

# SecuraBank

## SGT

David Aucancela



# Temario



Introducción general, sistemas transaccionales, seguridad web, OWASP



Sistema, base de datos, funcionabilidades, requisitos, tecnologías, estilos y arquitectura, product-backlog



Despliegue, railway, pruebas, KaLinux, Owap-Zap, Nikto

# Introducción

En la actualidad, la seguridad en los sistemas de transacciones web es un aspecto crítico para las instituciones financieras y bancarias.

Con el crecimiento del comercio electrónico y la digitalización de los servicios financieros, los ciberdelincuentes han incrementado sus ataques contra plataformas de pago, banca en línea y aplicaciones transaccionales.



+2200 ataques  
web por día

---

Según Security Magazine

Un ataque cada 39 segundos

\$2 000 000

---

Perdidas en el sector  
financiero

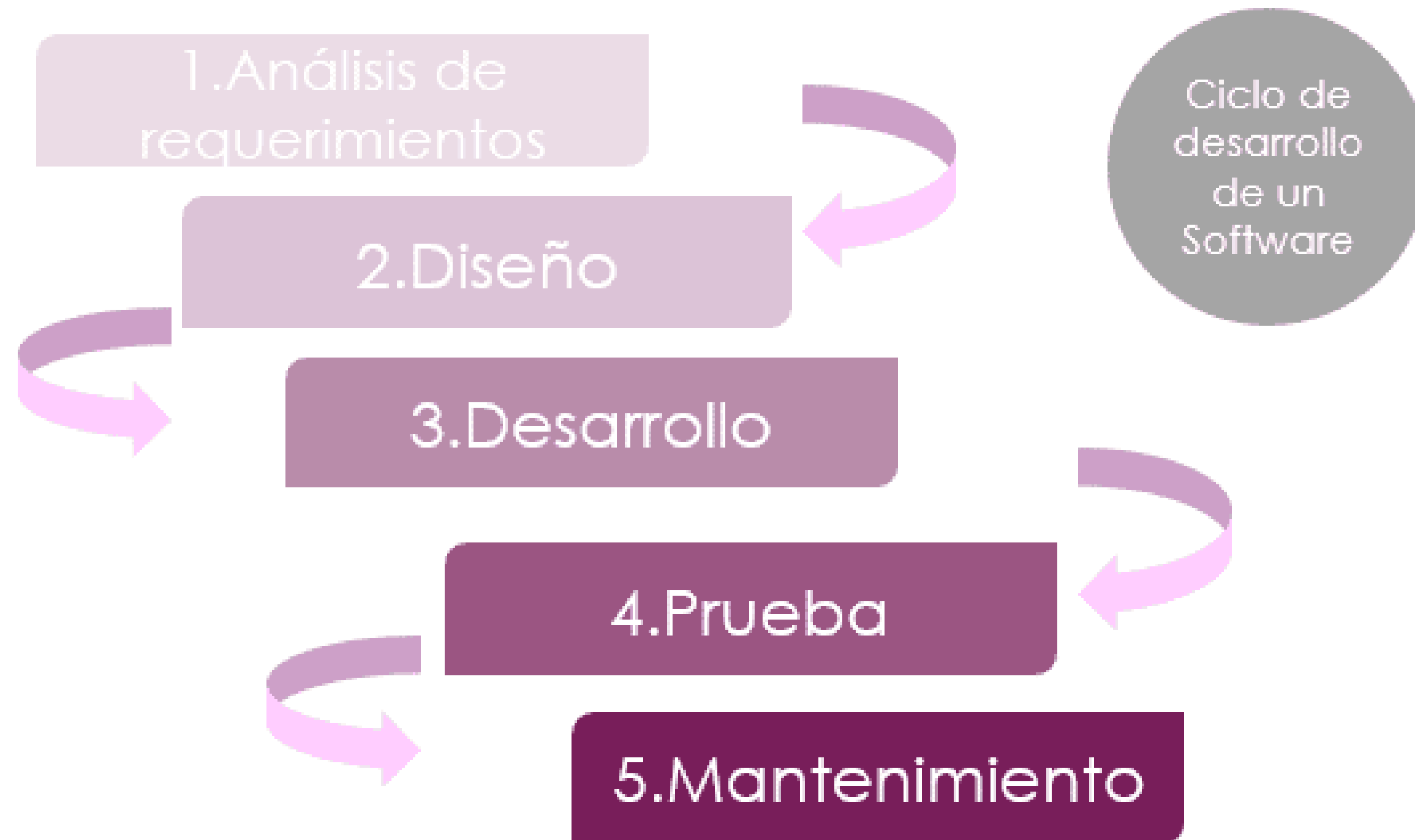
2020

\$150 000 000

---

Perdidas en el sector  
financiero

2024



Las fases de un producto de software permiten la creación de un sistema funcional.

Se brinda un enfoque de mayor seguridad al implementar seguridad en todas las capas del proyecto.

# Seguridad web

# Seguridad web

Protege contra:

Robos de software, hardware o datos

Interrupción de servicios

Infracciones de seguridad

Pérdida de datos

¿Cómo crear  
un sistema  
transaccional  
seguro?

## Sistema seguro

---

Implementar medidas de seguridad robustas para proteger los sistemas transaccionales contra amenazas emergentes.



*awasp!*

# TOP 10 OWASP

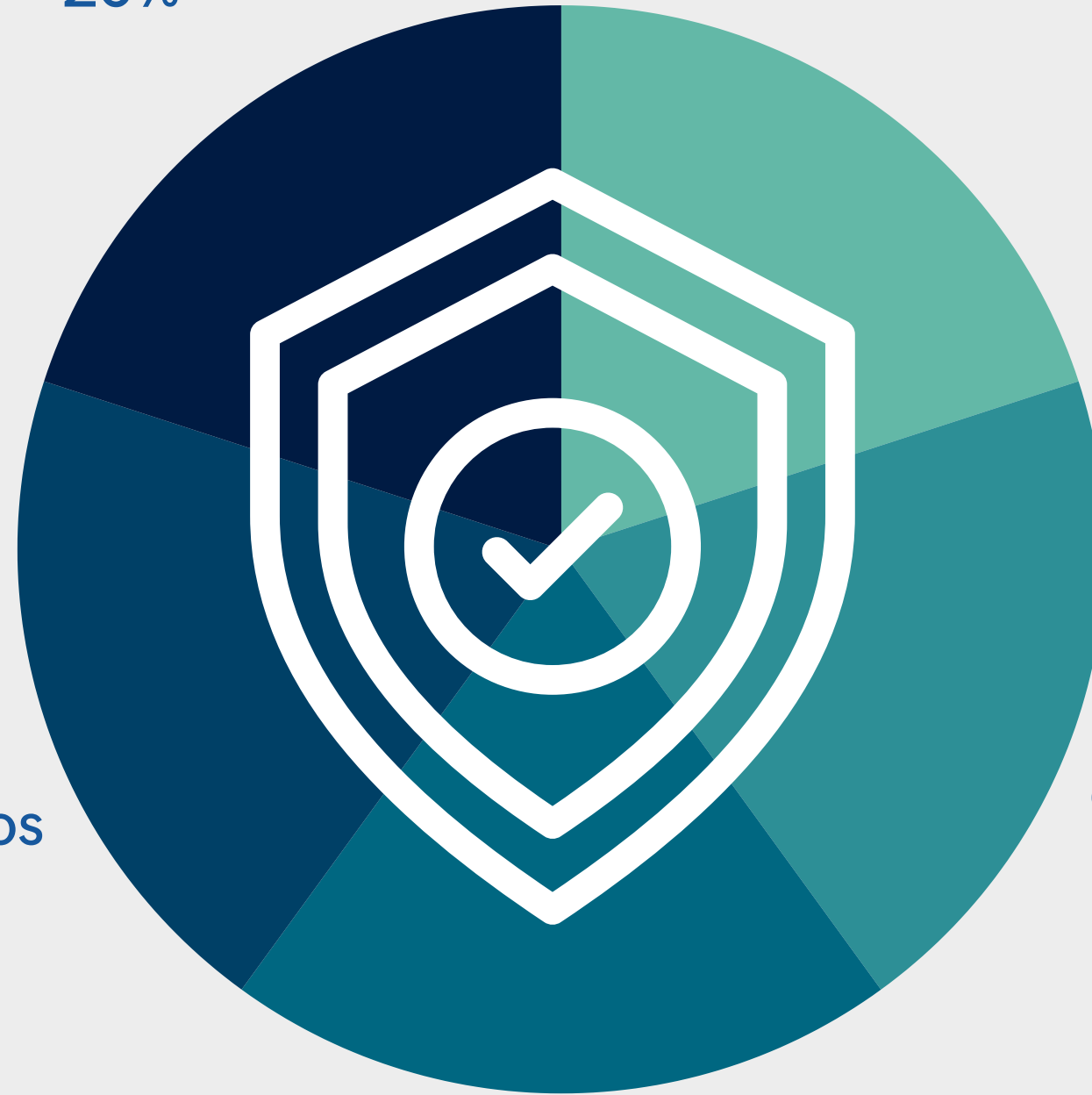
Seguridad en APIs  
20%

Protección contra ataques de autenticación  
20%

Configuración de la base de datos  
20%

Gestión segura de sesiones  
20%

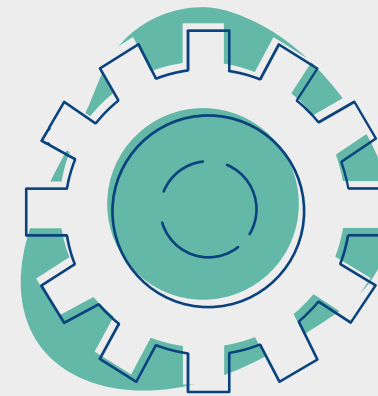
Protección contra ataques de deserialización insegura  
20%





# Tecnologías

Al realizar la comparativa con otras opciones de tecnologías como Angular o Vue y bases de datos como MariaDB se han escogido las siguientes por las características y funcionalidades deseadas a obtener:



Django - Backend

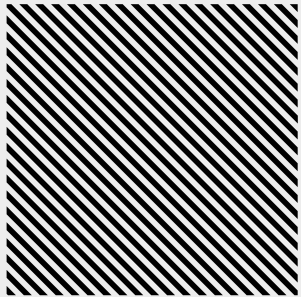


React - Frontend

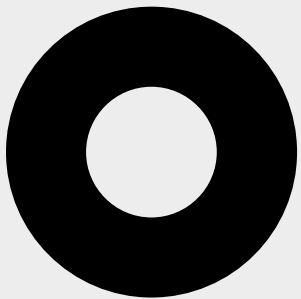


PostgreSQL

# Requerimientos

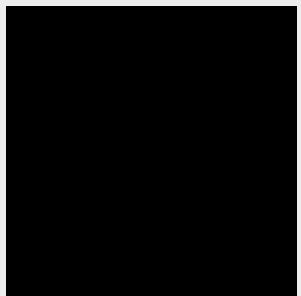


REQ-1: El sistema debe permitir a los usuarios autenticarse mediante nombre de usuario y contraseña.



