

# Gestión de identidades y credenciales. Caso de estudio en aplicaciones no seguras

Trabajo Fin de Máster

Kevin Carracedo Vázquez

Máster en Tecnologías y Aplicaciones en Ingeniería Informática – Escuela Superior de Ingeniería



UNIVERSIDAD  
DE ALMERÍA

# Índice

1. Introducción
2. Motivación
3. Planificación
4. Objetivos
5. Estado del arte
6. Herramientas y tecnologías
7. Descripción de la solución
8. Pruebas y resultados
9. Problemas encontrados
10. Conclusiones y trabajo futuro

# Índice

1. Introducción
2. Motivación
3. Planificación
4. Objetivos
5. Estado del arte
6. Herramientas y tecnologías
7. Descripción de la solución
8. Pruebas y resultados
9. Problemas encontrados
10. Conclusiones y trabajo futuro

# 1. Introducción

- ❖ La seguridad en aplicaciones es esencial ante el aumento de amenazas y la gestión de datos sensibles.
- ❖ Muchas aplicaciones carecen de sistemas propios de autenticación y control de accesos.
- ❖ El desarrollo de sistemas internos de autenticación es complejo y propenso a errores.
- ❖ Solución: externalizar la gestión de identidades y credenciales usando plataformas especializadas.



# Índice

1. Introducción
2. Motivación
3. Planificación
4. Objetivos
5. Estado del arte
6. Herramientas y tecnologías
7. Descripción de la solución
8. Pruebas y resultados
9. Problemas encontrados
10. Conclusiones y trabajo futuro

## 2. Motivación

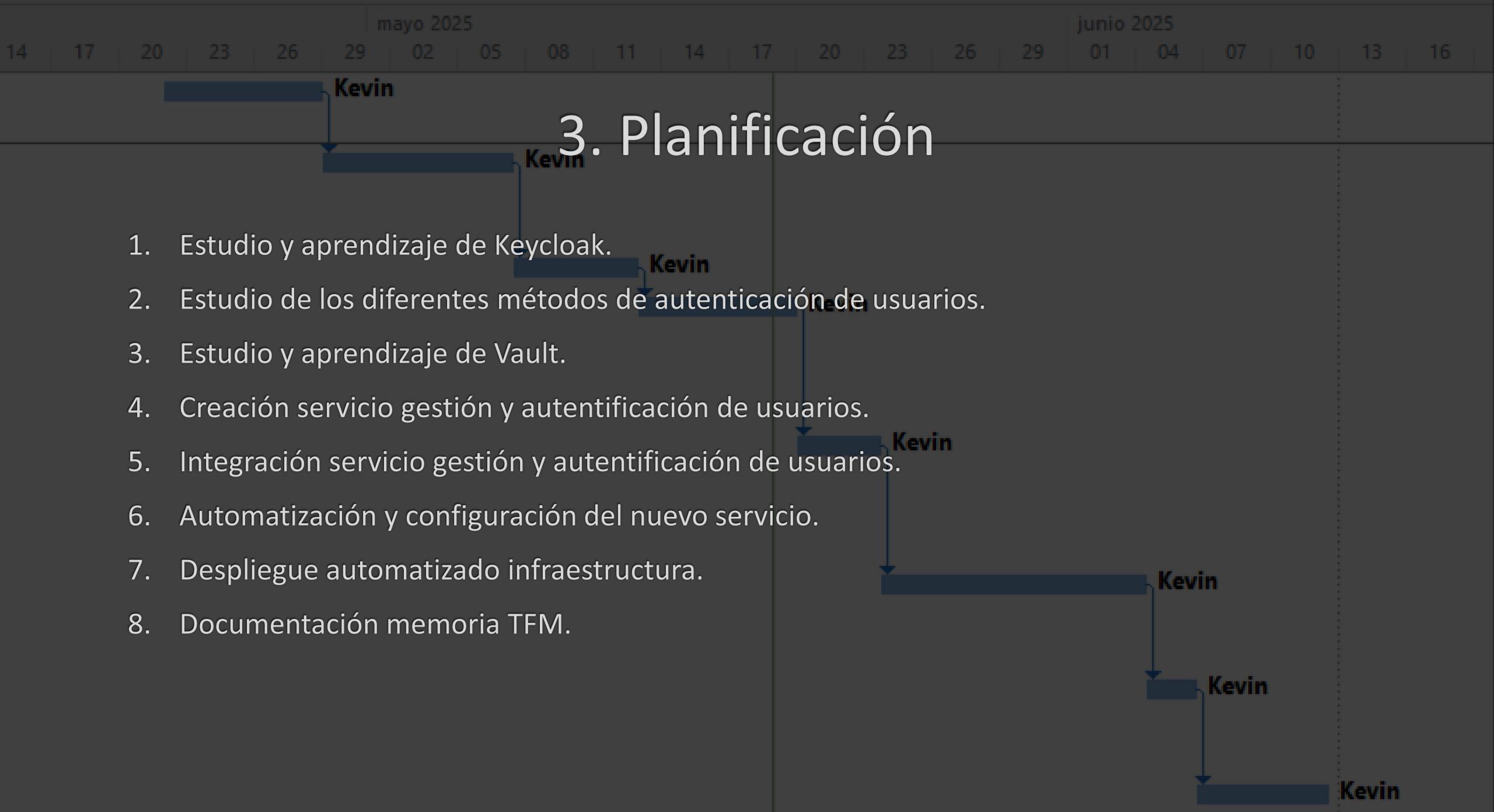
Creciente complejidad y exposición  
de las aplicaciones a amenazas  
(+ Funcionalidad | | + Volumen Datos  
=> + Vulnerabilidad)

Delegación capa de  
seguridad aplicaciones  
(gestión de autorización  
y autentificación) en  
servicios externos

Lógica de negocio de la  
aplicación, cumplir  
estándares, reducir  
riesgos, optimización de  
rendimiento

# Índice

1. Introducción
2. Motivación
3. Planificación
4. Objetivos
5. Estado del arte
6. Herramientas y tecnologías
7. Descripción de la solución
8. Pruebas y resultados
9. Problemas encontrados
10. Conclusiones y trabajo futuro



# Índice

1. Introducción
2. Motivación
3. Planificación
4. Objetivos
5. Estado del arte
6. Herramientas y tecnologías
7. Descripción de la solución
8. Pruebas y resultados
9. Problemas encontrados
10. Conclusiones y trabajo futuro

## 4. Objetivos

### ❖ Objetivo General

- Diseñar e implementar una solución integral de autenticación, autorización y gestión segura de credenciales para aplicaciones sin estas funcionalidades nativas.

### ❖ Objetivos Específicos

- Configurar un servidor de identidad (IdP) para autenticación y autorización.
- Proteger y gestionar credenciales sensibles con un gestor de secretos.
- Implementar ambas herramientas en contenedores aislados.
- Automatizar el despliegue con infraestructura como código (IaC).

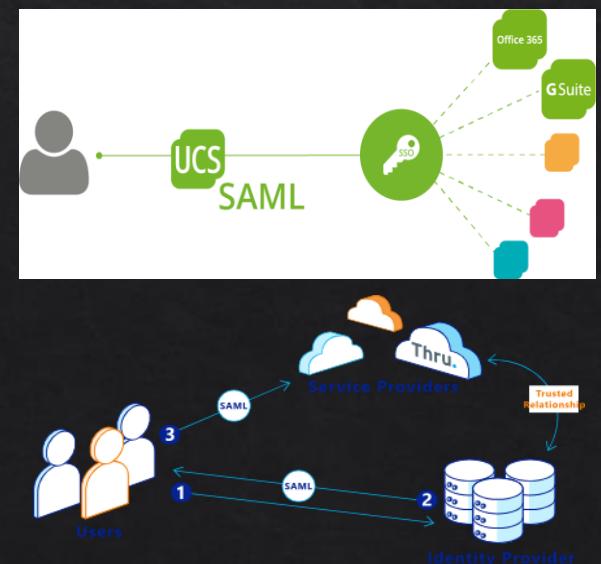
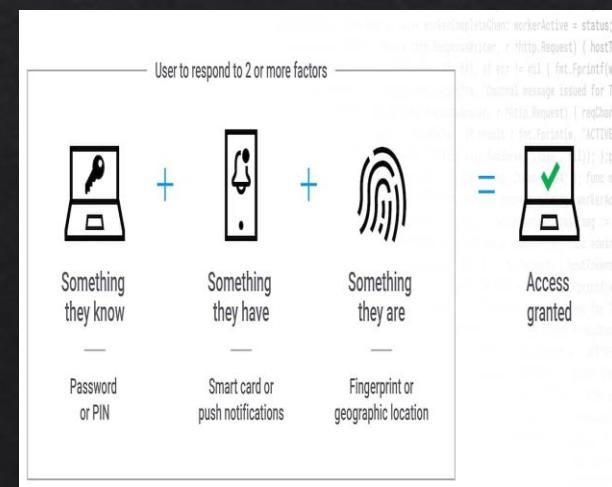
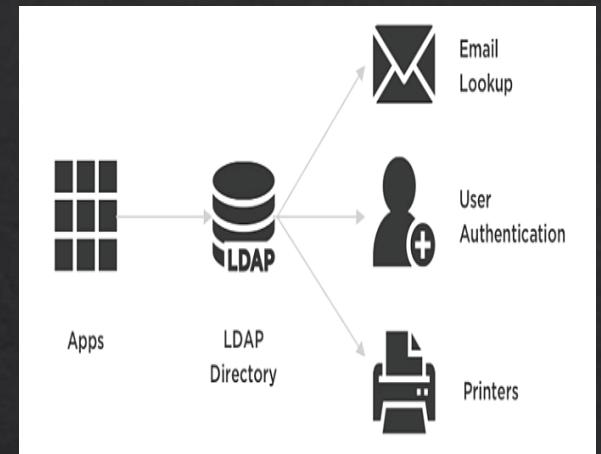
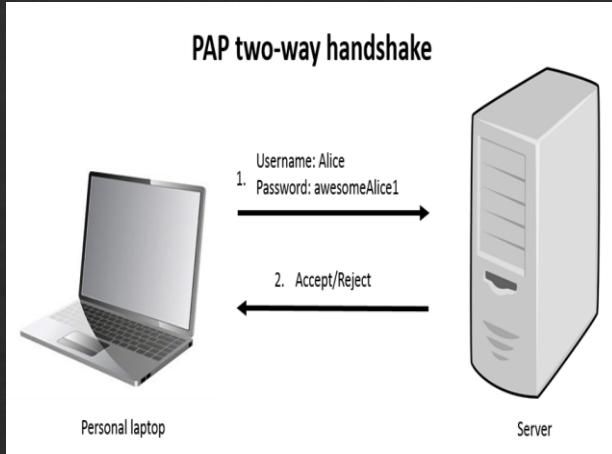
# Índice

