

```

s JS Navbars JS Pagination.js JS Home.js x
es > JS Home.js > ...
import React, { useState, useEffect } from 'react'

import Search from '../components/Search/Search'
import Card from '../components/Card/Card'
import Pagination from '../components/Pagination/Pagination'
import Filter from '../components/Filter/Filter'
export default function Home() {

  let [pageNumber, setPageNumber] = useState(1);
  let [status, updateStatus] = useState('');
  let [gender, updateGender] = useState('');
  let [species, updateSpecies] = useState('');
  let [search, setSearch] = useState('');
  let [fetchData, updateFetchData] = useState([]);
  let { info, results } = fetchData;

  let api = `https://rickandmortyapi.com/api/character/?page=${pageNumber}&name=${search}&status=${status}&gender=${gender}&species=${species}`;

  useEffect(() => {
    ;(async function () {
      const data = await fetch(api).then((res) => res.json())
      updateFetchData(data)
    })()
  }, [api])

  return (
    <div className="App">
      <h1 className="text-center mb-3">Characters</h1>
      <Search setPageNumber={setPageNumber} setSearch={setSearch} />
      <div className="container">
        <div className="row">
          <div className="col-lg-8 col-12">
            <div className="row">
              <Card page="/" results={results} />
            </div>
          </div>
        </div>
      </div>
    </div>
  )
}

```



Adicional a esto, el sistema permite filtrarlos por Status, Gender, Species, Búsqueda por name y filtro total.

Status esta filtrado por color, verde es vivo, muerto: rojo y Desconocido: Gris

The screenshot displays a web application interface with a grid of character cards and a filter sidebar on the right.

**Character Cards (Top Row):**

- Rick Sánchez** (Vivo): Última ubicación Ciudadela de Ricks
- Morty Smith** (Vivo): Última ubicación Ciudadela de Ricks
- Verano Smith** (Vivo): Última ubicación Tierra (dimensión de reemplazo)

**Character Cards (Bottom Row):**

- Beth Smith** (Vivo): Última ubicación
- Jerry Smith** (Vivo): Última ubicación
- Abadango Cluster Princess** (Vivo): Última ubicación

**Filter Sidebar (Filtros):**

- Estado:** Vivo (selected), Muerto, Desconocido
- Especie:** (dropdown arrow)
- Género:** (dropdown arrow)

**Character Cards (Bottom Row):**

- Izzy** (Vivo): Última ubicación Tierra (dimensión de reemplazo)
- Millones de hormigas** (Muerto): Última ubicación Guarida de Worldender
- Jefe de ratas** (Muerto): Última ubicación Tierra (dimensión de reemplazo)

**Character Cards (Bottom Row):**

- Escrociano** (Vivo): Última ubicación
- Snuffles (bola de** (Vivo): Última ubicación
- Fido** (Vivo): Última ubicación

**Filter Sidebar (Filtros):**

- Humanoide**
- Poopybutthole**
- Mitológico**
- Desconocido**
- Animal** (selected)
- Enfermedad**
- Robot**
- Cronenberg**
- Planeta**
- Género:** (dropdown arrow)