REQ-3: Si el segundo factor de autenticación es inválido, el sistema mostrará un mensaje de error y permitirá reintentos limitados.

REQ-6: Al ingresar configuraciones incorrectas, el sistema debe mostrar advertencias y sugerencias de corrección.



REQ-7: El sistema permitirá al administrador crear, editar y eliminar usuarios

REQ-13: El sistema permitirá a los usuarios verificar y gestionar los dispositivos conectados a su cuenta.

REQ-14: Los usuarios podrán cerrar sesiones activas en dispositivos no autorizados

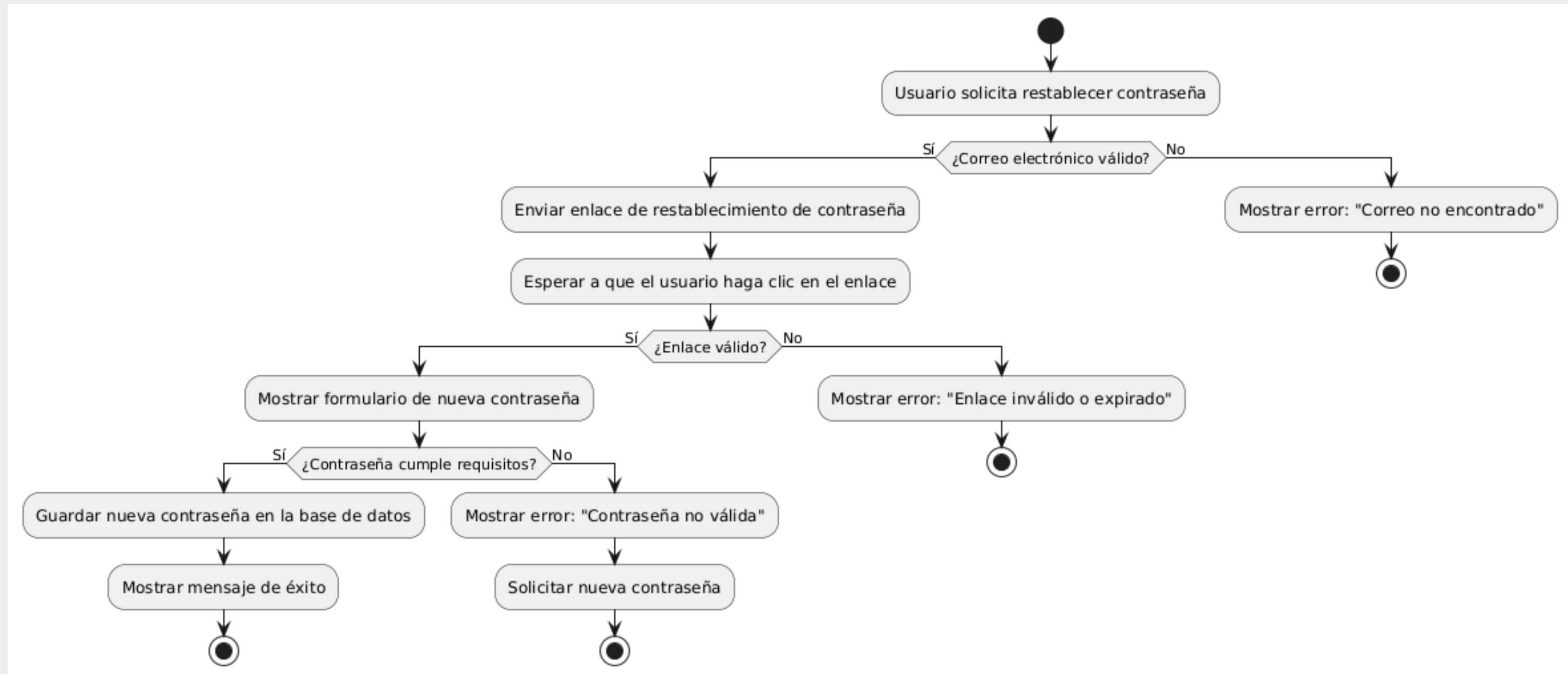
REQ-19: El sistema permitirá simular transacciones bancarias de manera segura y controlada.

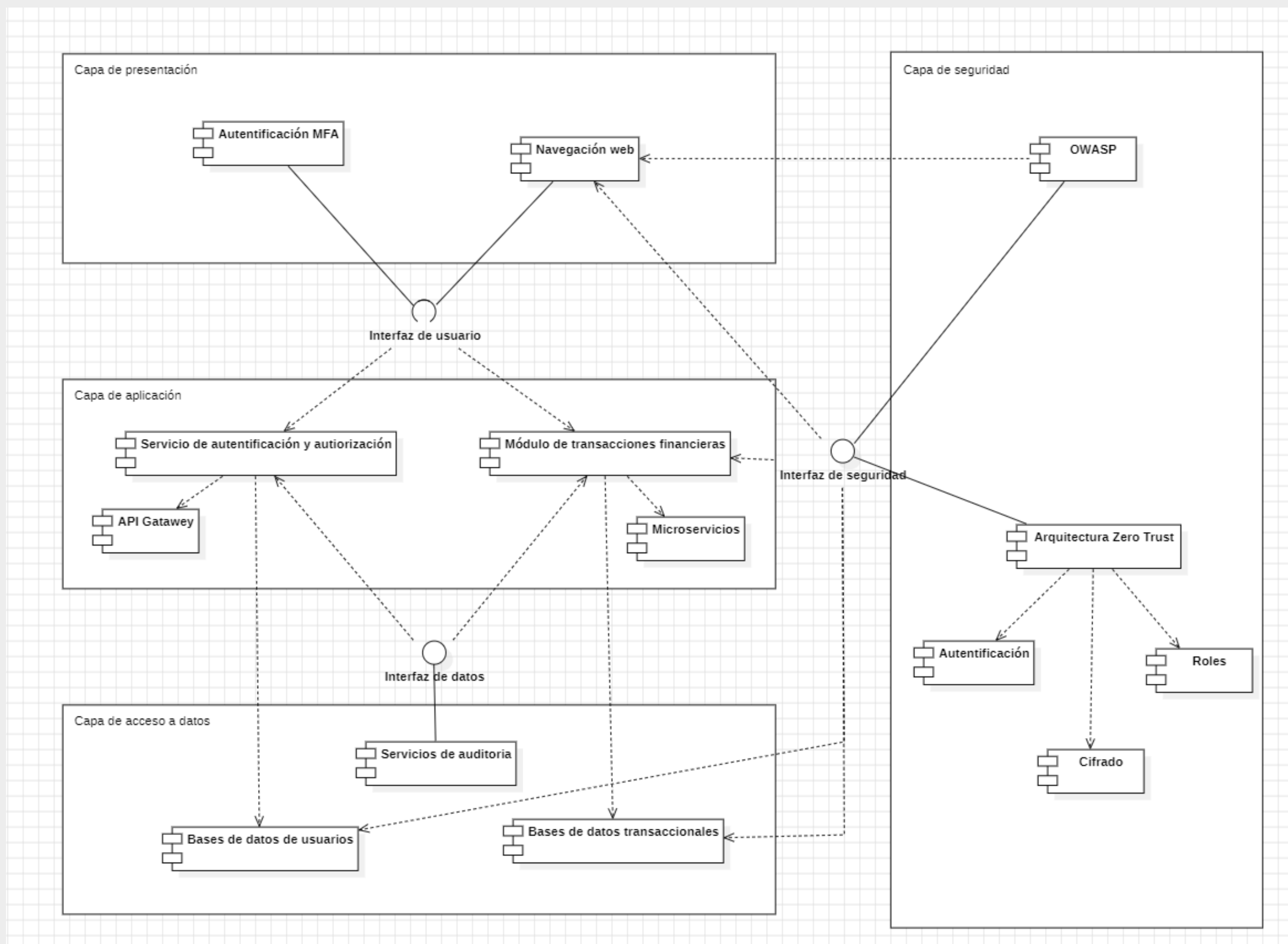
REQ-20: El sistema debe validar todos los datos ingresados en la simulación para evitar errores.