1. Introducción
2. Motivación
3. Planificación
4. Objetivos
5. Estado del arte
6. Herramientas y tecnologías
7. Descripción de la solución
8. Pruebas y resultados
9. Problemas encontrados
10. Conclusiones y trabajo futuro

# Métodos de autenticación tradicionales

- PAP: Contraseñas estáticas, vulnerables a ataques y reutilización.
- LDAP: Administración centralizada de usuarios, pero dependiente de contraseñas.
- MFA: Añade factores de autenticación, incrementando seguridad.
- SAML y SSO: Interoperabilidad y acceso único a múltiples aplicaciones.

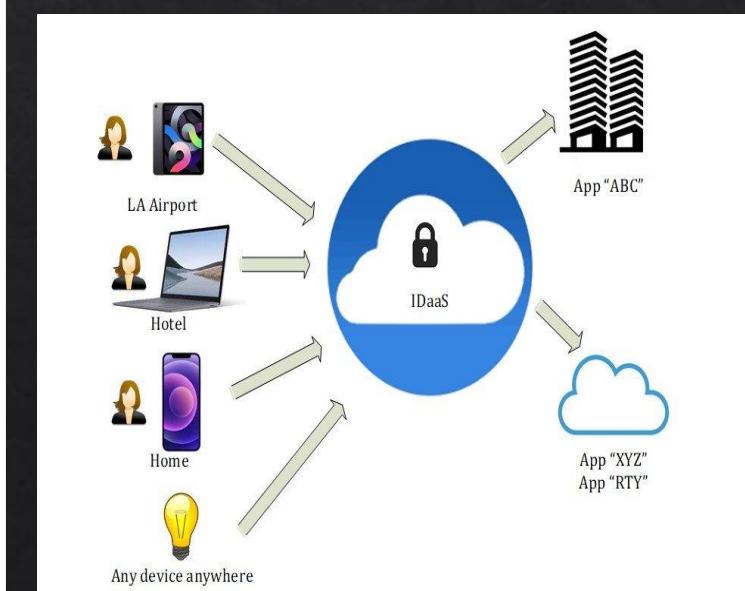
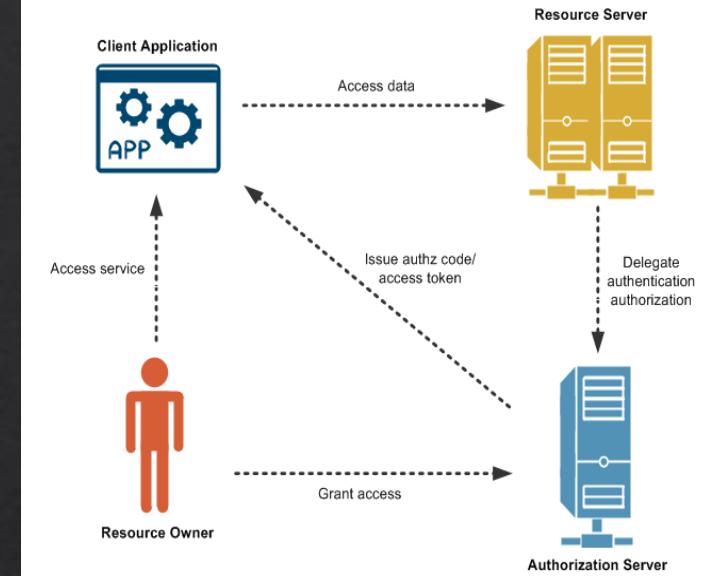
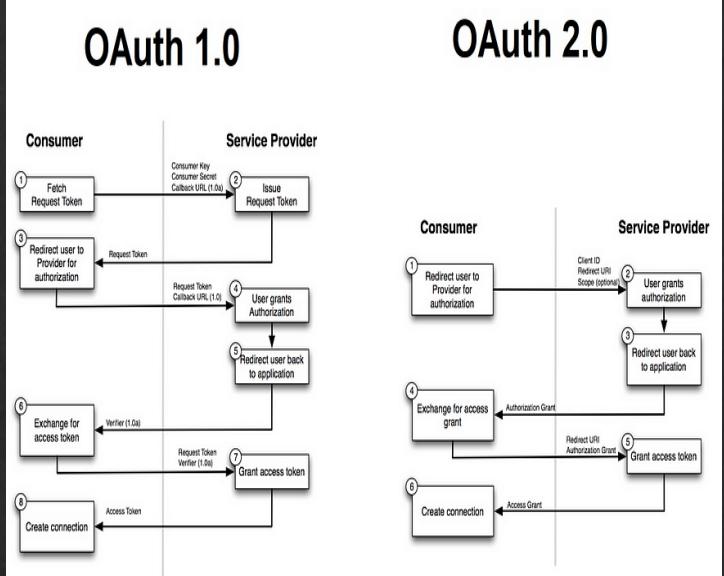
Limitaciones: escalabilidad, usabilidad, cumplimiento.



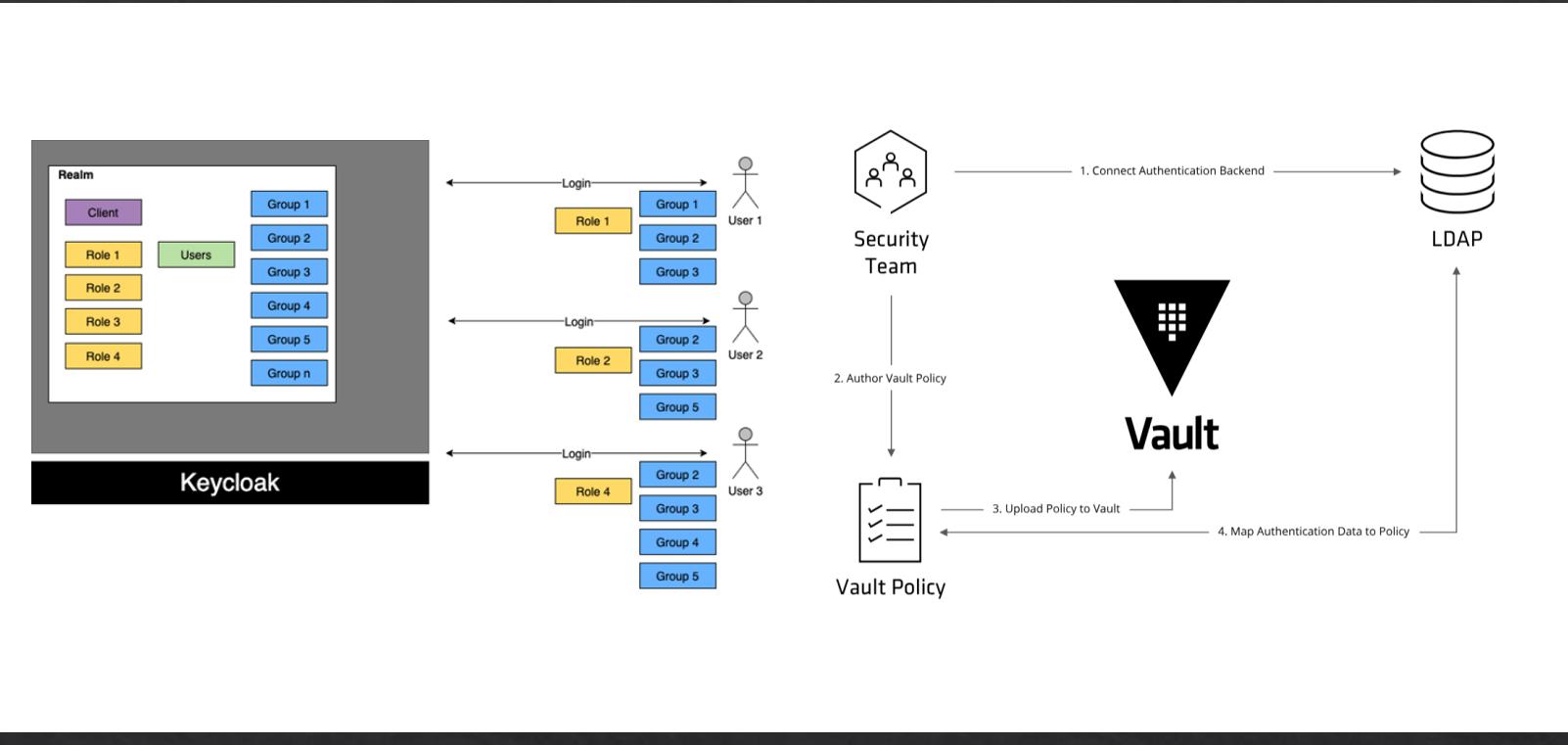
# Métodos de autenticación modernos

- OAuth 2.0: Delegación de acceso para aplicaciones de terceros.
- OpenID Connect (OIDC): Capa de autenticación sobre OAuth.
- JWT: Tokens compactos y seguros.
- IDaaS: Gestión de identidades como servicio en la nube.