Cuando el sistema no encuentra ningún personaje muestra una alerta

```

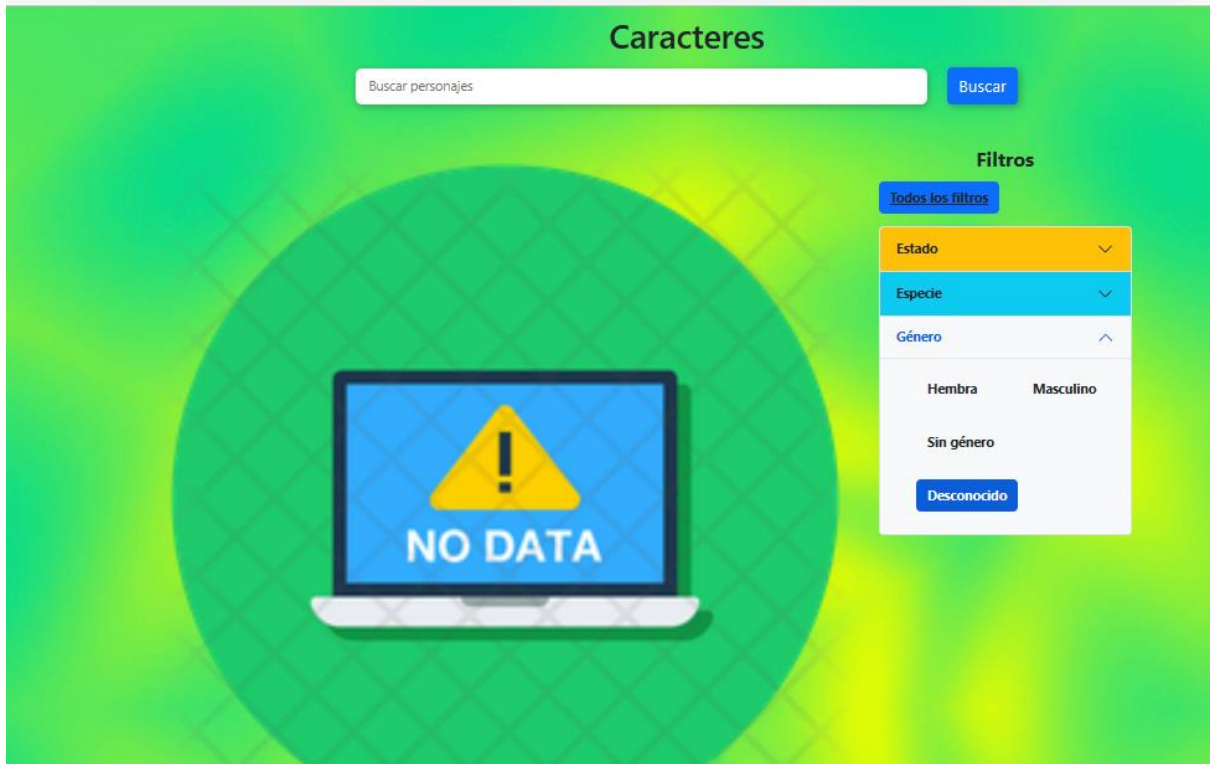
<img className={` ${styles.img} zoom img-fluid img-thumbnail rounded-4`} src={image} alt="" />
<div className={` ${styles.content} text-muted`} >
  <div className="fs-5 fw-bold ">{name}</div>
  <div className="">
    <div className=" fw-normal">Last Location</div>
    <div className="">{location.name}</div>
  </div>
</div>
</div>

{() => {
  if (status === 'Dead') {
    return <div className={` ${styles.badge} position-absolute badge bg-danger`} >{status}</div>
  } else if (status === 'Alive') {
    return <div className={` ${styles.badge} position-absolute badge bg-success`} >{status}</div>
  } else {
    return <div className={` ${styles.badge} position-absolute badge bg-secondary`} >{status}</div>
  }
}}()
</Link>
)
})
} else {display = 
} return <{display}</>

d.propTypes = {
  page: PropTypes.oneOfType([PropTypes.string, PropTypes.number]),
  results: PropTypes.array,
}

export default Card