REQ-35: Los administradores podrán habilitar o deshabilitar la autenticación multifactorial (MFA)

# Diagrama de actividades

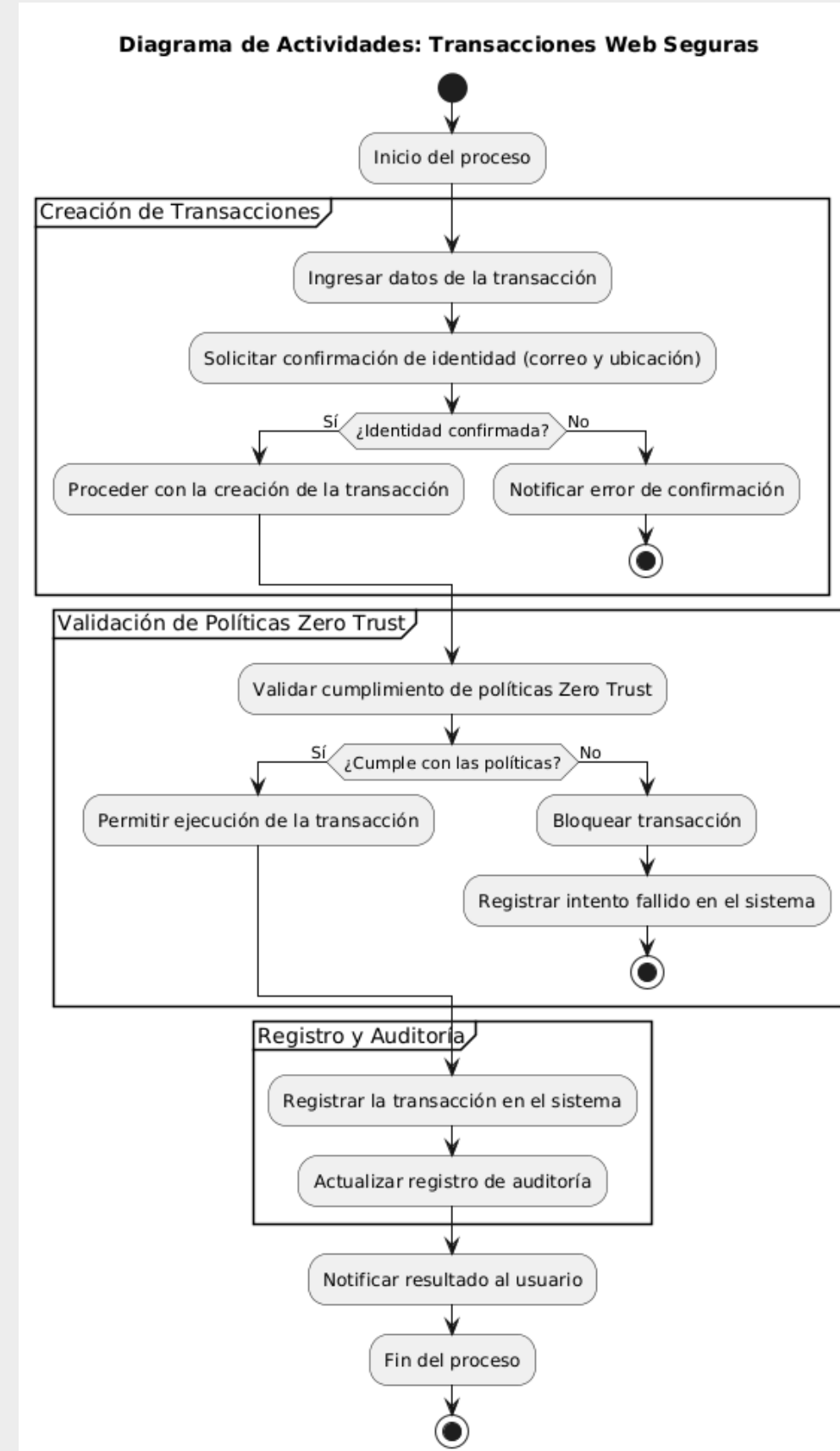
## Restablecer contraseña





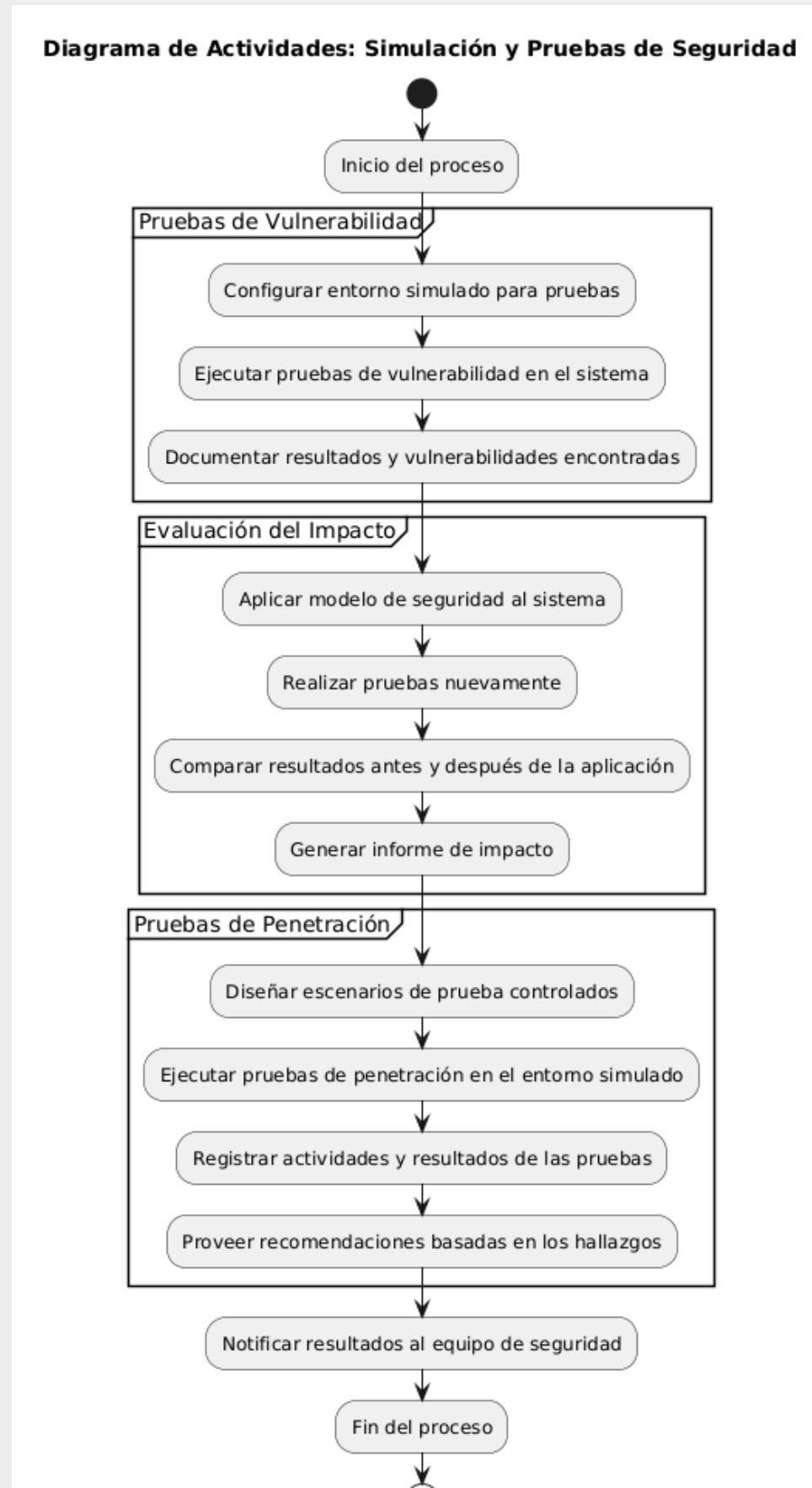
# Diagrama de actividades

## Transacciones



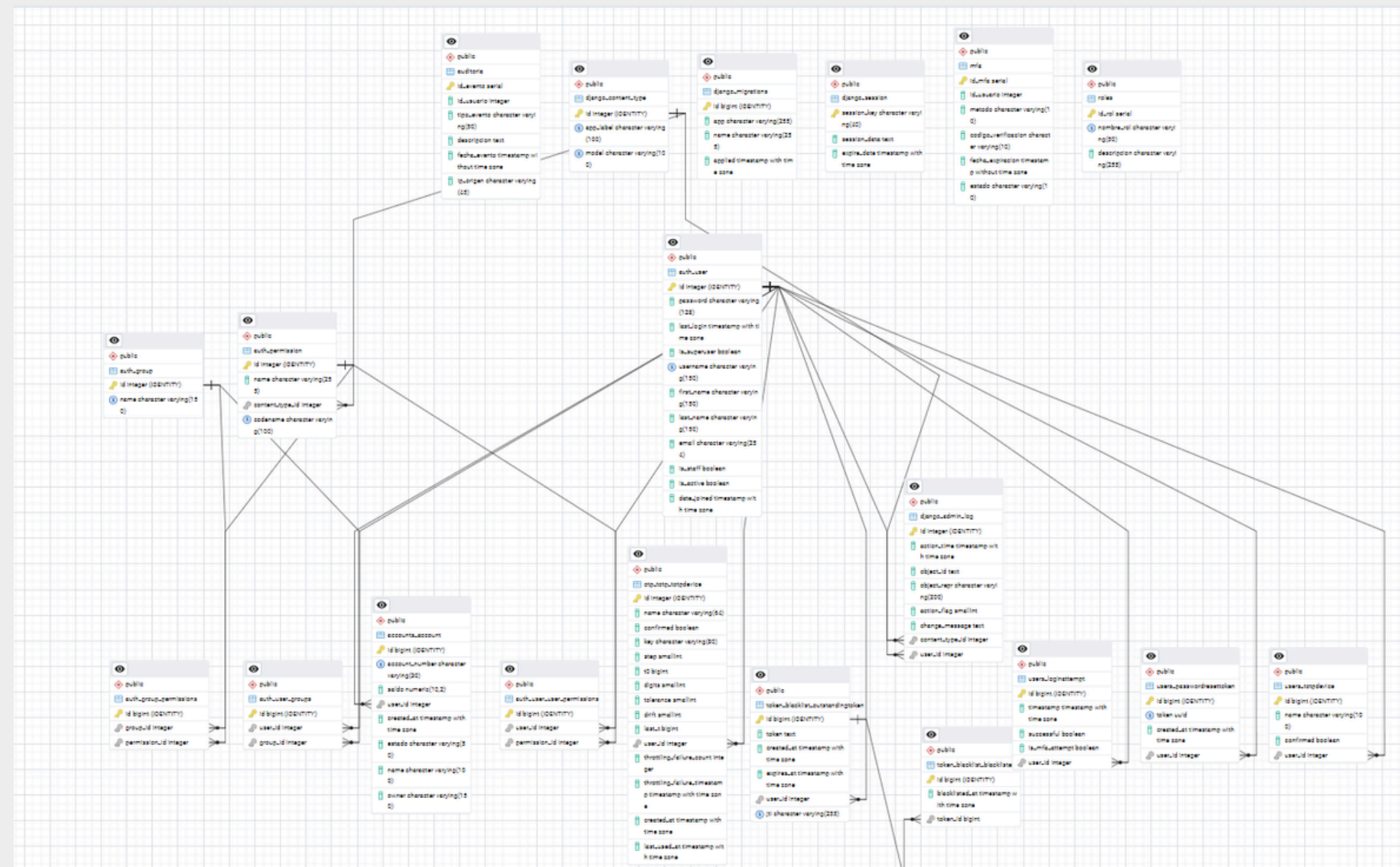
# Diagrama de actividades

## Pruebas de seguridad

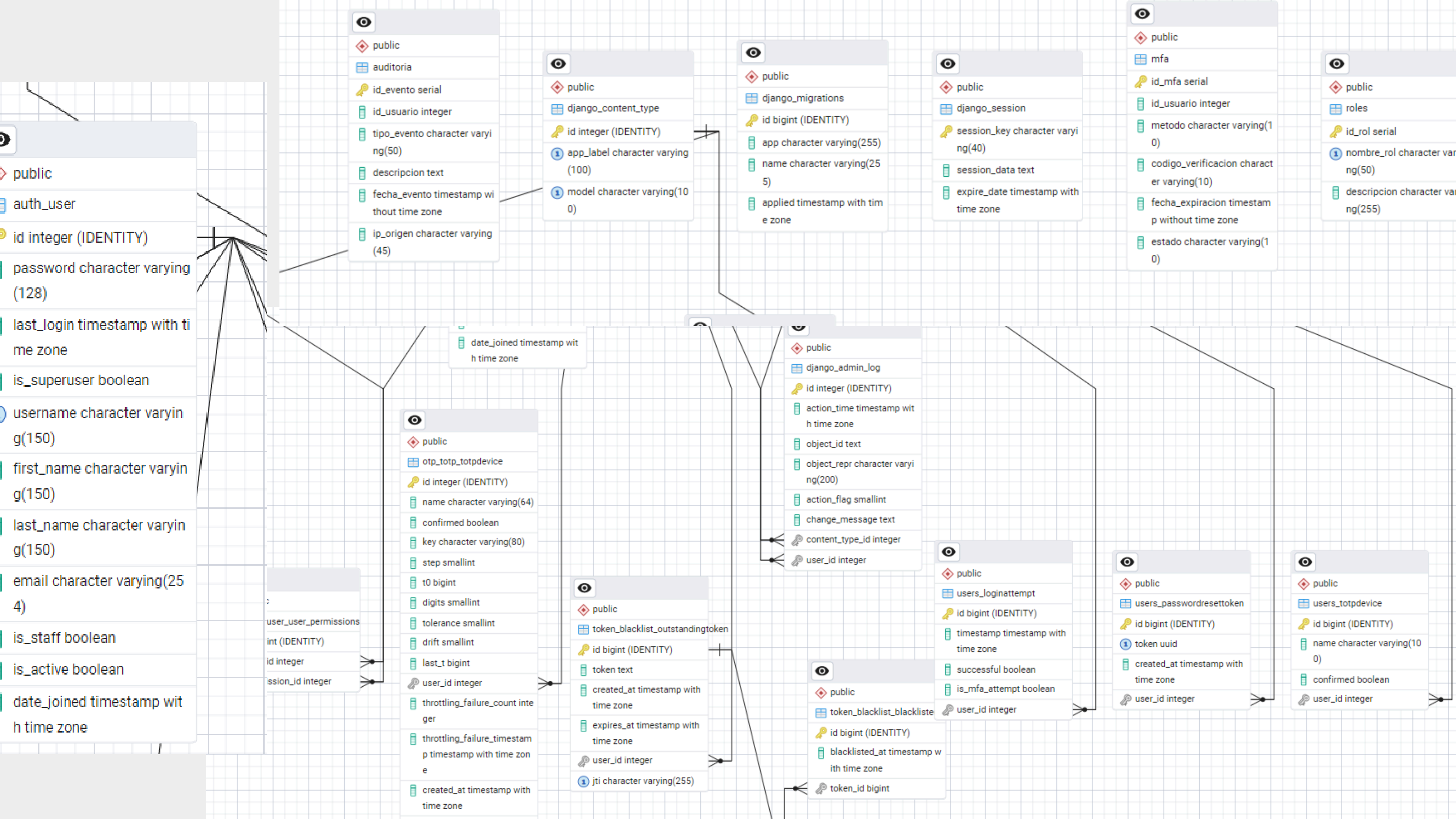


# Diseño de la base de datos

Al utilizar el framework de django para el backend se puede utilizar herramientas que brinda este framework para la creación de las tablas y relaciones que se necesite especificar.









# Protección contra ataques de autenticación

---

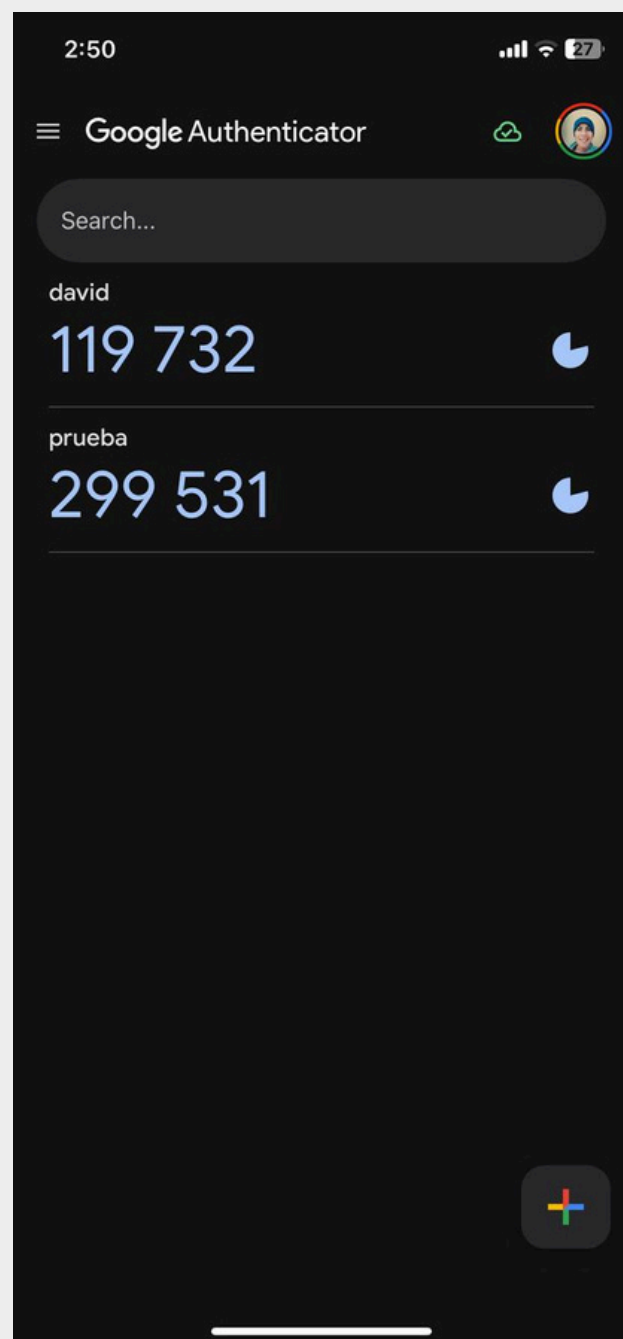
Implementación de límites de intentos fallidos y uso de CAPTCHA.

```
# Si el usuario existe, verificar intentos fallidos recientes
if user:
    block_duration = timedelta(minutes=5)
    recent_attempts = LoginAttempt.objects.filter(
        user=user,
        timestamp__gte=datetime.now() - block_duration,
        successful=False
    )
    failed_attempts = recent_attempts.count()

    if failed_attempts >= 3:
        return Response(
            {"detail": "Demasiados intentos fallidos. Inténtalo de nuevo más tarde."},
            status=status.HTTP_429_TOO_MANY_REQUESTS
        )
```



=253WLTTULJRADD64JECW6I6WNKNARIN&algorithm=S