Ventajas: flexibilidad, interoperabilidad.



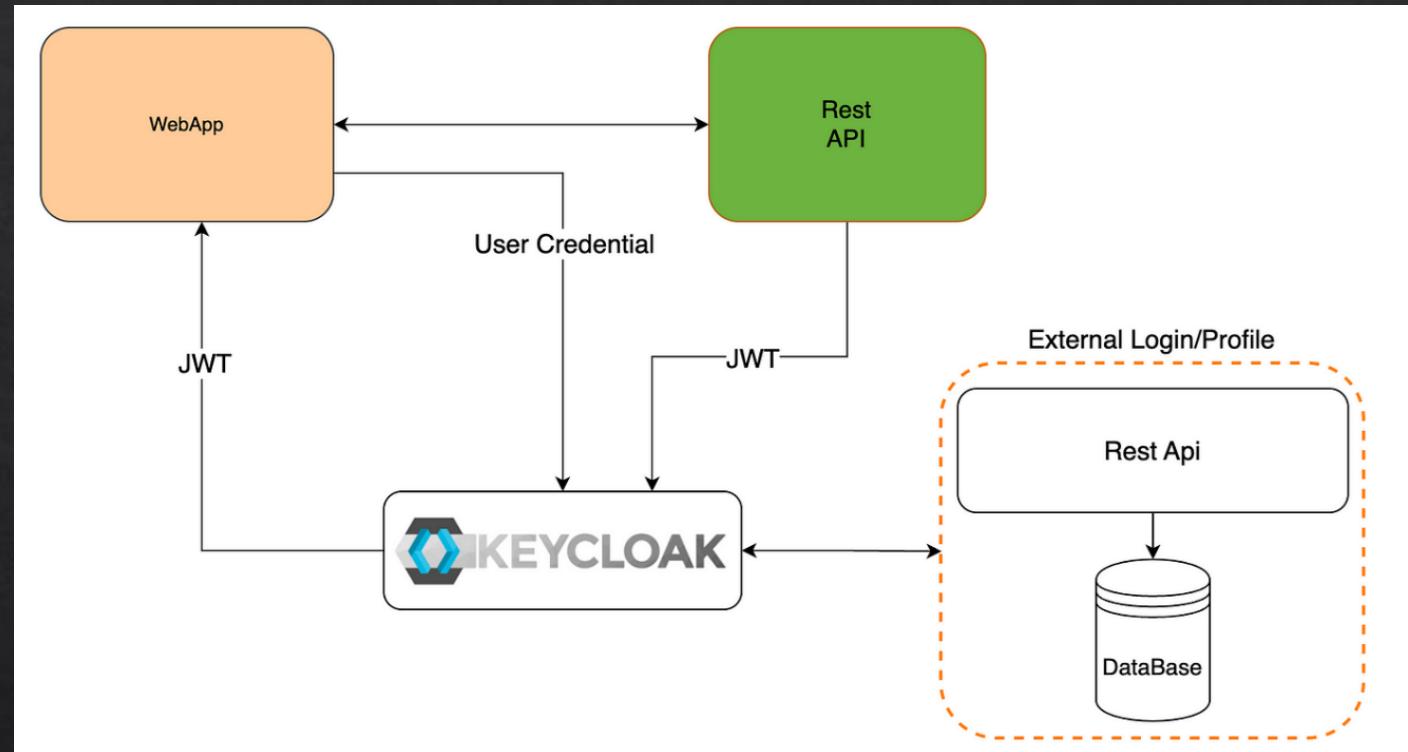
Método	Origen / Año	Tipo	Mecanismo / Factores	Ventajas principales	Desventajas principales	Uso típico
PAP (Password Access Protocol)	MIT, CTSS (década de 1960)	Autenticación	Contraseña (algo que el usuario sabe)	– Muy sencillo de implementar – Compatibilidad universal	– Vulnerable a ataques de fuerza bruta y phishing – Reutilización de contraseñas – Gestión segura del almacenamiento crítico	Sistemas heredados, acceso básico a terminales
LDAP	ISO X.500 / DAP simplificado (1990s)	Autenticación y autorización	Contraseña + atributos en directorio centralizado	– Gestión centralizada de identidades – Integración con múltiples sistemas – Buena escalabilidad	– Sigue dependiendo de contraseñas – Requiere conocimientos específicos para su implementación y mantenimiento	Gestión de usuarios y permisos en entornos corporativos
MFA (Autenticación multifactor)	Finales de los 1990 / principios de 2000	Autenticación	Varios factores independientes (conocimiento, posesión, inherencia)	– Aumenta drásticamente la seguridad – Dificulta el acceso con credenciales comprometidas	– Coste adicional (tokens, aplicaciones móviles) – Mayor complejidad para el usuario	Acceso a servicios financieros, entornos críticos
SAML / SSO	OASIS (2002)	Autenticación y autorización	Aserciones XML intercambiadas entre proveedor de identidad (IdP) y proveedor de servicio (SP)	– Interoperabilidad entre dominios – Menos inicios de sesión repetidos, mejor experiencia de usuario	– Implementación compleja – Menor adecuación a arquitecturas de microservicios y móviles	Federaciones de identidad en empresas, aplicaciones web
OAuth 1.0 / OAuth 2.0	OAuth 1.0 (2007) / OAuth 2.0 (2012)	Autorización	Tokens de acceso delegados (flujos de concesión)	– Delegación de permisos sin compartir credenciales – Adaptable a distintos tipos de aplicación	– No define un mecanismo de autenticación de usuario – Peligro si se configura incorrectamente	Integración de aplicaciones con redes sociales y API
OpenID Connect (OIDC)	Fundación OpenID (2014)	Autenticación y autorización	Capa de identidad sobre OAuth 2.0 (ID Token en JWT)	– Añade autenticación estándar a OAuth 2.0 – Uso de JWT para transmitir identidad de forma segura – Amplia interoperabilidad	– Ligeramente más complejo que OAuth 2.0 puro – Gestión de tokens y validación de JWT	Inicio de sesión único en aplicaciones web y móviles
JSON Web Tokens (JWT)	RFC 7519 (2015)	Autenticación y autorización	Token compacto firmado (JSON) y sin estado	– Arquitectura sin estado en el servidor – Ligero y portable – Integridad garantizada criptográficamente	– Dificultad para revocar tokens – Exposición de claves si no se protegen adecuadamente	Microservicios, API REST, flujos de OIDC/OAuth
IDaaS (Identity as a Service)	SaaS en la nube (2010s)	Autenticación y autorización	Servicio en la nube compatible con OAuth, OIDC y SAML	– Gestión de identidades centralizada en la nube – Alta escalabilidad y flexibilidad – No requiere infraestructura local	– Dependencia del proveedor del servicio – Retos de cumplimiento normativo (GDPR) – Integración con sistemas heredados	Gestión de usuarios en entornos multicloud y SaaS



# La propuesta de Keycloak y Vault

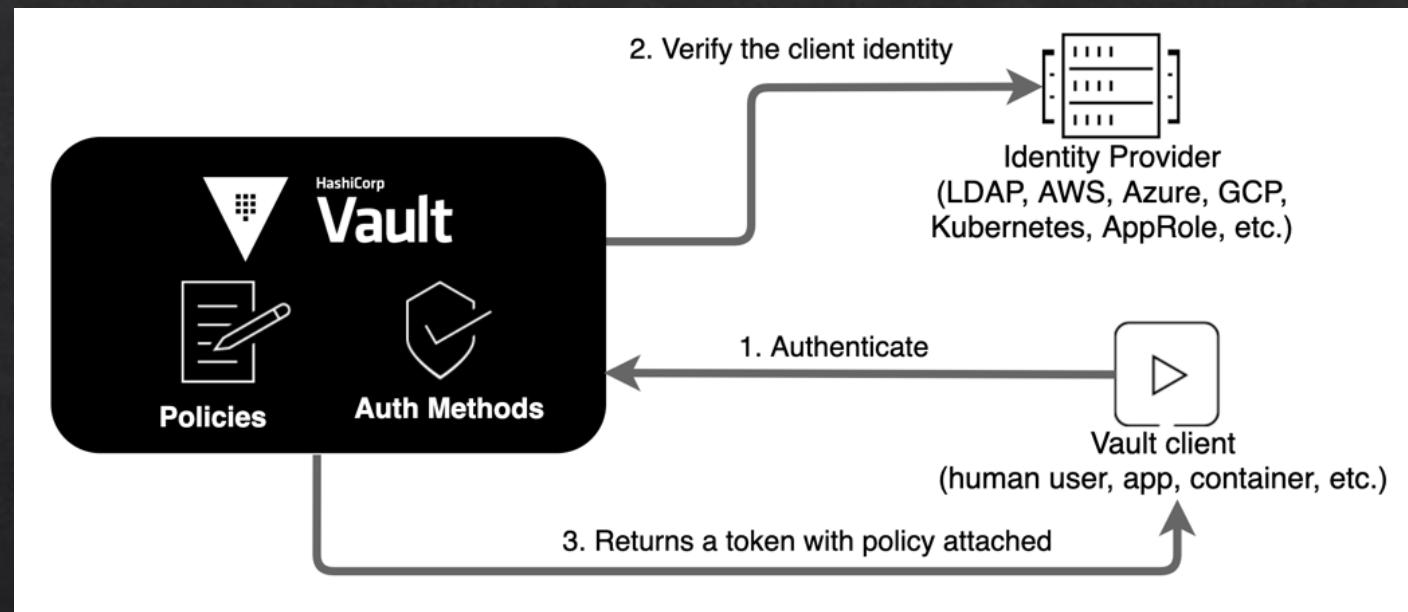
# Keycloak

Es una solución de gestión de identidades y accesos (IAM) de código abierto desarrollada por Red Hat. Proporciona autenticación y autorización centralizadas a aplicaciones y servicios, incluyendo inicio de sesión único (SSO) y soportando los protocolos estándar OAuth 2.0, OpenID Connect y SAML 2.0. Permite integrar fuentes externas de usuarios (LDAP/Active Directory), definir flujos de autenticación personalizados (MFA, preguntas de seguridad), y gestionar roles y permisos (RBAC) de forma centralizada.