```



Podemos dar click en cada card y me muestra la información de cada personaje

```
import React, { useState, useEffect } from 'react'
import { useParams } from 'react-router-dom'

import './Card.module.scss';

const CardDetails = () => {
  const { id } = useParams()

  const [fetchData, updateFetchData] = useState([])
  const { name, location, origin, gender, image, status, species, episode } = fetchData

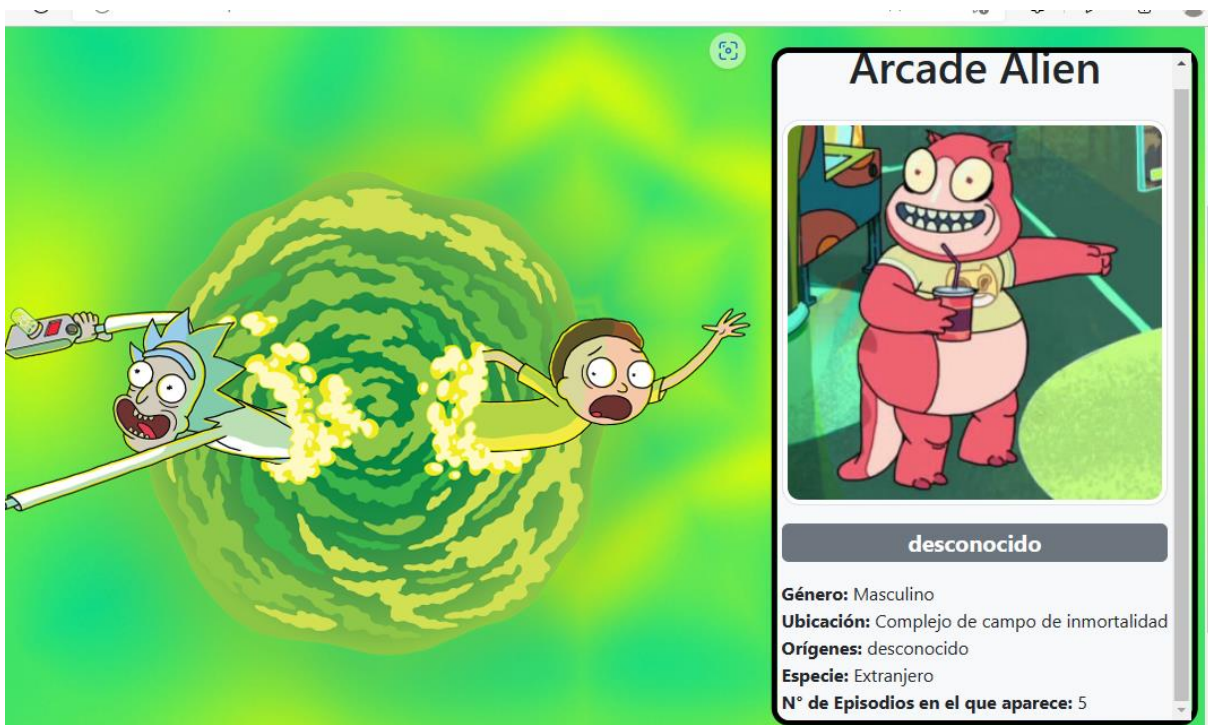
  const api = `https://rickandmortyapi.com/api/character/${id}`

  useEffect(() => {
    ;(async function () {
      const data = await fetch(api).then((res) => res.json())
      updateFetchData(data)
    })()
  }, [api])

  return (
    <div className="container d-flex justify-content-center mb-5">
      <div className="img-thumbnail zoom d-flex flex-column mx-3 gap-3 bg-light rounded-4 border-4"
        style={{
          position: "absolute",
          top: "50%",
          left: "50%",
          transform: "translate(40%, -10%)",
          minWidth: 300,
          maxWidth: 500,
          bgcolor: "background.primary",
          border: "5px solid #000",
          maxHeight: 600,
          overflowY: "auto",
          boxShadow: "24px 24px 0px #000"
        }}
      >
        <img alt="Character image placeholder" data-bbox="338 312 562 438" />
      </div>
    </div>
  )
}
```



```
}  
}  
>  
  
<h1 className="text-center">{name}</h1>  
  
<img className="img-thumbnail zoom rounded-4 border-4" src={image} alt="" />  
{(() => {  
  if (status === 'Dead') {  
    return <div className="badge bg-danger fs-5">{status}</div>  
  } else if (status === 'Alive') {  
    return <div className="badge bg-success fs-5">{status}</div>  
  } else {  
    return <div className="badge bg-secondary fs-5">{status}</div>  
  }  
})()}  
<div className="content" id="content"  
>  
  <div className="">  
    <span className="fw-bold">Gender : </span>  
    {gender}  
  </div>  
  <div className="">  
    <span className="fw-bold">Location: </span>  
    {location?.name}  
  </div>  
  <div className="">  
    <span className="fw-bold">Origin: </span>  
    {origin?.name}  
  </div>  
  <div className="">  
    <span className="fw-bold">Species: </span>  
    {species}  
  </div>  
  <div className="">  
  
    <span className="fw-bold">Nº de Episodios en el que aparece: </span>  
    {episode?.length}  
  </div>  
</div>
```