```
# ===== MFA (TOTP) =====

@api_view(['GET'])
@permission_classes([IsAuthenticated])
def generate_mfa_qr(request):
    """
    Genera el código QR para configurar TOTP en la app de Autenticación (Google Auth).
    """
    user = request.user
    device, created = TOTPDevice.objects.get_or_create(user=user, name='default')

    # Si el dispositivo aún no está confirmado, generamos el QR
    if not device.confirmed:
        qr_url = device.config_url
        img = qrcode.make(qr_url, image_factory=qrcode.image.svg.SvgImage)
        buffer = BytesIO()
        img.save(buffer)
        svg = buffer.getvalue().decode()
        return Response({
            'qr_code': svg,
            'secret': device.bin_key
        })
    return Response({'detail': 'MFA ya está configurado.'}, status=status.HTTP_400_BAD_REQUEST)
```

# Gestión segura de sesiones

---

Uso de tokens de sesión seguros y configuración de tiempos de expiración adecuados.

```
# ===== LOGIN (JWT) =====  
class CustomTokenObtainPairView(TokenObtainPairView):  
    """  
    Vista personalizada para forzar MFA antes de entregar tokens.  
    Implementa control de intentos de login y bloqueo tras 3 intentos fallidos en 5 minutos.  
    """  
    permission_classes = (AllowAny,)  
    serializer_class = CustomTokenObtainPairSerializer  
  
    def post(self, request, *args, **kwargs):
```