# Vault

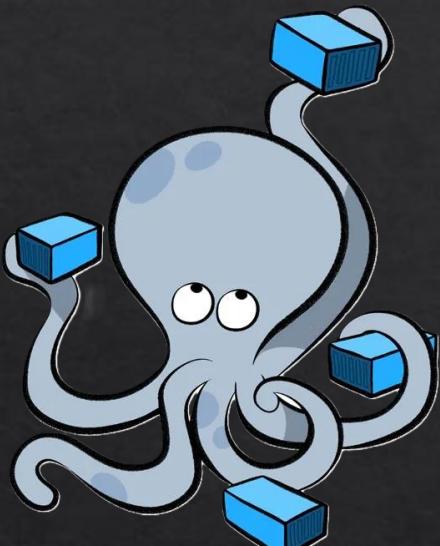
Es una herramienta de gestión de secretos de código abierto creada por HashiCorp. Se encarga de almacenar y controlar el acceso a información sensible (contraseñas, claves API, certificados) cifrados, y de generar credenciales dinámicas y secretos con tiempo de vida limitado para los servicios y aplicaciones que lo integren, con un sistema de políticas y backend de almacenamiento seguros.



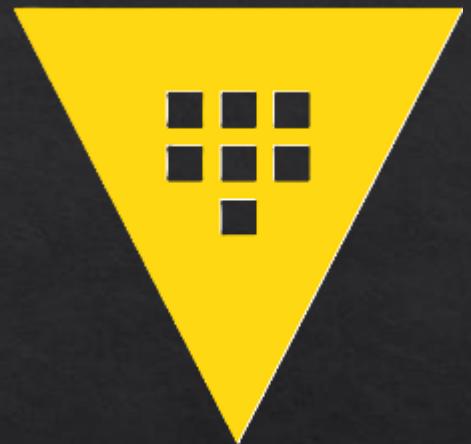
# Índice

1. Introducción
2. Motivación
3. Planificación
4. Objetivos
5. Estado del arte
6. Herramientas y tecnologías
7. Descripción de la solución
8. Pruebas y resultados
9. Problemas encontrados
10. Conclusiones y trabajo futuro

# 6. Herramientas y tecnologías



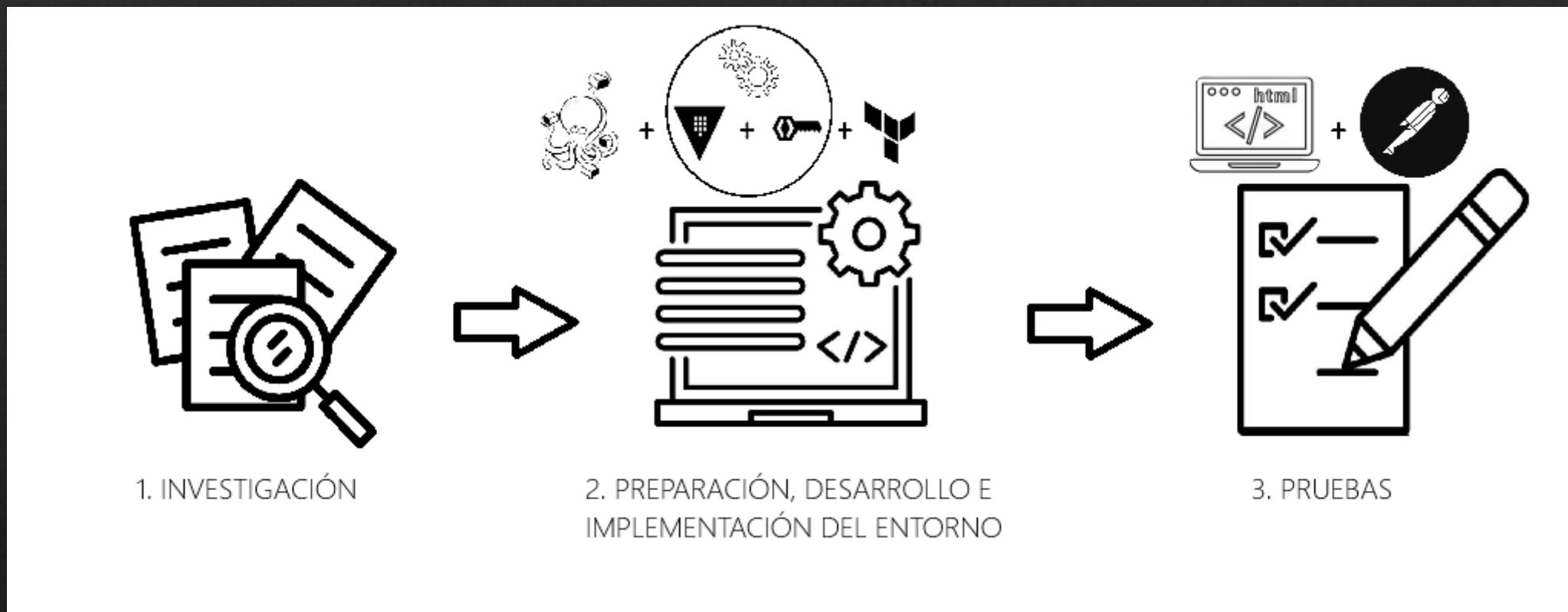
Tecnologías	Función principal
Docker Compose	Contenedorización, aislamiento y orquestación de servicios
Keycloak	Gestión centralizada de identidades y accesos (IAM)
Vault	Gestión y protección de secretos
Terraform	Automatización y despliegue de la infraestructura



# Índice

1. Introducción
2. Motivación
3. Planificación
4. Objetivos
5. Estado del arte
6. Herramientas y tecnologías
7. Descripción de la solución
8. Pruebas y resultados
9. Problemas encontrados
10. Conclusiones y trabajo futuro