## 2. El componente Episodes muestra todos los personajes almacenados en la api por episodios.

```
import React, { useEffect, useState } from 'react'
import Card from '../components/Card/Card'
import InputGroup from '../components/Filter/category/InputGroup'

export default function Episodes() {

  const [results, setResults] = React.useState([])
  const [info, setInfo] = useState([])
  const { air_date: airDate, name } = info
  const [id, setID] = useState(1)

  const api = `https://rickandmortyapi.com/api/episode/${id}`

  useEffect(() => {
    ;(async function () {
      const data = await fetch(api).then((res) => res.json())
      setInfo(data)

      const a = await Promise.all(
        data.characters.map((x) => {
          return fetch(x).then((res) => res.json())
        })
      )
      setResults(a)
    })()
  }, [api])

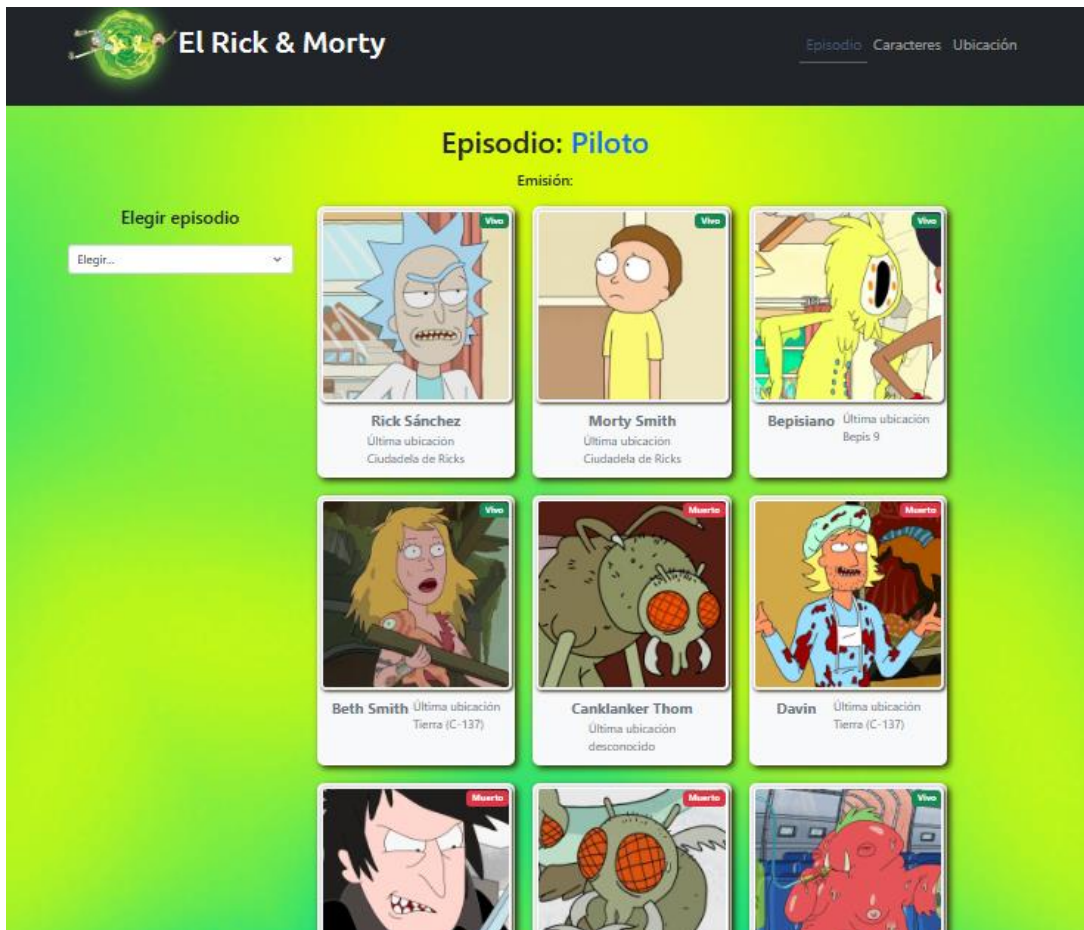
  return (
    <div className="container">
      <div className="row mb-3">
        <h1 className="text-center mb-3">
          Episode : <span className="text-primary">{name === '' ? 'Unknown' : name}</span>
        </h1>
        <h5 className="text-center">Air Date: {airDate === '' ? 'Unknown' : airDate}</h5>
      </div>
      <div className="row">
        <div className="col-lg-3 col-12 mb-4">
          <h4 className="text-center mb-4">Pick Episode</h4>
          <InputGroup name="Episode" changeID={setID} total={51} />
        </div>

