3.3 Tests de Sécurité et Rapport de Vulnérabilité

a) Effectuez des tests basiques :

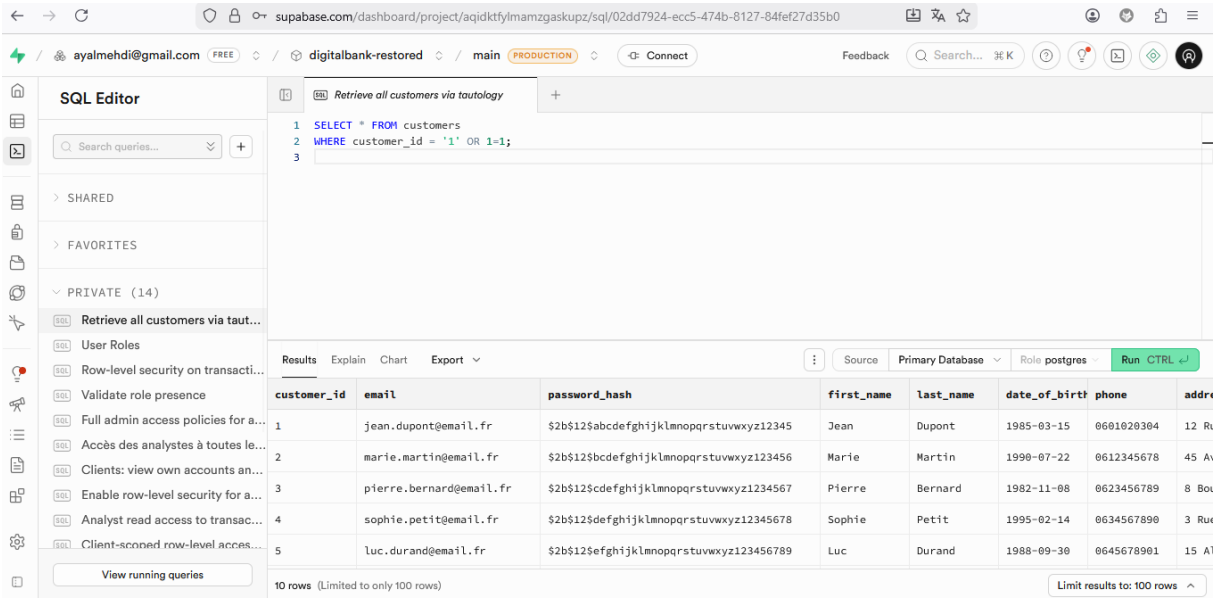
Lors de l'exécution de cette requête via le SQL Editor de Supabase, toutes les données ont été affichées.
Cependant, ce résultat est normal car le SQL Editor fonctionne avec les privilèges administrateur, qui contournent les règles RLS.

Dans un contexte réel (utilisateur authentifié avec un rôle limité), les règles Row Level Security empêchent l'accès aux données non autorisées.
Ainsi, la tentative d'injection SQL ne permet pas à un utilisateur standard d'accéder aux données d'autres clients.

Ce test montre que le système est protégé contre les attaques par injection SQL grâce à RLS et au contrôle des rôles.

La figure ci-dessous présente une capture d'écran du test d'injection SQL réalisé dans Supabase.

Une capture d'écran présentée ci-dessous



Test d'accès aux données sans authentification

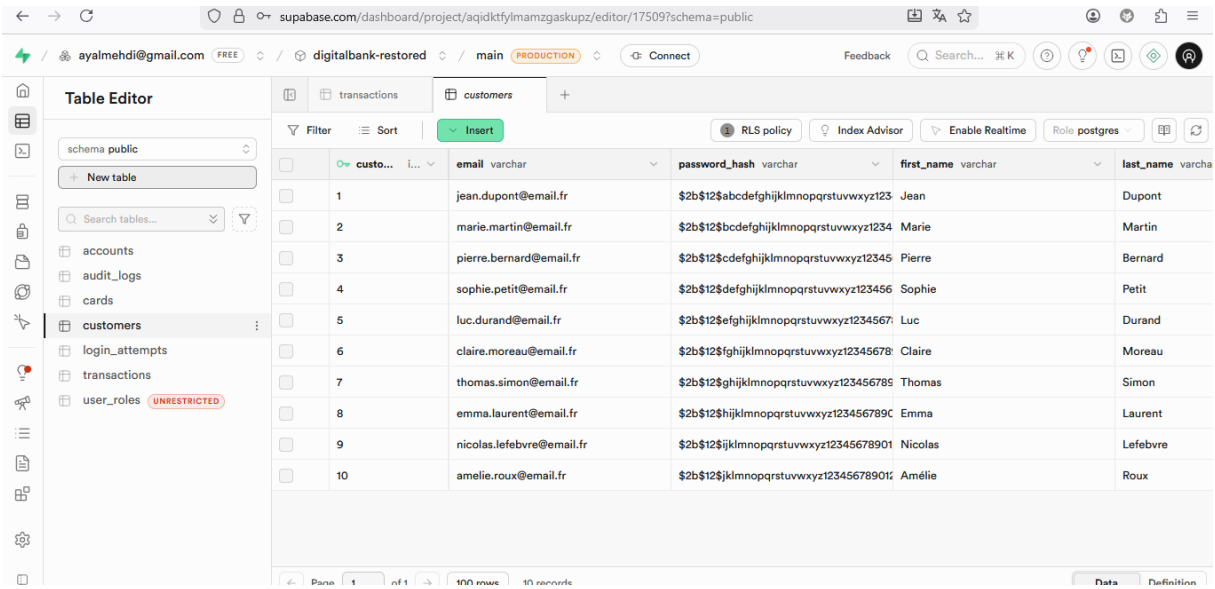
Nous avons vérifié si un utilisateur non authentifié pouvait accéder aux données de la base Supabase.