## 7. Descripción de la solución

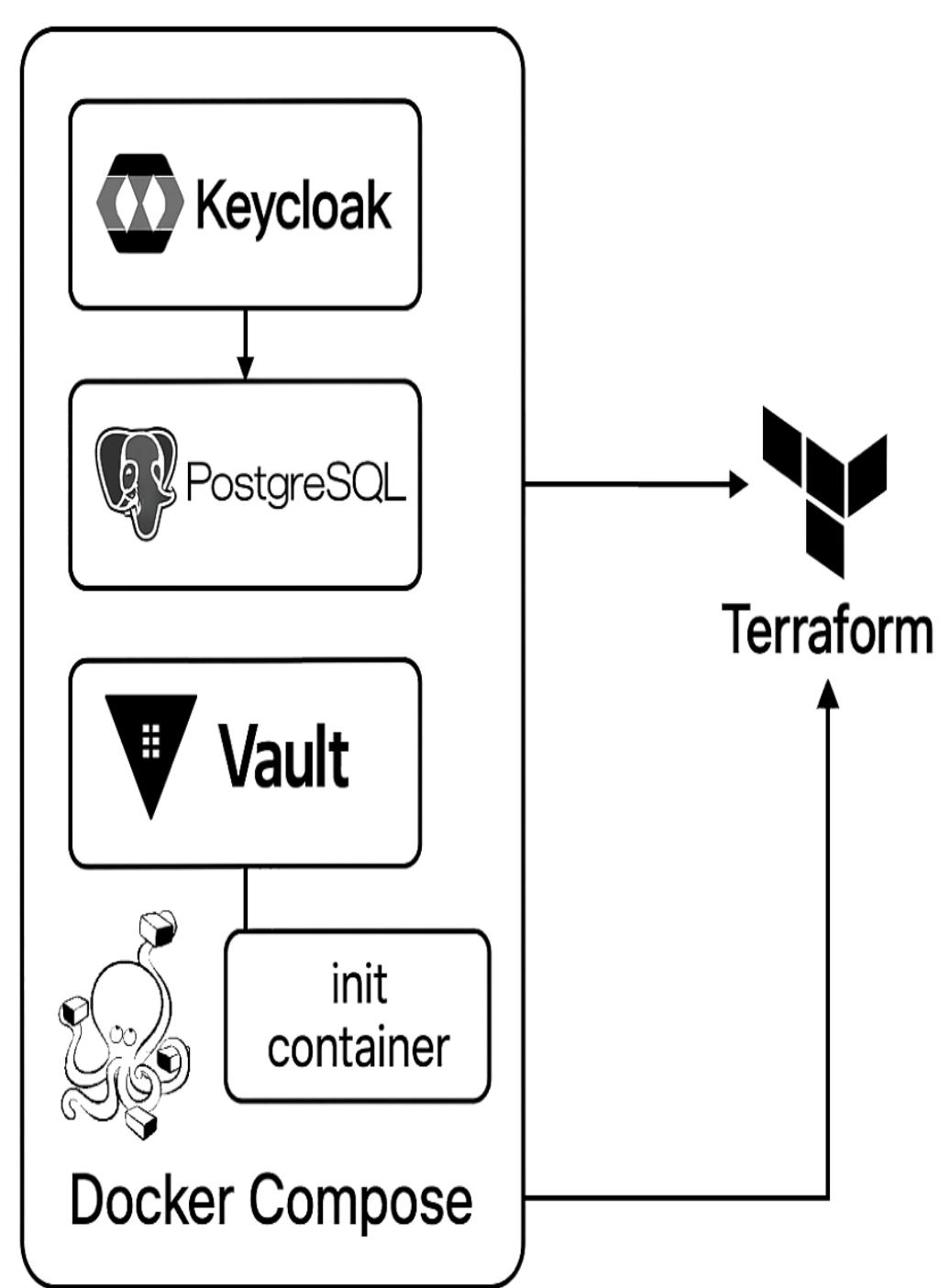


# Fase 1 - Investigación



1. INVESTIGACIÓN





# Diseño de la solución

- ❖ Keycloak como proveedor de identidad (OIDC).
- ❖ Vault como gestor centralizado de secretos.
- ❖ Docker Compose para orquestar servicios, Terraform para crear realms, usuarios y políticas.
- ❖ Terraform para automatización y despliegue de la infraestructura.

Aplicaciones Python (Flask) federadas con Keycloak. Vault consume tokens JWT de Keycloak y gestiona accesos a secretos.

## Fase 2 – Preparación, desarrollo e implementación del entorno



```
15          KC_DB_USERNAME: kcv239
16          KC_DB_PASSWORD: inlumine.ual.es
17      ports:
18
19      keycloak:
20          image: quay.io/keycloak/keycloak:26.1.3
21          container_name: keycloak
22          hostname: keycloak
23          user: root
24          environment:
25              KC_BOOTSTRAP_ADMIN_USERNAME: admin
26              KC_BOOTSTRAP_ADMIN_PASSWORD: admin
27              KC_DB: postgres
28              KC_DB_URL_HOST: keycloak-db
29              KC_DB_URL_PORT: 5432
30              KC_DB_URL_DATABASE: keycloak
31              KC_DB_USERNAME: kcv239
32              KC_DB_PASSWORD: inlumine.ual.es
33      ports:
34          - "8080:8080"
35          - "8443:8443"
36      networks:
37          - tfm-net
38      volumes:
39          - ./keycloak/volume/data:/opt/keycloak/data
40          - ./keycloak/init-keycloak.sh:/keycloak/init-scripts/init-keycloak.sh
41          - ./keycloak/providers:/opt/keycloak/providers
42      entrypoint: ["/bin/sh", "/keycloak/init-scripts/init-keycloak.sh"]
43      depends_on:
44          - keycloak-db
45      restart: on-failure
46
47      restart: on-failure
48
```

# Keycloak

```
data:/opt/keycloak/data
keycloak.sh:/keycloak/init-scripts/init-keycloak.sh
rs:/opt/keycloak/providers
, "/keycloak/init-scripts/init-keycloak.sh"]
```

```
15      KC_DB_USERNAME: kcv239
16      KC_DB_PASSWORD: inlumine.ual.es
17      ports:
18
19      keycloak-db:
20          image: postgres:13.3
21          container_name: keycloak-db
22          hostname: keycloak-db
23          environment:
24              POSTGRES_DB: keycloak
25              POSTGRES_USER: kcv239
26              POSTGRES_PASSWORD: inlumine.ual.es
27          volumes:
28              - ./db/volume/postgresql/data:/var/lib/postgresql/data
29          ports:
30              - "5432:5432"
31
32      restart: on-failure
```

# PostgreSQL

```
data:/opt/keycloak/data
keycloak.sh:/keycloak/init-scripts/init-keycloak.sh
rs:/opt/keycloak/providers
, "/keycloak/init-scripts/init-keycloak.sh"]
```

```
15          KC_DB_USERNAME: kcv239
16          KC_DB_PASSWORD: inlumine.ual.es
17      ports:
18
19      vault:
20          image: hashicorp/vault:1.18
21          container_name: vault
22          hostname: vault
23          ports:
24              - "8200:8200"
25          volumes:
26              - ./vault/config:/vault/config
27              - ./vault/volume/data:/vault/data
28              - ./vault/volume/logs:/vault/logs
29          command: ["sh", "-c", "vault server -config=/vault/config/config.hcl"]
30          cap_add:
31              - IPC_LOCK
32          networks:
33              - tfm-net
34          restart: on-failure
35
36      vault-init:
37          image: hashicorp/vault:1.18
38          container_name: vault-init
39          depends_on:
40              - vault
41          volumes:
42              - ./vault/init-vault.sh:/init-vault.sh
43              - ./vault/volume/data:/vault/data
44          networks:
45              - tfm-net
46          entrypoint: ["sh", "-c", "apk add --no-cache curl jq && sh /init-vault.sh"]
47          restart: on-failure
48
49      restart: on-failure
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
```

# Vault

```
data:/opt/keycloak/data
keycloak.sh:/keycloak/init-scripts/init-keycloak.sh
rs:/opt/keycloak/providers
, "/keycloak/init-scripts/init-keycloak.sh"]
```

# Terraform providers

```
terraform {  
    required_providers {  
        keycloak = {  
            source  = "mrparkers/keycloak"  
            version = ">= 3.6.0"  
        }  
        vault = {  
            source  = "hashicorp/vault"  
            version = ">= 3.14.0"  
        }  
    }  
  
    provider "keycloak" {  
        url      = var.keycloak_url  
        client_id = var.keycloak_client_id  
        username  = var.keycloak_username  
        password  = var.keycloak_password  
        realm     = var.keycloak_realm  
    }  
  
    provider "vault" {  
        address = var.vault_address  
        token   = local.vault_token  
    }  
}
```

# Realm “tfm”

```
# Realm "tfm"
resource "keycloak_realm" "tfm" {
    realm      = var.keycloak_realm_name
    enabled    = var.keycloak_realm_enabled
}

# Cliente "vault"
resource "keycloak_openid_client" "vault" {
    realm_id          = keycloak_realm.tfm.id
    client_id         = var.vault_client_id
    name              = var.vault_client_name
    enabled           = var.vault_client_enabled
    standard_flow_enabled = var.vault_standard_flow_enabled
    access_type        = var.vault_access_type
    service_accounts_enabled = var.vault_service_accounts_enab
    client_secret      = var.vault_client_secret
    valid_redirect_uris = var.vault_valid_redirect_uris
    web_origins         = var.vault_web_origins
}

# Crear el usuario "kcv239"
resource "keycloak_user" "kcv239" {
    realm_id = keycloak_realm.tfm.id
    username = var.kcv239_username
    enabled   = var.kcv239_enabled

    initial_password {
        temporary = var.kcv239_temporary_password
        value     = var.kcv239_password_value
    }
}
```

# Configuración de login OIDC Vault

```
# Configuración OIDC usando vault_generic_endpoint
resource "vault_generic_endpoint" "oidc_config" {
    depends_on = [keycloak_realm.tfm]
    path       = "auth/${vault_auth_backend.oidc.path}/config"
    disable_read = false
    data_json = jsonencode({
        oidc_discovery_url = "${var.keycloak_url}/realms/tfm",
        oidc_client_id     = var.vault_client_id,
        oidc_client_secret = var.vault_client_secret,
        default_role       = var.oidc_default_role
    })
}

# Definición del rol para OIDC vault
resource "vault_generic_endpoint" "oidc_role" {
    path = "auth/${vault_auth_backend.oidc.path}/role/default"
    data_json = jsonencode({
        allowed_redirect_uris = var.oidc_allowed_redirect_uris,
        user_claim            = var.oidc_user_claim,
        role_type              = var.oidc_role_type,
        oidc_scopes            = var.oidc_scopes,
        bound_issuer           = "${var.keycloak_url}/realms/tfm",
        policies               = var.oidc_policies,
        ttl                   = var.oidc_ttl
    })
}

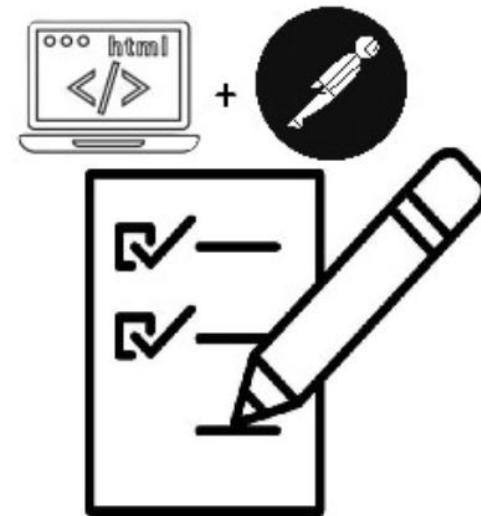
# Política "default" vault
resource "vault_policy" "default" {
    name    = var.vault_policy_name
    policy  = var.vault_policy_content
}
```

