Sistema simple con Autenticación Multi Factorial

Presenta :
César Augusto Ortega Buendía

mazo de 2025

1. Justificación

Se crea este sistema con el fin de ofrecer una demostración de autenticación y uso de prácticas seguras al diseñar software. En la nuestra era, la seguridad y la protección de los sistemas informáticos son fundamentales para mantener la integridad de la información y prevenir ataques cibernéticos. Sin embargo, muchos sistemas actuales aún utilizan métodos de autenticación débiles y vulnerables, lo que pone en riesgo la seguridad de los usuarios y la integridad de los datos.

Un sistema de autenticación multi-factorial (MFA) es la solución propuesta y realizada para abordar este desafío.

2. Introducción

El objetivo de este proyecto es diseñar y desarrollar un sistema de autenticación multi-factorial utilizando Django como framework de backend y React como framework de frontend.

El sistema permitirá a los usuarios registrarse y iniciar sesión utilizando una combinación de nombre de usuario y contraseña, así como un segundo factor de autenticación, como un código de verificación desplegado en un código QR posterior a la Autenticación básica que redigirá a Google Authenticator para poder obtener una contraseña para una ocasión (OTP - One Time Password).

3. Desarrollo

El backend hecho en django consta de una integración de varias librerías de python para hacer posible el proyecto

- Django: La plataforma de elección para el sistema
- django-cors-headers: Esta dependencia permite controlar el acceso CORS (Cross-Origin Resource Sharing) en la aplicación Django, lo que permite que los navegadores web realicen solicitudes a la API desde orígenes diferentes.
- djangorestframework: Es una biblioteca que permite desarrollar APIs REST en Django de manera fácil y eficiente.
- pytz: pytz es una biblioteca que proporciona información sobre zonas horarias
- qrcode: Esta dependencia permite generar códigos QR
- pyotp: Es una biblioteca que permite generar y verificar códigos de autenticación de un solo uso (OTP)
- django-ratelimit: Esta dependencia permite limitar la velocidad a la que se pueden realizar solicitudes a la API
- psycopg2: psycopg2 es una biblioteca que permite conectar a bases de datos PostgreSQL

3.1. Desarrollo de Authenticator - Modelos

- La app Authenticator fue creada dentro de un **Entorno Virtual** en python para facilitar el uso de las herramientas previamente mencionadas
- Se modelaron los objetos
  - **customUser**: Con el find de manejar una vez el usuario ingresado el status de si el MFA está activo o no
  - **LoginAttempt**: Con la finalidad de contar los intentos por usuario para acceder a la plataforma
  - **User**: Con la responsabilidad de guardar y actualizar la información básica del usuario

3.2. Desarrollo de Authenticator - Vistas

- La parte de las rutas refiere a los siguientes urls:
  - **loginReact** : Para enviar una petición al sistema para loggearse dentro de él
  - **registerReact** : Registra el usuario para almacenarlo en el sistema
  - **verify_mfaReact** : Para verificar si la OTP es correcta
  - **update_mfa_enabledReact** : Para actualizar si el usuario tiene o no la opción de usar el código OTP al iniciar sesión

4. Desarrollo de Frontend - Reactapp:

En la actualidad, la creación de aplicaciones web requiere de tecnologías robustas y eficientes que permitan desarrollar interfaces de usuario atractivas y funcionales. React se basa en la idea de componentes reutilizables, lo que permite crear interfaces de usuario modulares y fáciles de mantener.

Para esto se crean las vistas

- La parte de las rutas refiere a los siguientes urls:
  - **App** : La página principal
  - **Navbar** : Representa la barra de navegación en su estado normal
  - **NavbarIn** : Versión alterna de navbar para mostrar el botón de salir una vez que se ingresa
  - **Register** : Viasta respobsable de crear usuarios y mandarlos a la DB
  - **Login** : Vista que valida el Login, con OTP o no dependiendo de si el usuario lo dió de alta
  - **ProtectedRoute** : La página principal del sistema con un check que permite activar/desactivar el MFA

5. Despliege

5.1. Local

Ambas apps, están disponibles en los repositorios de github:

- **https://github.com/cesarob98/reactapp**
- **https://github.com/cesarob98/Creze**
- **El código de la base de datos se encuentra dentro del repositorio Creze**

5.2. AWS

Para el despliegue de las aplicaciones se utlizó AWS:

- **AWS Elastic Beanstalk** para el backend disponible en: https://django-env.eba-qkkpspsz.us-west-2.elasticbeanstalk.com
- **AWS Amplify** para el frontend, disponible en: https://master.d1s89094099o2d.amplifyapp.com
- **AWS RDS** para la base de datos disponible en: creze.c9ussqiak44u.us-east-2.rds.amazonaws.com

6. Tour Visual

Figura 1: Landing page

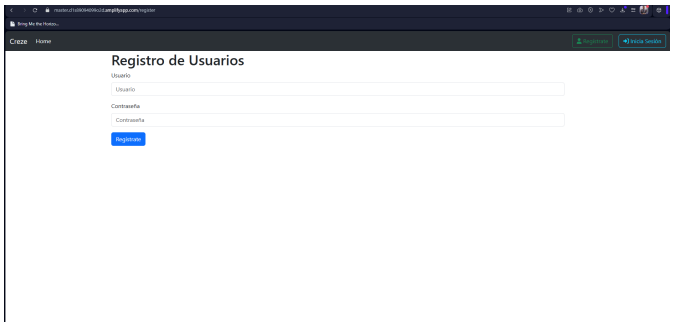

Figura 2: Registro de usuarios

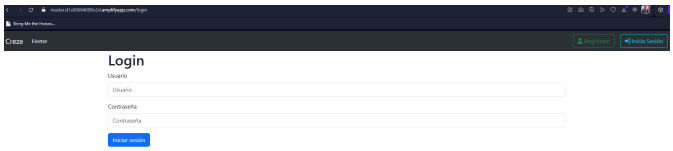

Figura 3: Login

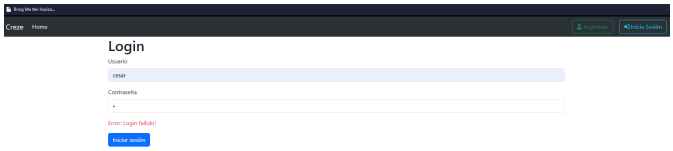

Figura 4: Login Erroneo

Figura 5: Antes de activar el MFA

Figura 6: Después de activar el MFA

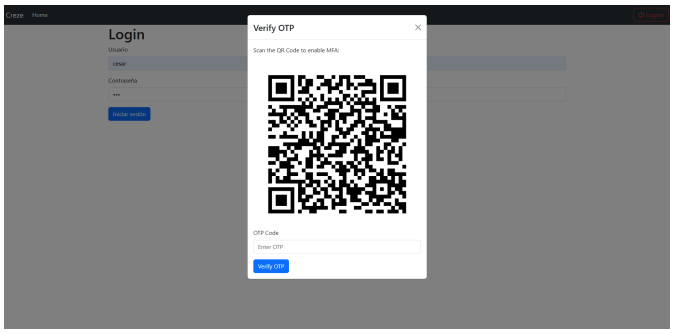

Figura 7: Una vez exitoso el login, se requiere ingresar la OTP

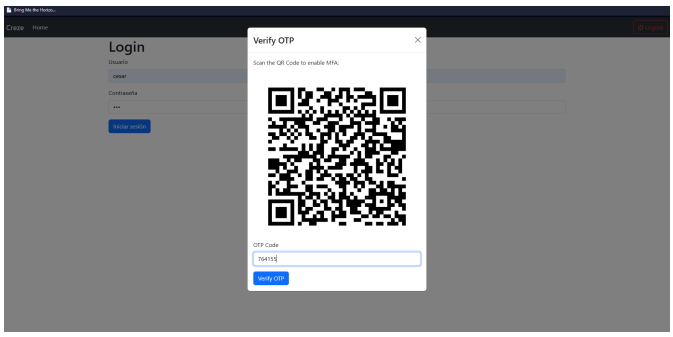

Figura 8: Se ingresa la OTP después de abrir Google Authentticator

Figura 9: Usuario accedió exitosamente después de ingresar la OTP

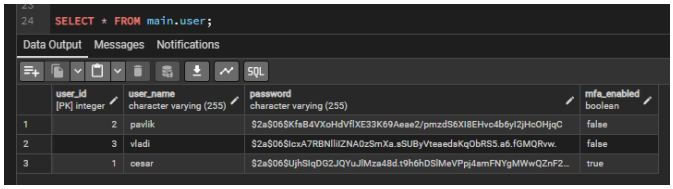

Figura 10: Vista de los usuarios disponibles en la BD