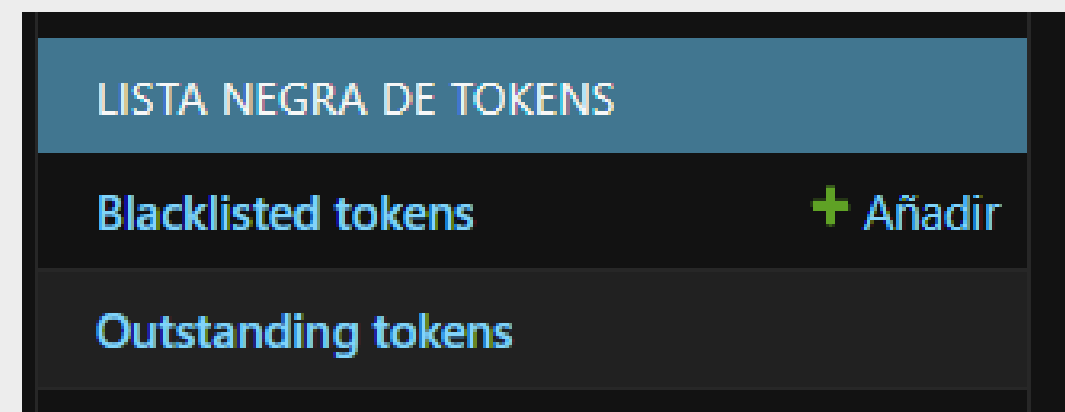
Se controla que el token se obtenga para completar el proceso de LOGIN, JWT



# Protección contra ataques de deserialización insegura

---

Validación estricta de objetos serializados y uso de listas negras.



```
def perform_create(self, serializer):
    # Obtener datos de la transacción
    from_account_id = self.request.data.get('from_account')
    to_account_id = self.request.data.get('to_account')
    monto = float(self.request.data.get('monto'))

    # 1) Verificar que la cuenta de origen existe y pertenece al usuario
    from_account = get_object_or_404(Account, id=from_account_id, user=self.request.user)

    # 2) Verificar que la cuenta de destino existe
    to_account = get_object_or_404(Account, id=to_account_id)

    # 3) Verificar saldo suficiente
    if from_account.saldo < monto:
        return Response(
            {"error": "Saldo insuficiente en la cuenta de origen"},
            status=status.HTTP_400_BAD_REQUEST
        )

    # 4) Actualizar los saldos
    from_account.saldo -= monto
    to_account.saldo += monto

    # Guardar los cambios en las cuentas
    from_account.save()
    to_account.save()
```

# Implementación de controles de acceso

---

Tokens de autenticación y límites de peticiones.

```
class CustomTokenObtainPairSerializer(TokenObtainPairSerializer):  
    """  
    Serializer personalizado para login con JWT.  
    Retorna username y email como parte del token.  
    """  
  
    @classmethod  
    def get_token(cls, user):  
        token = super().get_token(user)  
  
        # Añadir campos personalizados  
        token['username'] = user.username  
        token['email'] = user.email  
  
        return token
```

```
@api_view(['POST'])
@permission_classes([AllowAny]) # Permitido para usuarios sin tokens (MFA pendiente)
def confirm_mfa(request):
    """
    Endpoint que NO requiere estar autenticado,
    pues el usuario aún no tiene tokens (MFA pendiente).
    Se espera { username, token } en el body.
    Implementa control de intentos de MFA y bloqueo tras 3 intentos fallidos en 5 minutos.
    """

    username = request.data.get('username')
    totp_code = request.data.get('token')

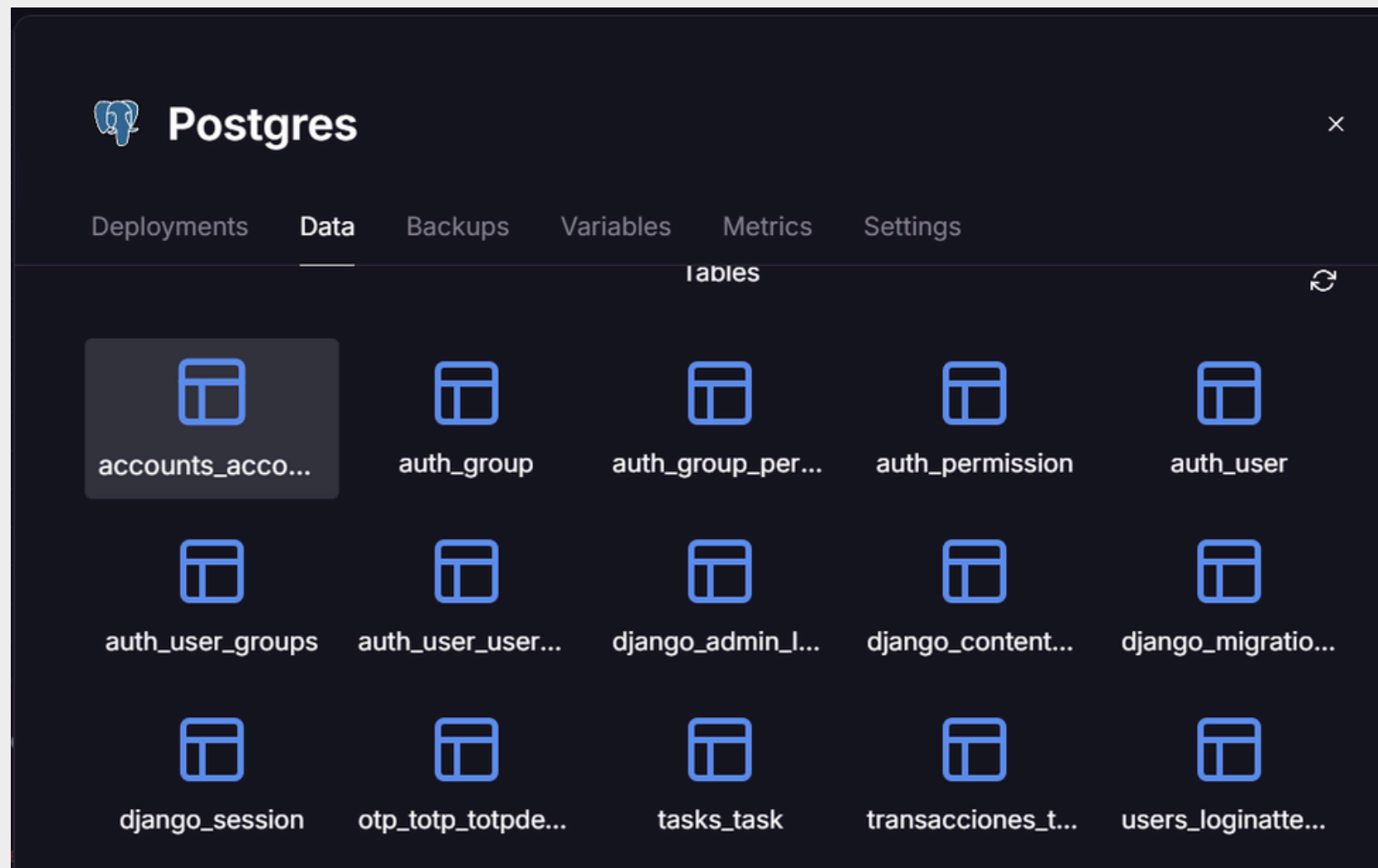
    if not username or not totp_code:
        return Response({'detail': 'Faltan campos.'}, status=status.HTTP_400_BAD_REQUEST)

    # Buscar el usuario
    try:
        user = User.objects.get(username=username)
    except User.DoesNotExist:
        return Response({'detail': 'Usuario no encontrado.'}, status=status.HTTP_400_BAD_REQUEST)

    # Verificar si el usuario está bloqueado por intentos fallidos de MFA
    block_duration = timedelta(minutes=5)
    recent_mfa_attempts = LoginAttempt.objects.filter(
        user=user,
        timestamp__gte=timezone.now() - block_duration,
        successful=False,
        is_mfa_attempt=True
    )
    failed_mfa_attempts = recent_mfa_attempts.count()
```

# Configuración de la base de datos

Cambio de base de datos, seguridad en la base de datos



```
#Database
#postgresql -SEGURA
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': config('DB_NAME'),
        'USER': config('DB_USER'),
        'PASSWORD': config('DB_PASSWORD'),
        'HOST': config('DB_HOST'),
        'PORT': config('DB_PORT'),
    }
}

#postgresql - nube
DATABASES = {
    'default': dj_database_url.config(default="postgresql
    # 'default': dj_database_url.parse(default=os.ge
}
```

```
src > .env
1 DB_NAME=dbSGT
2 DB_USER=postgres
3 DB_PASSWORD=admin
4 DB_HOST=localhost
5 DB_PORT=5434
6
```



# Protección contra ataques de autenticación


---

Implementación de límites de intentos fallidos y uso de CAPTCHA.

```
// Temporizador para desbloquear MFA después del bloqueo
useEffect(() => {
  let blockTimerInterval;
  if (isBlocked) {
    blockTimerInterval = setInterval(() => {
      setBlockTimer(prev => {
        if (prev <= 1) {
          clearInterval(blockTimerInterval);
          setIsBlocked(false);
          setAttempts(0);
          return 300;
        }
        return prev - 1;
      });
    }, 1000);
  }
  return () => clearInterval(blockTimerInterval);
}, [isBlocked]);
```

**pruebaQR (prueba)**

**Identity**

**User:**   **prueba**  
The user that this device belongs to.

**Name:**   
The human-readable name of this device.

☒ **Confirmed**  
Is this device ready for use?

```
jwt-decode : ^4.0.0 ,
"qrcode.react": "^4.2.0",
"react": "^18.0.0"
```