# Federar aplicación

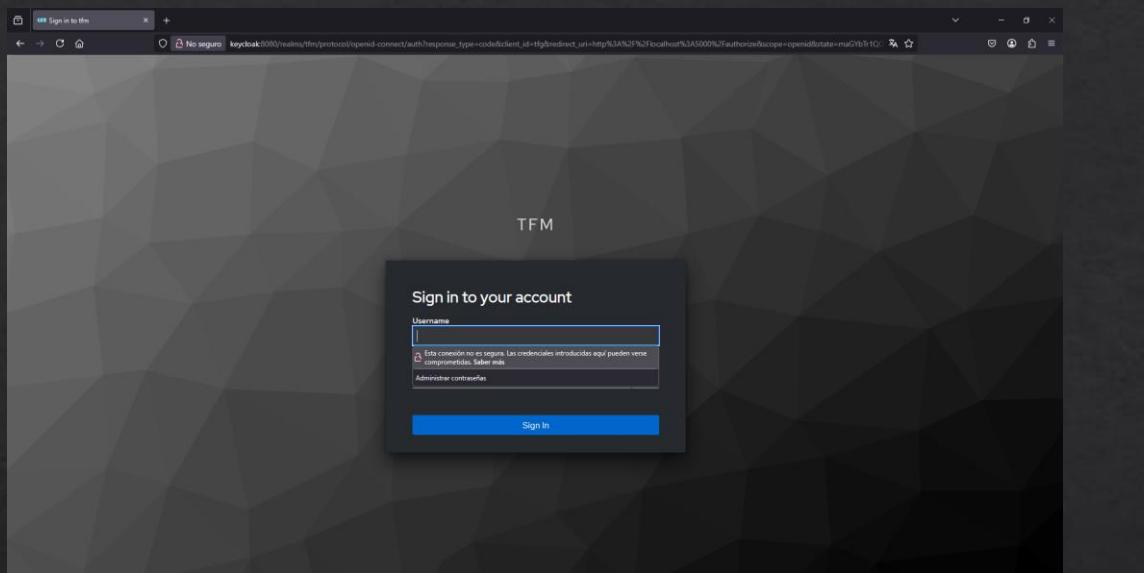
```
{  
    "web": {  
        "issuer": "http://keycloak:8080/realm/tfm",  
        "auth_uri": "http://keycloak:8080/realm/tfm/protocol/openid-connect/auth",  
        "client_id": "tfg",  
        "client_secret": "inlumine.ual.es",  
        "redirect_uris": ["http://localhost:5000/*"],  
        "token_uri": "http://keycloak:8080/realm/tfm/protocol/openid-connect/token",  
        "userinfo_uri": "http://keycloak:8080/realm/tfm/protocol/openid-connect/userinfo"  
    }  
  
    # Configuracion de OIDC  
    app.config.update({  
        'OIDC_CLIENT_SECRETS': './client_secrets.json',  
        'OIDC_ID_TOKEN_COOKIE_SECURE': False,  
        'OIDC_SCOPES': ['openid'],  
        'OIDC_INTROSPECTION_AUTH_METHOD': 'client_secret_post',  
        'SESSION_TYPE': 'filesystem',  
        'SESSION_FILE_DIR': '/tmp/flask_session',  
        'SESSION_PERMANENT': False  
    })  
  
    Session(app)  
  
    oidc = OpenIDConnect(app)
```

```
@app.route('/')  
@oidc.require_login  
def home():  
    return render_template('index.html')  
  
@app.route('/login')  
def login():  
    return redirect(url_for('home'))  
  
@app.route('/logout')  
def logout():  
    oidc.logout()  
    flash("Sesión cerrada")  
    return redirect(url_for('home'))  
  
@app.route('/loadInitCSV', methods=['GET'])  
@oidc.require_login  
def upload_file():  
    return render_template('subida_fichero.html')  
  
@app.route('/uploadInitCSV', methods=['POST'])  
def uploader():  
    if request.method == 'POST':  
        file_form = request.files['file_request']  
        file = secure_filename(file_form.filename)  
  
        filename = file.split('.').  
        f_name = filename[0]  
        f_extension = filename[-1]
```

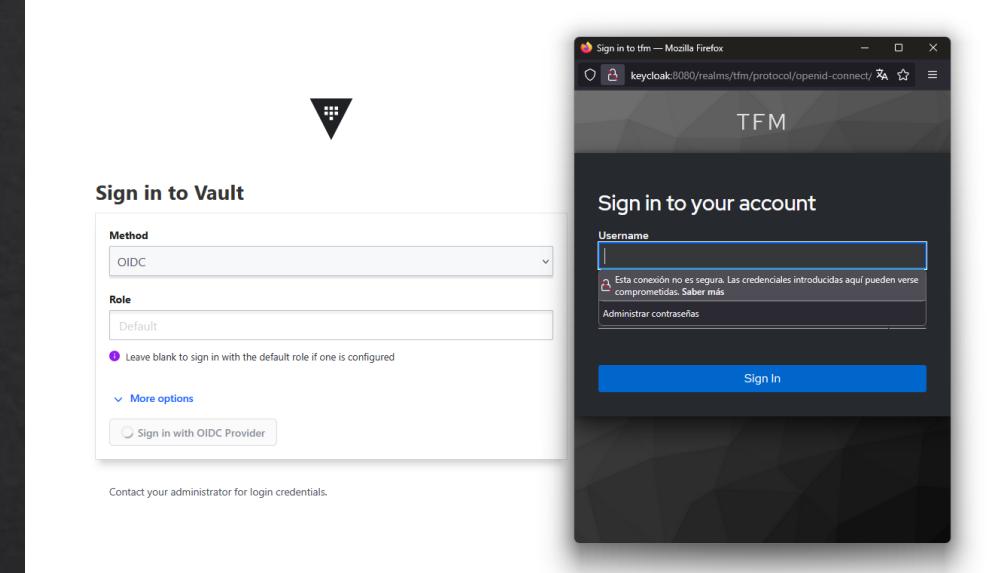
# Fase 3 - Pruebas



3. PRUEBAS

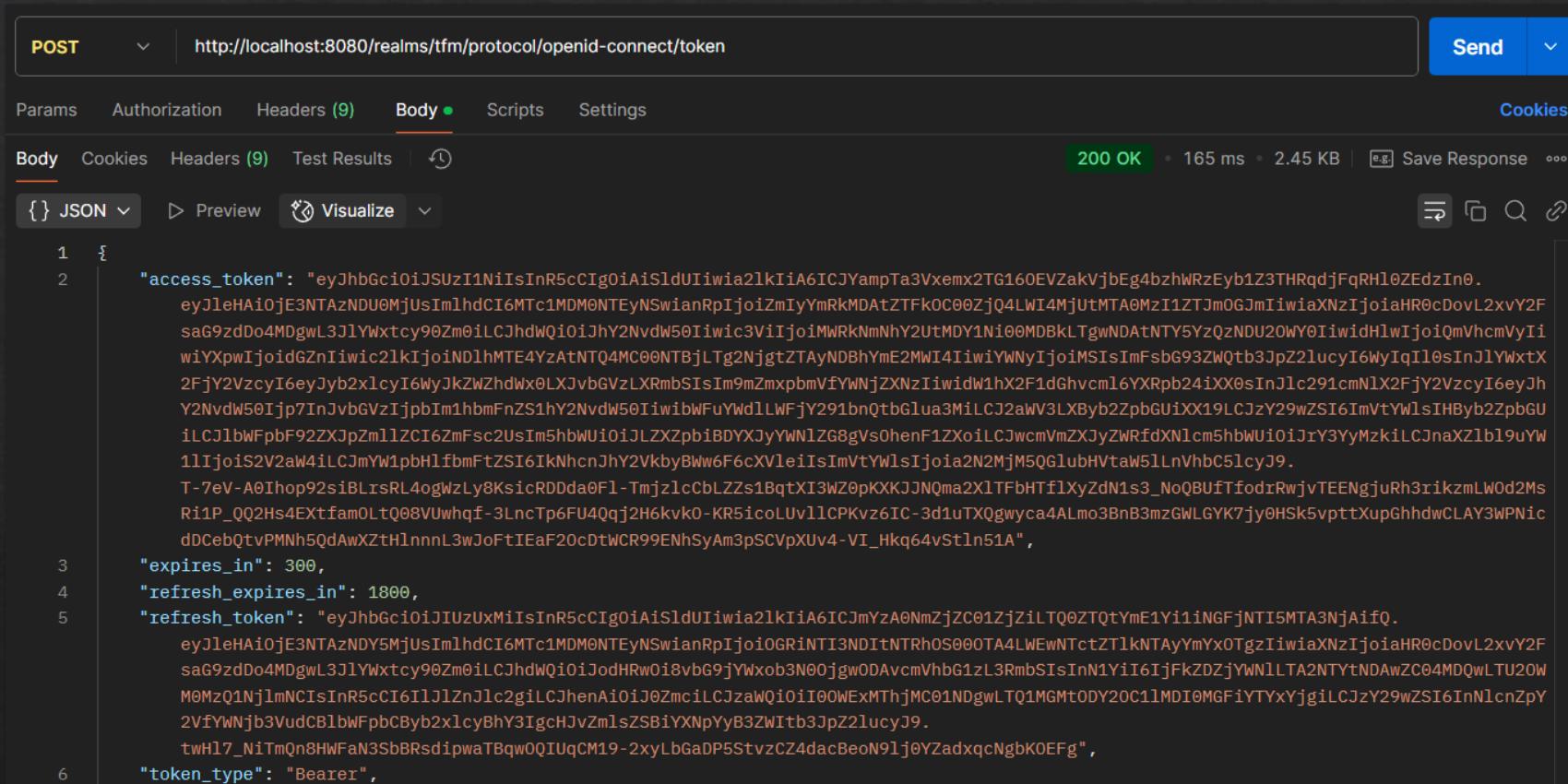


Login OIDC app



Login OIDC Vault

# Token usuario registrado en cliente Keycloak



A screenshot of the Postman application interface. The top bar shows a POST request to `http://localhost:8080/realm/tfm/protocol/openid-connect/token`. The 'Body' tab is selected, showing a JSON response with a large access token string. The status bar at the bottom indicates a 200 OK response with 165 ms duration and 2.45 KB size.

```
POST http://localhost:8080/realm/tfm/protocol/openid-connect/token
```