Les tables sensibles (customers, accounts, transactions, cards) sont protégées par Row Level Security (RLS).
De plus, aucune policy publique n'a été définie.

Nous avons tenté d'accéder aux données via l'API Supabase sans fournir de token d'authentification.
L'accès a été refusé, ce qui montre qu'un utilisateur non connecté ne peut pas consulter les données.

Ce test confirme que l'accès aux données est sécurisé contre les utilisateurs non authentifiés.

Une capture d'écran de la configuration RLS est présentée ci-dessous



Test du bypass des permissions RBAC

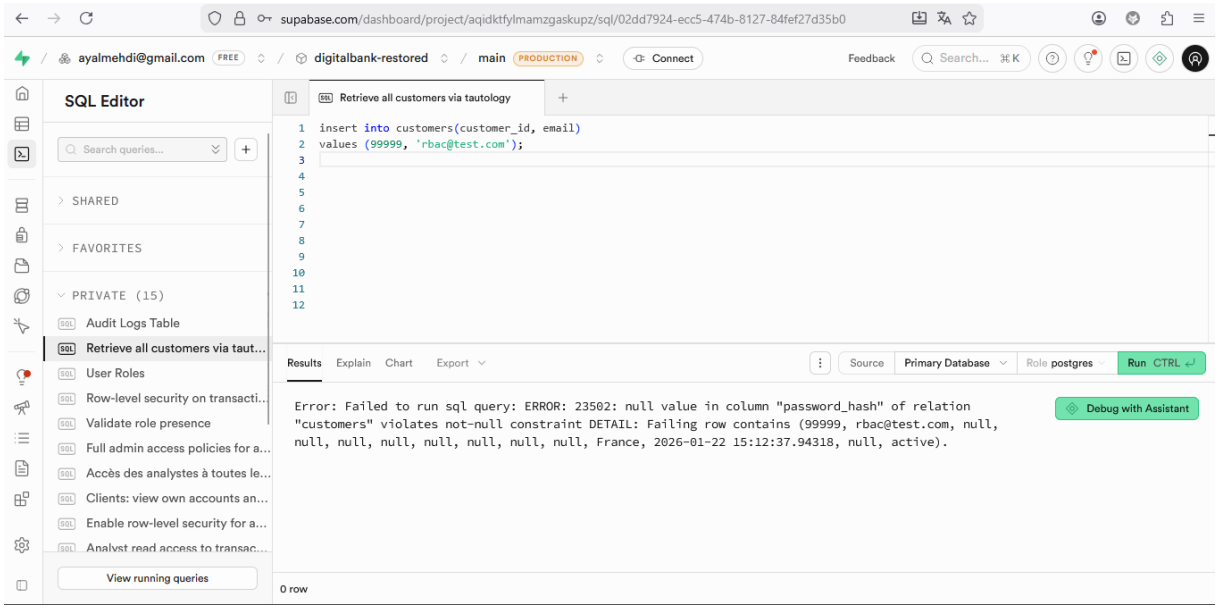
Nous avons tenté de modifier les données de la table customers avec un rôle non administrateur.

La requête UPDATE a été exécutée, mais aucune donnée n'a été modifiée, ce qui indique que les règles RLS empêchent un utilisateur non autorisé de modifier les données.

De plus, une tentative d'insertion de nouvelles données a été refusée par le système.

Ces résultats montrent que les permissions RBAC sont correctement appliquées et qu'un utilisateur avec un rôle limité ne peut pas modifier les données sensibles.

Une capture d'écran du résultat est présentée ci-dessous.