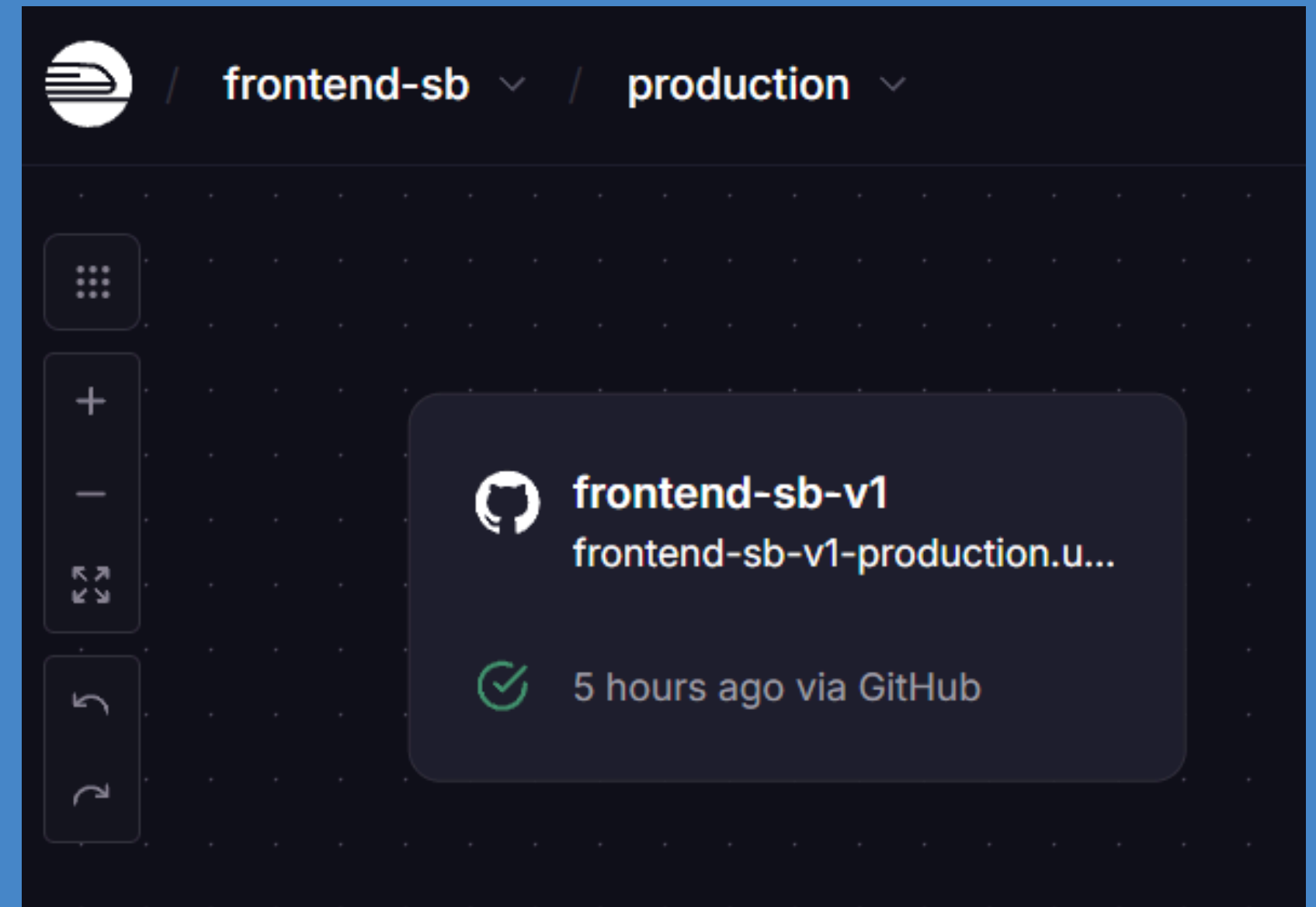
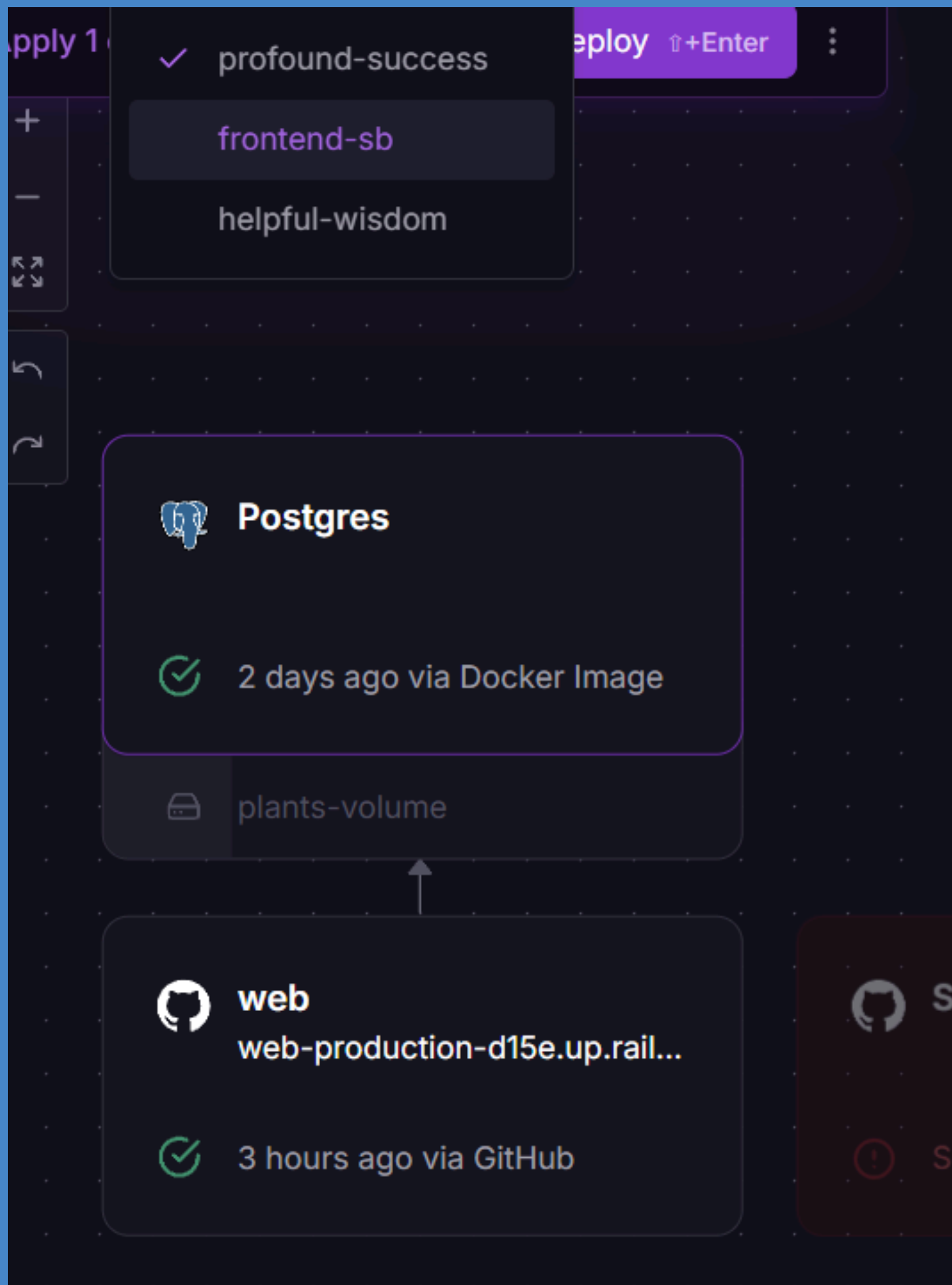
# Despliegue

Para continuar con la siguiente fase del ciclo de vida de software se realiza el despliegue en un servidor.

Railway es una plataforma de infraestructura donde puedes aprovisionar infraestructura, desarrollar con esa infraestructura localmente y luego implementarla en la nube.



**Railway**



<https://web-production-d15e.up.railway.app/api/users/login/>

<http://localhost:8000/api/users/login/>

# Pruebas

Al tener el sistema desplegado en un servidor (web) se puede realizar distintas pruebas para medir el nivel de seguridad del aplicativo, buscando implementar los objetivos OWASP deseados.



Owasp Zap

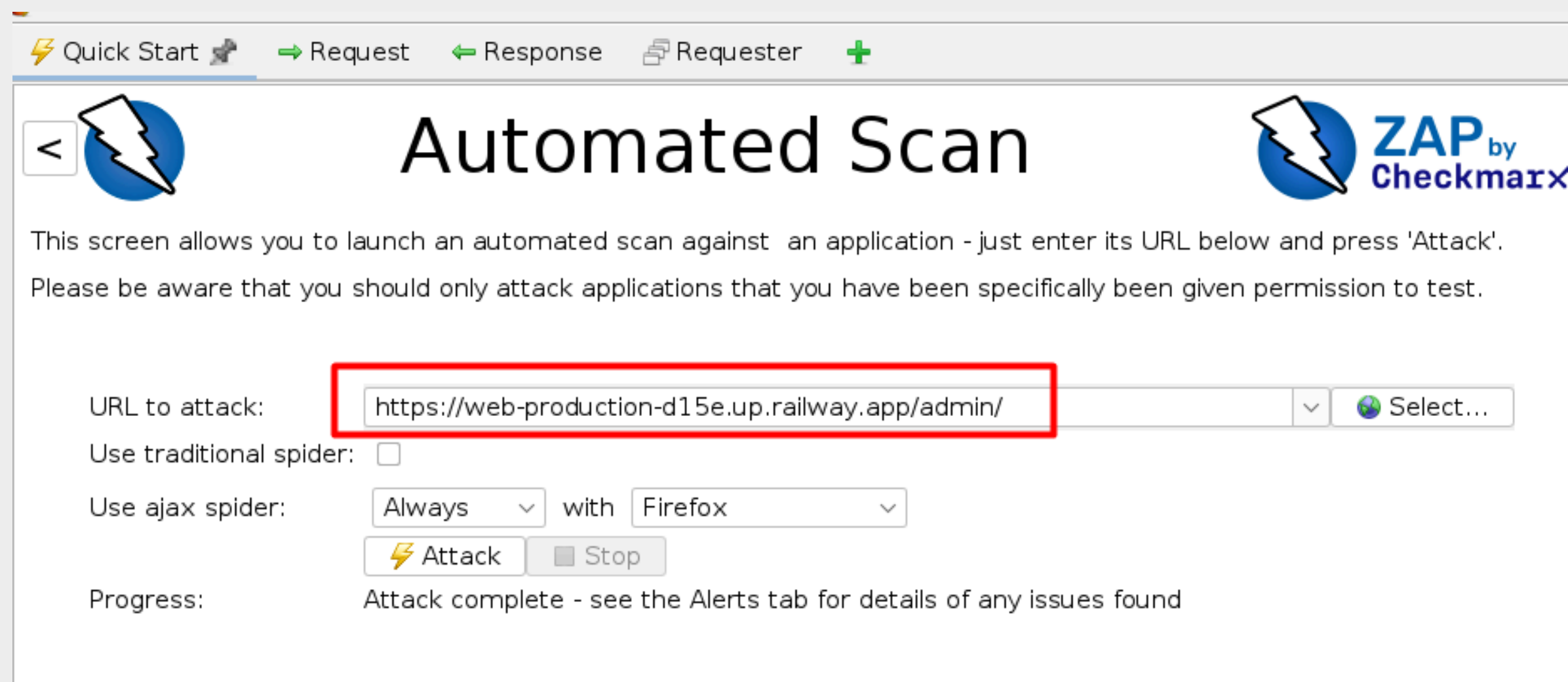


Nikto

# OWASP ZAP

Analiza peticiones HTTP en busca de vulnerabilidades.