Params Authorization Headers (9) **Body** Scripts Settings Cookies

Body Cookies Headers (9) Test Results ⏱

200 OK • 165 ms • 2.45 KB | e.g. Save Response ⋮

{ } JSON ▾ ▶ Preview ⏷ Visualize ▾

```
1 {  
2   "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJYampTa3Vxemx2TG160EVZakVjbEg4bzhWRzEyb1Z3THRqdjFqRH10ZEdzIn0...  
eyJleHAi0jE3NTAzNDU0MjUsImLhdCI6MTc1MDM0NTEyNSwianRpIjoizMiYmRkMDAtZTFk0C00ZjQ4LWI4MjUtMTA0MzI1ZTJmOGJmIiwiXNzIjoiaHR0cDovL2xvY2F  
saG9zdDo4MDgwL3J1YWxtcy90Zm0iLCJhdWQj0iJhY2NvdW50Iiwiic3ViIjoimWRkNmNhY2UtMDY1N100MDBkLTgwNDAtNTY5YzQzNDU20WY0IiwidHlwIjoiQmVhcmbVii  
wiYXpwIjoidGZnIiwc2lkIjoiNDlhMTE4YzAtNTQ4MC00NTBjLTg2NjgtZTAyNDBhYmE2MWI4IiwiYWNyIjoiMSIsImFsbG93ZWQtb3JpZ2lucyI6WyIqIl0sInJlYwxtX  
2FjY2VzcyI6eyJyb2xlcyI6WyJkZWZhdWx0LXJvbGVzLXRmbSISim9mZmpbmVfYWNjZXNzIiwidW1hX2F1dGhvcmI6YXRpb24iXX0sInJlc291cmNlx2FjY2VzcyI6eyJh  
Y2NvdW50Ijp7InJvbGVzIjpbIm1hbmfNzS1hY2NvdW50IiwiwbFuYwd1LWFjY291bnQtbglua3M1LCJ2aWV3LXByb2ZpbGuixX19LCJzY29wZSI6ImVtYwlsIHByb2ZpbGU  
iLCJlbWfpbF92ZXJpZm1lZCI6ZmFsc2UsIm5hbWUi0iJLZXZpbjBDYXJyYWN1ZG8gvOs0henF1ZXo1LCJwcmVmZXJyZWRfdXNlcm5hbWUi0iJrY3YyMzkilCJnaXzb19uYW  
1lIjois2V2aW4iLCJmYW1pbHlfbmFtZSI6IkNhcnJhY2VkbvBWW6F6cXveiIsImVtYwlsIjoiia2N2MjM5QGlubHVtaW51LnvhbC5lcyJ9.  
T-7eV-A0Ihop92siBLrsRL4ogWzLy8KsicRDDda0F1-TmjzlcCbLZZs1BqtXI3WZ0pKXKJJNQma2X1TFbHTflxyZdN1s3_NoQBUftTfodrRwjvTEENgjuRh3rikzmLw0d2Ms  
Ri1P_QQ2hs4Extfam0L0tQ08Vwhqf-3LncTp6FU4Qjqj2H6kvk0-KR5icoLuV11CPKv6IC-3d1uTXQgwca4ALmo3BnB3mzGWLGYK7jy0HSk5vpttXupGhhdwCLAY3WPNic  
dDCebQtvPMNh5QdAwXZtHlnnnL3wJoFtIEaF20cDtWCR99ENhSyAm3pSCVpXuv4-VI_Hkq64vStln51A",  
3   "expires_in": 300,  
4   "refresh_expires_in": 1800,  
5   "refresh_token": "eyJhbGciOiJIUzUxMiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJmYzA0NmZjZC01ZjZiltQ0ZTQtYmE1Yi1iNGFjNTI5MTA3NjAifQ.  
eyJleHAi0jE3NTAzNDY5MjUsImLhdCI6MTc1MDM0NTEyNSwianRpIjo0GRiNTI3NDItNTRh0S000TA4LWEwNTctZT1kNTAyYmYxOTgzIiwiXNzIjoiaHR0cDovL2xvY2F  
saG9zdDo4MDgwL3J1YWxtcy90Zm0iLCJhdWQj0iJodHRw0i8vbG9jYwxob3N00jgw0DAvcvmbG1zL3RmbSISinN1Yi1i6IjFkZDZjYWN1lTA2NTYtNDawZC04MDQwLTU20W  
M0MzQ1NjlmNCIsInR5cCI6I1JlZnJlc2giLCJhenAi0iJ0ZmcilCJzaWQj0ii00WExMThjMC01NDgwtQ1MGMT0DY20C1lMDI0MGFiYTYxYjgiLCJzY29wZSI6InNlcnPzY  
2VfYWNjb3VudCB1bwFpbCByb2xlcyBhY3IgcHJvZmlsZSBiYXNpYyB3ZWItb3JpZ2lucyJ9.  
twHl7_NiTmQn8HWFaN3SbRsdiipwaTBqw0QIUqCM19-2xyLbGaDP5StvzCZ4dacBeoN9lj0YZadxqcNgbK0EFg",  
6   "token_type": "Bearer",
```

# Usuario no registrado en cliente Keycloak

The screenshot shows a POST request to `http://localhost:8080/realm/tfm/protocol/openid-connect/token`. The request body is set to `x-www-form-urlencoded` and contains the following parameters:

Key	Value	Description	Bulk Edit
client_id	tfg		
client_secret	inlumine.ual.es		
grant_type	password		
username	Prueba		
password	Prueba		

The response status is **400 Bad Request** with a duration of 34 ms and a body size of 409 B. The error message is:

```
1: { "error": "unauthorized_client",
2:   "error_description": "Client not allowed for direct access grants"
3:
4: }
```

# Acceso no autorizado

The screenshot shows a POSTMAN API client interface. At the top, there is a header bar with 'GET' selected, a URL field containing 'http://localhost:5000/', and a 'Send' button. Below the header are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Scripts', and 'Settings'. The 'Params' tab is active. Under 'Query Params', there is a table with columns 'Key', 'Value', and 'Description'. A single row is present with 'Key' and 'Value' both set to 'Key' and 'Description' left empty. To the right of the table is a 'Bulk Edit' button. At the bottom of the interface, there is a navigation bar with tabs for 'Body', 'Cookies (1)', 'Headers (6)', 'Test Results', and a refresh icon. The 'Body' tab is active. On the right side of the body panel, there is a status bar showing '401 UNAUTHORIZED', '13 ms', '214 B', and a globe icon. Below the status bar are buttons for 'Copy', 'Save Response', and other options. The main body area displays a JSON response:

```
1 {  
2   "error": "Unauthorized"  
3 }
```

# Token JWT caducado

The screenshot shows a POSTMAN API request configuration and its resulting response.

**Request Details:**

- Method: **DELETE**
- URL: **http://localhost:5001/deleteModel**
- Auth Type: **Bearer Token**
- Token: **{}{{bearerToken}}**

**Response Details:**

- Status: **401 UNAUTHORIZED**
- Time: **7 ms**
- Size: **239 B**
- Headers:
  - Content-Type: application/json
  - Date: Mon, 12 Dec 2022 10:45:47 GMT
  - Server: Kestrel
- Save Response

**Body:**

```
1 {  
2   "error": "Unauthorized",  
3   "reason": "token_expired"  
4 }
```

# Índice

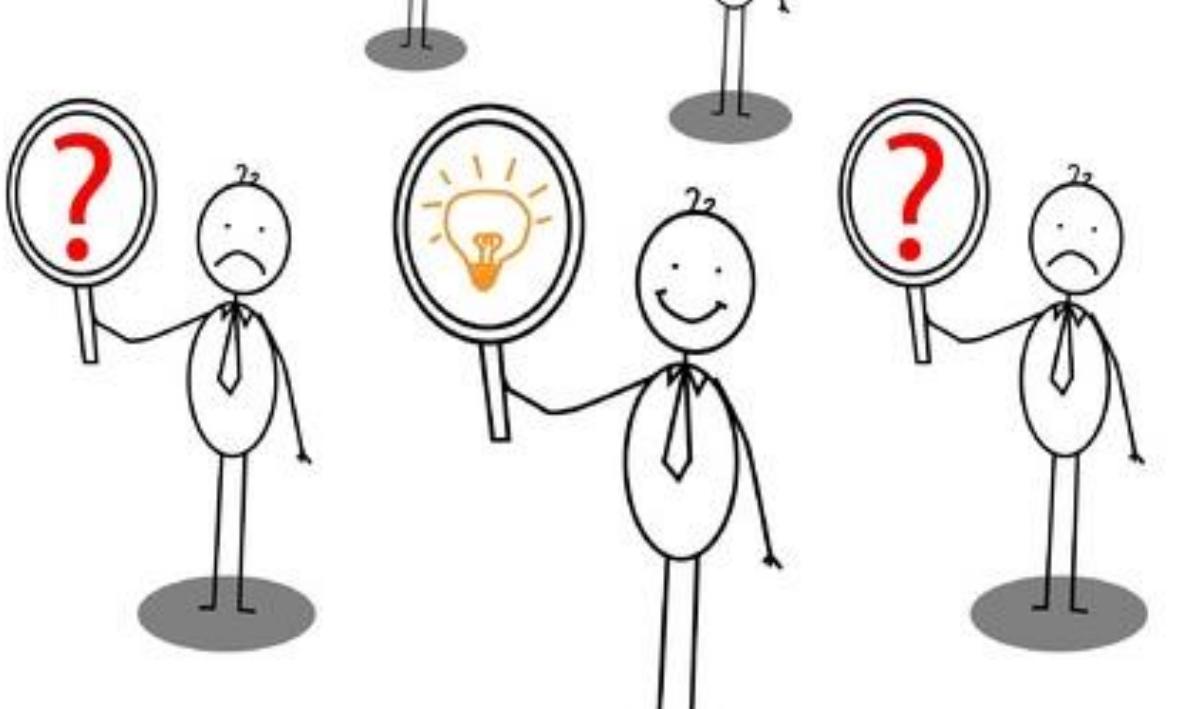
1. Introducción
2. Motivación
3. Planificación
4. Objetivos
5. Estado del arte
6. Herramientas y tecnologías
7. Descripción de la solución
8. Pruebas y resultados
9. Problemas encontrados
10. Conclusiones y trabajo futuro

# 8. Pruebas y resultados

1. Pruebas de login OIDC desde Vault y aplicaciones ejemplo.
1. Pruebas de acceso a endpoints protegidos y manejo de errores:
  - I. Acceso con token válido: éxito.
  - II. Acceso sin permisos o con token caducado: denegado.