```

```

      </div>
      <div className="col-lg-8 col-12 mx-2">
        <div className="row">
          <Card page="/episodes/" results={results} />
        </div>
      </div>
    </div>
  )
}

```



Adicional a esto el sistema nos permite filtrar por episodio y también mirar cada uno a detalle





## Episodio: Algo Ricked de esta manera viene

Emisión:

Elegir episodio

Episodio - 9

Elegir...

Episodio - 1  
Episodio - 2  
Episodio - 3  
Episodio - 4  
Episodio - 5  
Episodio - 6  
Episodio - 7  
Episodio - 8  
Episodio - 9  
Episodio - 10  
Episodio - 11  
Episodio - 12  
Episodio - 13  
Episodio - 14  
Episodio - 15  
Episodio - 16  
Episodio - 17  
Episodio - 18  
Episodio - 19

**Rick Sánchez**Última ubicación  
Ciudadela de Ricks**Morty Smith**Última ubicación  
Ciudadela de Ricks**Verano Smith**Última ubicación  
Tierra (dimensión de reemplazo)**Beth Smith**Última ubicación  
Tierra (dimensión de reemplazo)**Jerry Smith**Última ubicación  
Tierra (dimensión de reemplazo)**Cynthia**Última ubicación  
Tierra (dimensión de reemplazo)**Rey Flippy Nips**Última ubicación  
Plutón**Sr. Goldenfold**Última ubicación  
Tierra (dimensión de reemplazo)**Sr. Needful**Última ubicación  
Tierra (dimensión de reemplazo)