Al tener el proyecto parcialmente desplegada se pueda realizar el escaneo de distintas aplicaciones.





The screenshot shows the 'Automated Scan' window of OWASP ZAP. At the top, there is a navigation bar with tabs: 'Quick Start' (active), 'Request', 'Response', 'Requester', and a '+' icon. The main header area contains a back button, a lightning bolt icon, the title 'Automated Scan', and the 'ZAP by Checkmarx' logo. Below the header, a text block explains the purpose of the screen and includes a warning about testing permissions. The 'URL to attack:' field is highlighted with a red rectangle and contains the URL 'https://web-production-d15e.up.railway.app/admin/'. To the right of this field is a dropdown arrow and a 'Select...' button with a globe icon. Below the URL field, there are checkboxes for 'Use traditional spider:' (unchecked) and 'Use ajax spider:' (checked). The 'Use ajax spider:' section includes a dropdown menu set to 'Always' and another dropdown menu set to 'Firefox'. At the bottom of the configuration section are two buttons: 'Attack' (with a lightning bolt icon) and 'Stop' (with a square icon). The 'Progress:' section at the very bottom shows the status 'Attack complete - see the Alerts tab for details of any issues found'.

Quick Start Request Response Requester +

## Automated Scan




This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.  
Please be aware that you should only attack applications that you have been specifically given permission to test.

URL to attack:    Select...

Use traditional spider: ☐

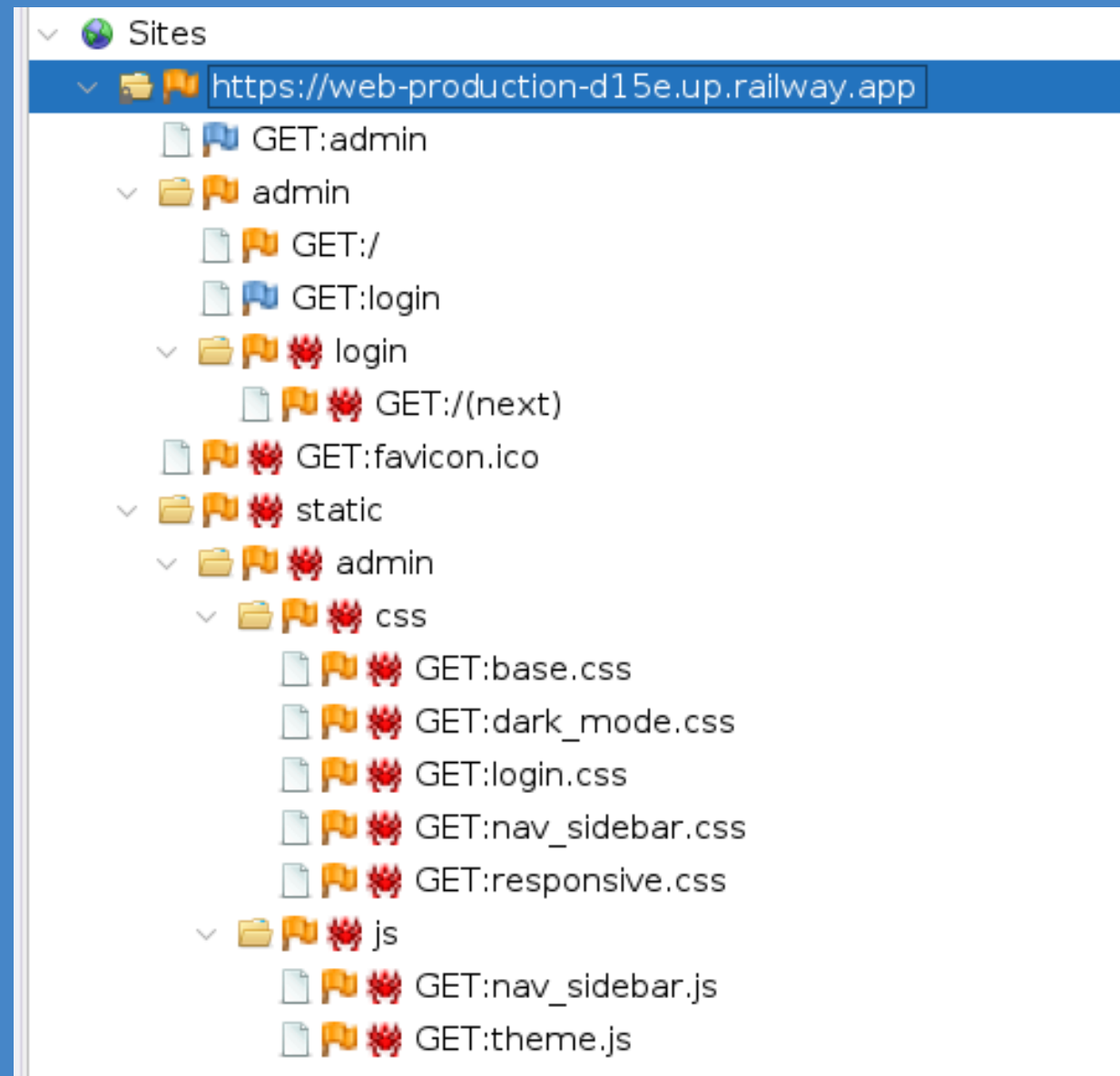
Use ajax spider:  with

 Attack

Progress: Attack complete - see the Alerts tab for details of any issues found

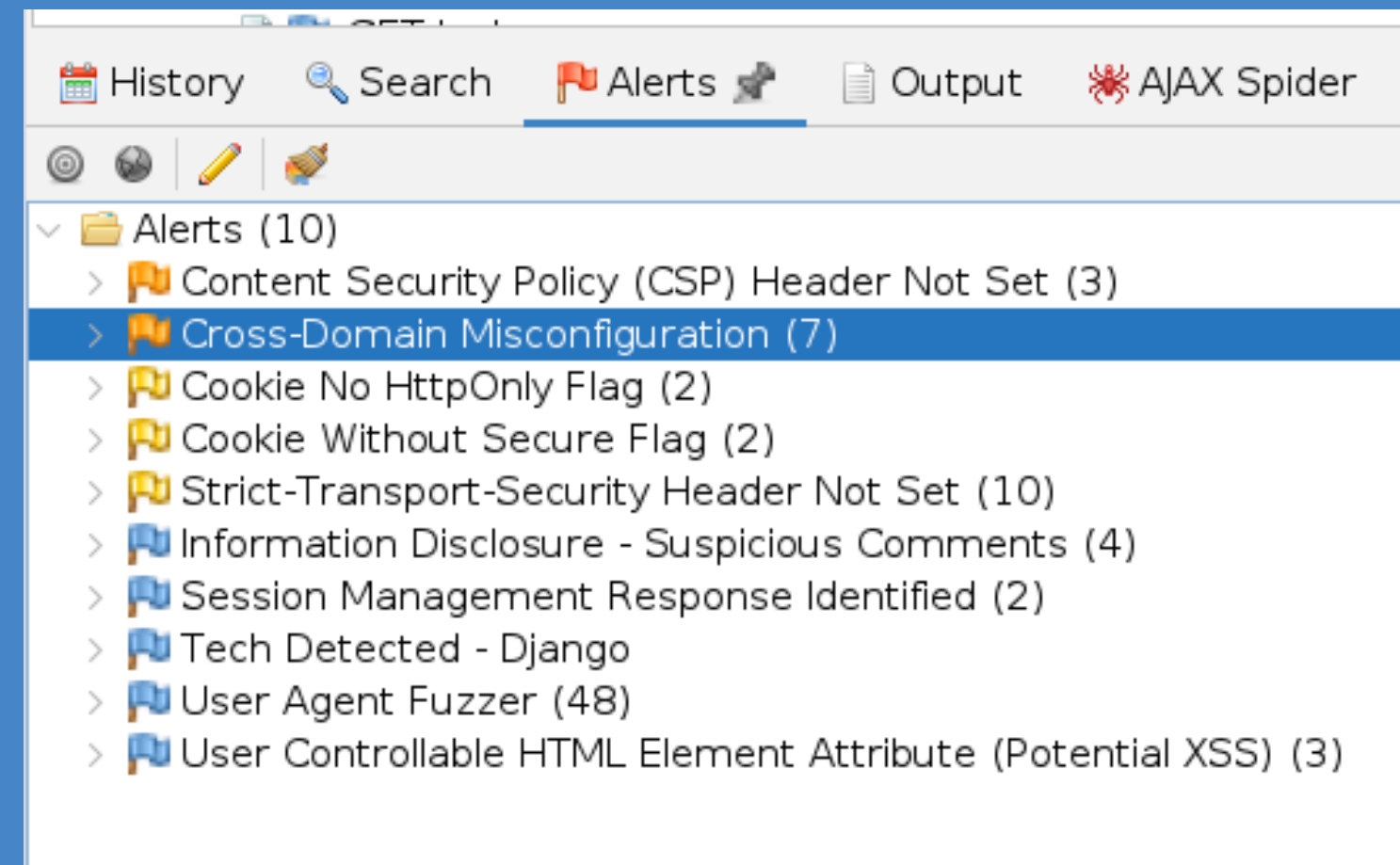
# Navegación

Organización y estructura de la aplicación



# Alertas - top10

Analiza mediante el top 10 de owasp la posibles vulnerabilidades que se podrían encontrar.





# Vulnerabilidad

## Implementación de HTTPS

Asegúrese de que su servidor web, servidor de aplicaciones, balanceador de carga, etc. esté configurado para aplicar la seguridad de transporte estricta.

