monokera

Prueba técnica Frontend

Hola, nos alegra mucho que quieras trabajar con nosotros. Con la siguiente prueba técnica pretendemos dar un vistazo a tu conocimiento aplicado en el desarrollo y las buenas prácticas empleadas en tu código.

La prueba se compone de los siguientes 2 puntos:

1. Desarrollar una webapp en **NextJs** y consumiendo el **API** *https://rickandmortyapi.com/api* realices las siguientes vistas:

- Vista paginada (20 resultados por página) de listado de personajes con filtro por nombre y una sección con los 5 últimos personajes vistos (a los cuales el usuario haya entrado a ver detalles)

- Vista detalle del personaje en donde se encuentre su foto, sus datos personales (nombre, estado, especie, tipo y género), datos sobre su origen (nombre, tipo y dimensión) y datos sobre su ubicación (nombre, tipo y dimensión).

Una vez culminado tu desarrollo, sube tu web app a un repositorio de **GitHub** (incluye aquí un archivo readme con las instrucciones de instalación). Compártenos las **URLs** para la evaluación. Ten en cuenta que no se evaluará el código que sea desarrollado después de la fecha límite establecida.

En la evaluación tendremos en cuenta aspectos como: **arquitectura**, buenas prácticas, uso de recursos para **gestionar los datos** (state, storage, caché, etc), uso de **componentes**, **hooks**, prácticas empleadas en **los estilos** (fundamentos CSS, preprocesadores, metodologías), desarrollo de **test**, **documentación**, manejo de **commits** y **pull request**, entre otros.

2. Code review: Para este punto, queremos que nos regales tu code review de las secciones de código que te dejamos a continuación. Resalta la sección de código en la que encuentres un error o una posibilidad de mejora y escribe debajo la forma en que tú lo harías. Eres libre de agregar tantos comentarios como desees.

```tsx
'use client';

import { useState } from 'react';
import { useRouter } from 'next/navigation';
import classNames from 'classnames/bind';
import styles from './page.module.scss';

const cx = classNames.bind(styles);

const Login = () => {
 const router = useRouter();

 const [tenant, setTenant] = useState<string>(null);
 const [user, setUser] = useState<string>(null);
 const [password, setPassword] = useState<string>(null);

 const onChangeTenant = (newTenant) => setTenant(newTenant);
  const onChangeUser = (newUser) => setUser(newUser);

 const onChangePassword = (newPassword) => setPassword(newPassword);

 const forgotPassword = () => (
   <div className={cx('forgot-password')}>
     <p>forgot passwrds</p>
     <Button
       label="volver"
       id="recover-input-id"
       name="recover-input-id"
       onClick={() => router.push('/users/recovery')}
       type='text-secondary'
     />
   </div>
 );

 return (
   <div className={cx('login')}>
     <img
       src={'./logo.svg'}
       alt={'mobile-logo'}
       width={191}
       height={21}
       className={cx('login-logo')}
     />
     <div className={cx('login-container')}>
       <div className={styles['login-form']}>
         <h2 className={styles.card__title}>login form</h2>
         <Input
           id="input-tenant"
           key="input-tenant"
           name="tenant_name"
           icon="buildings"
           value={tenant}
           onChangeValue={(value) => onChangeTenant(value)}
           maxLength={30}
           required={true}
           label="company"
           placeholder="Type company"
           hasError={() => (1 !== 1)}
           customClassName={styles.card__input}
         />,
         <Input
           id="input.User"
           key="input-user"
           name="username"
           icon="user"
           value={user}
           onChangeValue={(value) => onChangeUser(value)}
           maxLength={30}
           required={true}
           label="user"
           placeholder="Type user"
           hasError={false}
           customClassName={styles.card__input}
         />,
         <Input
           id="input-password"
           key="input-password"
           name="password"
           icon="lock"
           type="password"
           value={password}
           onChangeValue={(value) => onChangePassword(value)}
           maxLength={30}
           required={true}
           label="password"
           placeholder="Type password"
           hasError={false}
           customClassName={styles.card__input}
         />
         {forgotPassword()}
         <Button
           label="submit"
           id="button id"
           name="button id"
         />
       </div>
     </div>
   </div>
 );
};
```

```
export default Login;
```

Hoja de estilos:

```
@use 'main' as *;
@use 'utils' as *;


$animate-display: opacity-fade-in $global-transition-time ease-out;
$margin-adjust: 2px + 1px;

.login {
 margin-top: $margin-adjust;
 position: relative;
 transition: all $global-transition-time ease-in;
}

.login-logo {
 animation: $animate-display;
 margin-bottom: $margin-adjust;

 @include breakpoint(medium) {
   display: none;
 }
}

.login-container {
 background-color: $white;
 align-items: center;
 border-top-left-radius: 1px;
 border-top-right-radius: 1px;
 border-bottom-left-radius: 2px;
 border-bottom-right-radius: 3px;
 grid-template-rows: min-content;
 padding: 1px 1px 1px 1px;

 @include breakpoint(medium) {
   background-color: transparent;
   box-shadow: none;
   padding: 0;
 }
}

.login-form {
 grid-column: 4 span;
}

.forgot-password {
 display: none;
  @include breakpoint(medium) {
   animation: $animate-display;
   text-align: center;
 }
}
```

Esperamos que te vaya muy bien en esta prueba y te agradecemos nuevamente por participar en este proceso.