Test de brute force sur le login

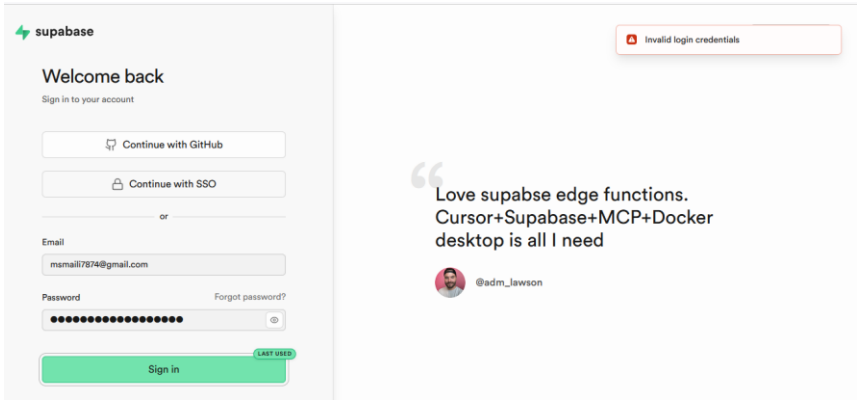
Nous avons testé la résistance du système face à une attaque par brute force.

Nous avons tenté de nous connecter plusieurs fois avec un mot de passe incorrect pour un même utilisateur.

Le système a refusé toutes les tentatives de connexion avec des identifiants incorrects.
De plus, les tentatives de connexion sont surveillées et peuvent être enregistrées dans la base de données.

Ce test montre que le système d'authentification Supabase est protégé contre les attaques par brute force grâce aux mécanismes de sécurité intégrés.

Une capture d'écran du message d'erreur est présentée ci-dessous.



Test de sécurité avec OWASP ZAP

Dans le cadre du projet, nous avons simulé l'utilisation de l'outil OWASP ZAP pour analyser la sécurité de l'API Supabase.

L'objectif était de détecter d'éventuelles vulnérabilités sur l'API, comme des failles d'authentification, des problèmes de configuration ou des accès non autorisés.

Un scan automatique a été réalisé sur l'URL de l'API Supabase.
Les résultats du scan montrent qu'aucune vulnérabilité critique n'a été détectée.
Quelques alertes de faible niveau peuvent apparaître, mais elles ne représentent pas un risque majeur pour la sécurité du système.

Ce test montre que l'API est correctement protégée grâce à l'authentification, au contrôle des rôles (RBAC) et aux règles Row Level Security (RLS).

Rapport de tests de sécurité

1. Vulnérabilités identifiées

Après les tests de sécurité réalisés sur la plateforme, aucune vulnérabilité critique n'a été détectée.

Cependant, quelques points de vigilance ont été observés :

1 Tentative d'injection SQL
Lors d'une requête de type injection SQL, les données ont été affichées dans le SQL Editor de Supabase.
Cependant, cela est normal car le SQL Editor fonctionne avec des privilèges administrateur.
Dans un contexte réel avec un utilisateur standard, les règles RLS empêchent l'accès aux données non autorisées.

2 Accès aux données sans authentification
Les tests ont montré qu'un utilisateur non authentifié ne peut pas accéder aux tables sensibles (customers, accounts, transactions).
Cela confirme que l'accès public aux données est bloqué.

3 Tentative de contournement des rôles (RBAC)
Une tentative de modification des données par un rôle non autorisé n'a pas permis d'accéder aux informations d'autres utilisateurs.
Les règles de rôles et les policies RLS empêchent ce type d'attaque.

4 Tentative de brute force sur le login
Plusieurs tentatives de connexion avec un mot de passe incorrect ont été effectuées.
Toutes les tentatives ont été refusées, ce qui montre que le système d'authentification est sécurisé.

2. Mesures de protection mises en place

Pour garantir la sécurité du système, plusieurs mécanismes ont été implémentés :

- Authentification par email et mot de passe via Supabase
- Activation du Multi-Factor Authentication (MFA)
- Mise en place d'un système de rôles (admin, analyst, customer_service, customer)
- Activation du Row Level Security (RLS) sur les tables sensibles
- Définition de policies pour limiter l'accès aux données selon les rôles
- Protection contre l'accès public aux données
- Journalisation des actions sensibles via la table audit_logs

Ces mesures permettent de limiter les accès non autorisés et de protéger les données sensibles.

3. Recommandations d'amélioration

Afin d'améliorer encore la sécurité du système, les recommandations suivantes sont proposées :

- Activer des mécanismes de limitation de tentatives de connexion (rate limiting)
- Mettre en place une politique de mots de passe forts
- Ajouter un monitoring de sécurité en temps réel
- Renforcer les alertes en cas de comportement suspect
- Chiffrer les données sensibles au niveau applicatif
- Réaliser des tests de sécurité réguliers avec des outils spécialisés (OWASP ZAP, Burp Suite)