### 3. El componente Location muestra todos los personajes almacenados en la api por localización .

```
import React, { useEffect, useState } from 'react'
import Card from '../components/Card/Card'
import InputGroup from '../components/Filter/category/InputGroup'
export default function Location() {
  const [results, setResults] = React.useState([])
  const [info, setInfo] = useState([])
  const { dimension, type, name } = info
  const [number, setNumber] = useState(1)

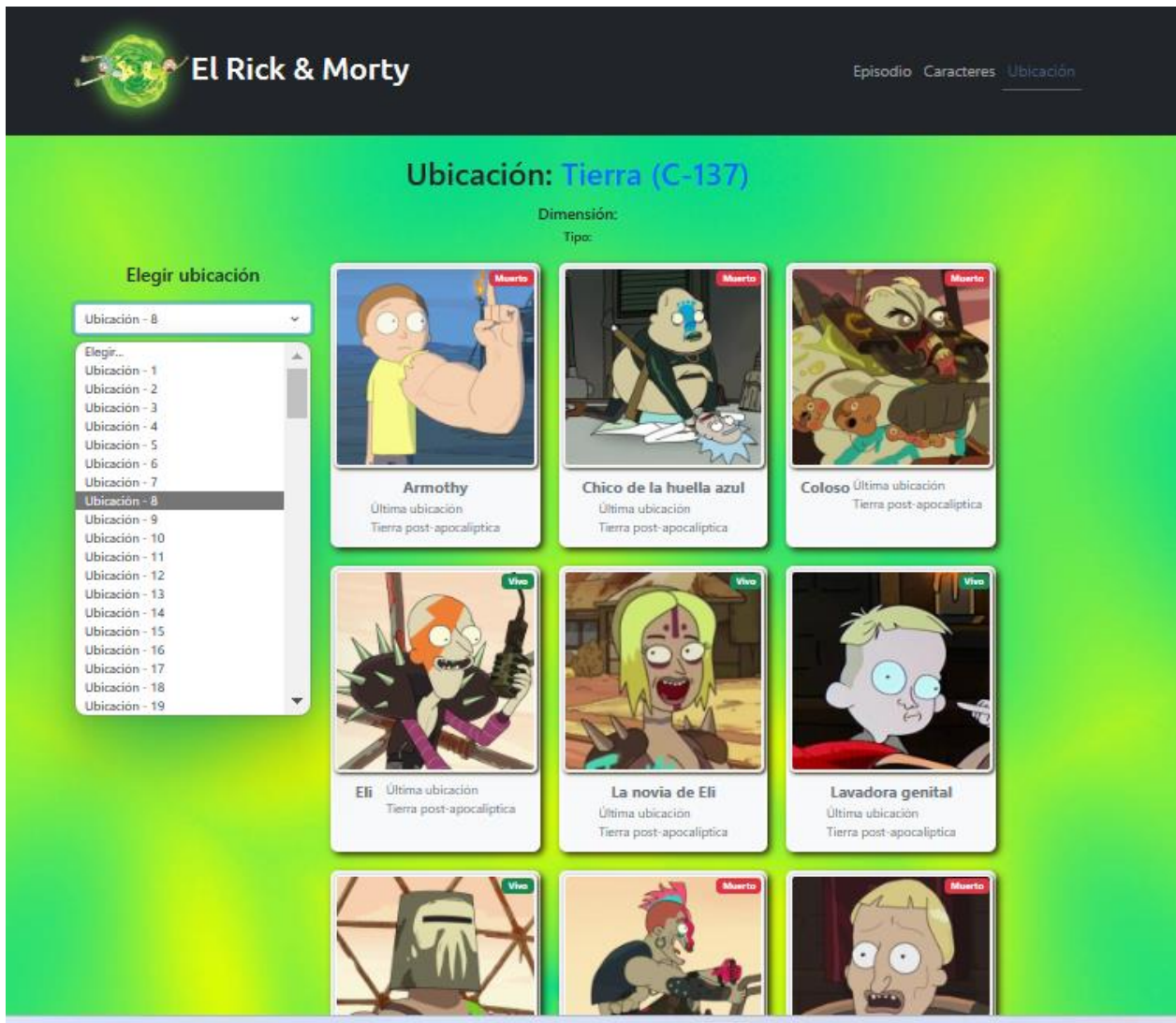
  const api = `https://rickandmortyapi.com/api/location/${number}`

  useEffect(() => {
    ;(async function () {
      const data = await fetch(api).then((res) => res.json())
      setInfo(data)

      const a = await Promise.all(
        data.residents.map((x) => {
          return fetch(x).then((res) => res.json())
        })
      )
      setResults(a)
    })()
  }, [api])

  return (
    <div className="container">
      <div className="row mb-3">
        <h1 className="text-center mb-3">
          Location :<span className="text-primary"> {name === '' ? 'Unknown' : name}</span>
        </h1>
        <h5 className="text-center">Dimension: {dimension === '' ? 'Unknown' : dimension}</h5>
        <h6 className="text-center">Type: {type === '' ? 'Unknown' : type}</h6>
      </div>
      <div className="row">
        <div className="col-lg-3 col-12 mb-4">
          <h4 className="text-center mb-4">Pick Location</h4>
          <InputGroup name="Location" changeID={setNumber} total={126} />
        </div>
      </div>
    </div>
  )
}
```





## REQUERIMIENTOS DE INSTALACIÓN

```
### Packages installation steps

...

npx create-react-app .

npm install bootstrap

npm install @popperjs/core --save

npm install sass

npm install react-paginate --save

npm install react-router-dom
```

```
npm start
...

### Font Awesome CDN
...
<link
  rel="stylesheet"
  href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css"
/>
...

### Google Font Families
...
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@100;200;300;400;500
;600;700;800;900&family=Ubuntu:wght@300;400;500;700&display=swap');

font-family: 'Poppins', sans-serif;
font-family: 'Ubuntu', sans-serif;
```