History

Search

Alerts

Output

AJAX Spider

Active Scan

Alerts (10)

> Content Security Policy (CSP) Header Not Set (3)

> Cross-Domain Misconfiguration (7)

> Cookie No HttpOnly Flag (2)

> Cookie Without Secure Flag (2)

GET: https://web-production-d15e.up.railway.app/admin/

GET: https://web-production-d15e.up.railway.app/admin/login/?ne:

> Strict-Transport-Security Header Not Set (10)

GET: https://web-production-d15e.up.railway.app/admin/

GET: https://web-production-d15e.up.railway.app/admin/login/?ne:

GET: https://web-production-d15e.up.railway.app/favicon.ico

GET: https://web-production-d15e.up.railway.app/static/admin/css

GET: https://web-production-d15e.up.railway.app/static/admin/css

GET: https://web-production-d15e.up.railway.app/static/admin/css

GET: https://web-production-d15e.up.railway.app/static/admin/css

GET: https://web-production-d15e.up.railway.app/static/admin/css

GET: https://web-production-d15e.up.railway.app/static/admin/js/r

GET: https://web-production-d15e.up.railway.app/static/admin/js/t

> Information Disclosure - Suspicious Comments (4)

> Session Management Response Identified (2)

> Tech Detected - Django

> User Agent Fuzzer (48)

CWE ID: 319

WASC ID: 15

Source: Passive (10035 - Strict-Transport-Security Header)

Alert Reference: 10035-1

Input Vector:

Description:  
HTTP Strict Transport Security (HSTS) is a web security policy mechanism whereby a web server declares that complying user agents (such as a web browser) are to interact with it using only secure HTTPS connections (i.e. HTTP layered over TLS/SSL). HSTS is an IETF standards track protocol and is specified in RFC 6797.

Other Info:

Solution:  
Ensure that your web server, application server, load balancer, etc. is configured to enforce Strict-Transport-Security.

Reference:  
[https://cheatsheetseries.owasp.org/cheatsheets/HTTP\\_Strict\\_Transport\\_Security\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html)  
<https://owasp.org/www-community/Security-Headers>  
[https://en.wikipedia.org/wiki/HTTP\\_Strict\\_Transport\\_Security](https://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security)

Alert Tags:

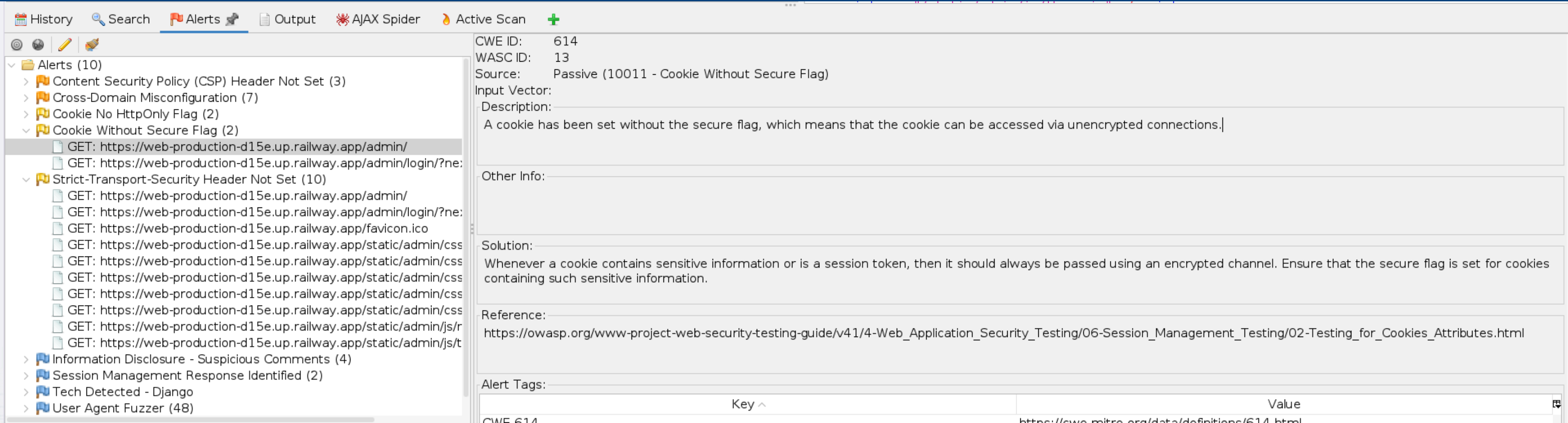
Key

Value

# Vulnerabilidad

Tratamiento de los tokens

Siempre que una cookie contenga información confidencial o sea un token de sesión, debe transmitirse siempre mediante un canal cifrado.





# NIKTO

Escaneo de vulnerabilidades en configuraciones del servidor web.

```
(david@kali)-[~]
$ nikto -h https://web-production-d15e.up.railway.app/admin/ -o reporte.html -Format html
- Nikto v2.5.0
-----
+ Target IP:      35.212.94.98
+ Target Hostname: web-production-d15e.up.railway.app
+ Target Port:    443
-----
+ SSL Info:      Subject: /CN=*.up.railway.app
                  Ciphers: TLS_AES_128_GCM_SHA256
                  Issuer:  /C=US/O=Let's Encrypt/CN=R11
+ Start Time:    2025-02-14 00:15:15 (GMT-6)
-----
```

web-production-  
d15e.up.railway.app /  
35.212.94.98 port 443

Target IP	35.212.94.98
Target hostname	web-production-d15e.up.railway.app
Target Port	443
HTTP Server	railway-edge
Site Link (Name)	<a href="https://web-production-d15e.up.railway.app:443/admin/">https://web-production-d15e.up.railway.app:443/admin/</a>
Site Link (IP)	<a href="https://35.212.94.98:443/admin/">https://35.212.94.98:443/admin/</a>

# Análisis

Identificación de los módulos de la aplicación

```
199a017 /C-03/8-EEC-3-Encrypt/CN-RI1
+ Start Time:      2025-02-14 00:08:16 (GMT-6)
-----
+ Server: railway-edge
V:Fri Feb 14 00:08:17 2025 - 302 for GET:      /admin/
+ /admin/: Uncommon header 'x-railway-request-id' found, with contents: Kkrp3M66QAuLrlURm_V
+ /admin/: The site uses TLS and the Strict-Transport-Security HTTP header is not defined.
+ Root page /admin redirects to: /admin/login/?next=/admin/
V:Fri Feb 14 00:08:17 2025 - 302 for GET:      /admin/admin/
V:Fri Feb 14 00:08:18 2025 - 302 for GET:      /admin/
V:Fri Feb 14 00:08:18 2025 - Testing error for file: /admin/hw5TaDB0.x-shop
V:Fri Feb 14 00:08:18 2025 - 302 for GET:      /admin/hw5TaDB0.x-shop
V:Fri Feb 14 00:08:18 2025 - Testing error for file: /admin/hw5TaDB0.genpopuplist
V:Fri Feb 14 00:08:19 2025 - 302 for GET:      /admin/hw5TaDB0.genpopuplist
V:Fri Feb 14 00:08:19 2025 - Testing error for file: /admin/hw5TaDB0.exe|dir
V:Fri Feb 14 00:08:19 2025 - 302 for GET:      /admin/hw5TaDB0.exe|dir
V:Fri Feb 14 00:08:19 2025 - Testing error for file: /admin/hw5TaDB0.eml
V:Fri Feb 14 00:08:20 2025 - 302 for GET:      /admin/hw5TaDB0.eml
V:Fri Feb 14 00:08:20 2025 - Testing error for file: /admin/hw5TaDB0.TPF
V:Fri Feb 14 00:08:20 2025 - 302 for GET:      /admin/hw5TaDB0.TPF
V:Fri Feb 14 00:08:20 2025 - Testing error for file: /admin/hw5TaDB0.cgi+
V:Fri Feb 14 00:08:21 2025 - 302 for GET:      /admin/hw5TaDB0.cgi+
V:Fri Feb 14 00:08:21 2025 - Testing error for file: /admin/hw5TaDB0.phtml
V:Fri Feb 14 00:08:22 2025 - 302 for GET:      /admin/hw5TaDB0.phtml
V:Fri Feb 14 00:08:22 2025 - Testing error for file: /admin/hw5TaDB0.AP
V:Fri Feb 14 00:08:22 2025 - 302 for GET:      /admin/hw5TaDB0.AP
V:Fri Feb 14 00:08:22 2025 - Testing error for file: /admin/hw5TaDB0.exe
```

# Resultado

De todo la aplicación

```
, -dc /var/lib/NIKTO/plugins/LW2.pm line 3234.
V:Fri Feb 14 00:10:58 2025 - for GET:
+ Scan terminated: 20 error(s) and 4 item(s) reported on remote host
+ End Time:      2025-02-14 00:10:58 (GMT-6) (162 seconds)
-----
```

# Vulnerabilidades

## Header - HTTPS

Se encontró un encabezado poco común 'x-railway-request-id', con contenido:  
gipf7uMgQxmZjeeo3Alu-g\_882434190.

<b>URI</b>	/admin/
<b>HTTP Method</b>	GET
<b>Description</b>	/admin/: Uncommon header 'x-railway-request-id' found, with contents: gipf7uMgQxmZjeeo3Alu-g_882434190.
<b>Test Links</b>	<a href="https://web-production-d15e.up.railway.app:443/admin/">https://web-production-d15e.up.railway.app:443/admin/</a> <a href="https://35.212.94.98:443/admin/">https://35.212.94.98:443/admin/</a>
<b>References</b>	

<b>URI</b>	/admin/
<b>HTTP Method</b>	GET
<b>Description</b>	/admin/: The site uses TLS and the Strict-Transport-Security HTTP header is not defined.
<b>Test Links</b>	<a href="https://web-production-d15e.up.railway.app:443/admin/">https://web-production-d15e.up.railway.app:443/admin/</a> <a href="https://35.212.94.98:443/admin/">https://35.212.94.98:443/admin/</a>
<b>References</b>	<a href="https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security">https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security</a>

El sitio usa TLS y el encabezado HTTP Strict-Transport-Security no está definido.



# Conclusión

Al brindar un mayor enfoque de seguridad en todas las fases de ciclo vida de software, asegura y brinda un mayor nivel de seguridad que es vital hoy en día.