# Índice

1. Introducción
2. Motivación
3. Planificación
4. Objetivos
5. Estado del arte
6. Herramientas y tecnologías
7. Descripción de la solución
8. Pruebas y resultados
9. Problemas encontrados
10. Conclusiones y trabajo futuro



## 9. Problemas encontrados

- ◆ Problemas detectados en la solución:
  1. Servicios sin replicación y timeouts.
  2. Vault sin auto-unseal tras reinicio.
  3. Pérdida de datos por fallos en volúmenes locales.
  
- ◆ Plan de mitigación problemas detectados:
  1. Despliegue en clústeres con balanceo de carga.
  2. Copias de seguridad y replicación de bases de datos.
  3. Sondas de liveness/readiness y alertas automáticas.

# Índice

1. Introducción
2. Motivación
3. Planificación
4. Objetivos
5. Estado del arte
6. Herramientas y tecnologías
7. Descripción de la solución
8. Pruebas y resultados
9. Problemas encontrados
10. Conclusiones y trabajo futuro

# 10. Conclusiones y trabajo futuro

## Conclusiones

- Externalizar autenticación y gestión de secretos reduce riesgos y complejidad del desarrollo.
- Keycloak y Vault permiten una integración segura y escalable para aplicaciones existentes.
- IaC garantiza reproducibilidad y facilidad de mantenimiento.

## Trabajo Futuro

- Despliegue en clústeres Kubernetes para alta disponibilidad.
- Integración de generación dinámica de secretos y certificados.
- Monitorización avanzada con Prometheus, Grafana y centralización de logs.

# Bibliografía

- [1] F. J. Corbató y V. A. Vyssotsky, "Introduction and overview of the Multics system", Proceedings of the Fall Joint Computer Conference, pp. 185-196, 1965.
- [2] T. Hunt, "Passwords are broken, and nobody seems to care", IEEE Security & Privacy, vol. 15, no. 5, pp. 70-74, 2017.
- [3] Verizon, "2016 Data Breach Investigations Report", Verizon Enterprise, 2016.
- [4] A. O. Udo, "Phishing: A growing challenge for internet users", Journal of Information Security and Applications, vol. 21, pp. 1-9, 2015.
- [5] W. Yeong, T. Howes, y S. Kille, "Lightweight Directory Access Protocol", RFC 1777, 1995.
- [6] S. Furnell, "Tokens and Beyond: Multi-Factor Authentication for the Masses", *Computer Fraud & Security*, vol. 2005, no. 8, pp. 12-16, 2005.
- [7] National Institute of Standards and Technology, "Electronic Authentication Guideline", NIST Special Publication 800-63-2, 2013.
- [8] P. Madsen, "A Guide to Understanding SAML", *Sun Microsystems*, 2003.
- [9] OASIS, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard, 2005.
- [10] D. R. Kuhn, "An overview of single sign-on technology", *Proceedings of the 24th NIST-NCSC National Information Systems Security Conference*, pp. 27-31, 2001.
- [11] A. K. Das, N. Kumar, y J. J. P. C. Rodrigues, "Authentication protocols and schemes for smart mobile devices: security vulnerabilities, attacks, countermeasures, and open issues", *Telecommunication Systems*, vol. 67, no. 2, pp. 249-274, 2018.
- [12] E. Hammer-Lahav, "The OAuth 1.0 Protocol", RFC 5849, 2010.
- [13] D. Hardt, "The OAuth 2.0 Authorization Framework", RFC 6749, 2012.
- [14] N. Sakimura et al., "OpenID Connect Core 1.0", The OpenID Foundation, 2014.
- [15] M. Jones, "JSON Web Token (JWT)", RFC 7519, 2015.
- [16] Gartner, "Magic Quadrant for Identity as a Service, Worldwide", Gartner Research, 2019.
- [17] Red Hat, "Keycloak Documentation". Disponible en <https://www.keycloak.org/documentation>. Consultado el 10 de junio de 2025.
- [18] Red Hat, "Server Developer Guide: Service Provider Interfaces (SPI)", Keycloak Documentation, 2023. Disponible en [https://www.keycloak.org/docs/latest/server\\_development/#\\_providers](https://www.keycloak.org/docs/latest/server_development/#_providers). Consultado el 10 de junio de 2025.
- [19] HashiCorp, "Vault: Secrets Management", Vault by HashiCorp, 2023. Disponible en <https://www.vaultproject.io/docs>. Consultado el 10 de junio de 2025.
- [20] K. Jackson, "Managing Secrets in Microservices with Vault", ACM Queue, vol. 18, no. 5, pp. 34-47, 2020.

# Bibliografía

- [21] M. Merkow y J. Breithaupt, "Virtualization and Containers: A Guide to Software Isolation and Security", 2.<sup>a</sup> ed. Boston, MA, EE. UU.: Pearson, 2021.
- [22] Docker Inc., "Docker Overview", Docker Documentation, 2023. Disponible en <https://docs.docker.com/get-started/overview>. Consultado el 10 de junio de 2025.
- [23] Docker Inc., "Best practices for writing Dockerfiles", Docker Documentation, 2023. Disponible en [https://docs.docker.com/develop/develop-images/dockerfile\\_best-practices](https://docs.docker.com/develop/develop-images/dockerfile_best-practices). Consultado el 10 de junio de 2025.
- [24] The Kubernetes Authors, "Kubernetes Documentation", Kubernetes.io, 2023. Disponible en <https://kubernetes.io/docs/home>. Consultado el 10 de junio de 2025.
- [25] The Kubernetes Authors, "Pod Security Standards", Kubernetes Documentation, 2023. Disponible en <https://kubernetes.io/docs/concepts/security/pod-security-standards>. Consultado el 10 de junio de 2025.
- [26] G. Kim, J. Humble, P. Debois y J. Willis, "The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations", 2.<sup>a</sup> ed. Portland, OR, EE. UU.: IT Revolution Press, 2021.
- [27] N. Zubair, "DevSecOps: Integrating Security into DevOps", IEEE Software, vol. 37, no. 3, pp. 20–27, 2020.
- [28] OWASP, "Testing Guide", OWASP Foundation, 2023. Disponible en <https://owasp.org/www-project-web-security-testing-guide>. Consultado el 10 de junio de 2025.
- [29] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg y I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", Future Generation Computer Systems, vol. 25, núm. 6, pp. 599–616, 2009.
- [30] P. Mell y T. Grance, "The NIST Definition of Cloud Computing", NIST Special Publication 800-145, Gaithersburg, MD, USA, 2011.
- [31] T. Erl, R. Puttini y Z. Mahmood, "Cloud Computing: Concepts, Technology & Architecture". Upper Saddle River, NJ, USA: Prentice Hall, 2013.
- [32] M. Armbrust et al., "A View of Cloud Computing", Communications of the ACM, vol. 53, núm. 4, pp. 50–58, 2010.
- [33] M. Cusumano, A. Gawer y D. B. Yoffie, "The Business of Platforms: Strategy in the Age of Digital Competition, Innovation, and Power". New York, NY, USA: Harper Business, 2019.
- [34] K. Morris, "Infrastructure as Code: Managing Servers in the Cloud". Sebastopol, CA, USA: O'Reilly Media, 2016.
- [35] P. M. Fitts, S. L. Atkinson y J. Watson, "Infrastructure as Code: Principles and Practices of Immutable Infrastructure," en Proceedings of the 2019 IEEE Conference on Software Engineering (ICSE), Montreal, Canada, 2019, pp. 123–130.
- [36] M. Duffy, "Infrastructure as Code with Terraform: Provisioning Cloud Infrastructure in a Modern and Reproducible Way". Sebastopol, CA, USA: O'Reilly Media, 2020.
- [37] Pulumi, "Pulumi Documentation". Disponible en <https://www.pulumi.com/docs>. Consultado el 10 de junio de 2025.
- [38] Red Hat, "Ansible Documentation". Disponible en <https://docs.ansible.com>. Consultado el 10 de junio de 2025.
- [39] Progress Software, "Chef Documentation". Disponible en <https://docs.chef.io>. Consultado el 10 de junio de 2025.
- [40] Puppet, "Puppet Documentation". Disponible en <https://puppet.com/docs>. Consultado el 10 de junio de 2025.

# ¿Preguntas?



MUCHAS GRACIAS POR SU ATENCIÓN



UNIVERSIDAD  
DE ALMERÍA