**Budapesti Műszaki Szakképzési Centrum**

**Neumann János Informatikai Technikum**

**VIZSGAREMEK**

**BiteMe Burger app, Éttermi webalkalmazás**

**06135013 Szoftverbiztonsági szakember**

**Készítette: Rimóczi Dániel**

**Budapest**

**2025**

**Eredetiség nyilatkozat**

Alulírott Rimóczi Dániel a 06135013 Szoftverbiztonsági szakember képzés résztvevője kijelentem, hogy a vizsgaremek saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem.

Budapest, 2025. 05. 15.

.................................................

Rimóczi Dániel

Tartalom

[Az alkalmazás funkciói 4](#_Toc199074591)

[Telepítési útmutató 5](#_Toc199074592)

[Technikai megvalósítás 6](#_Toc199074593)

[Alkalmazás programozás 6](#_Toc199074594)

[Infrastruktúra kialakítása 6](#_Toc199074595)

[Adatbázis struktúra: 8](#_Toc199074596)

[Tesztek és sérülékenység vizsgálati eredmények: 9](#_Toc199074597)

[Api végpont tesztek: 9](#_Toc199074598)

[Api dokumentáció: 9](#_Toc199074599)

[Trivy-vel image-eken végezett tesztelési eredmények: 10](#_Toc199074600)

[nginx:1.27.5-alpine-slim: 10](#_Toc199074601)

[datalust/seq:latest: 11](#_Toc199074602)

[mysql:latest: 12](#_Toc199074603)

[Zap report: 13](#_Toc199074604)

[Biztonsági funkciók illetve azok hiányai 14](#_Toc199074605)

[Az alkalmazás biztonsági funciói 14](#_Toc199074606)

[Az alkalmazás hiányossága és fejlesztési potenciáljai : 14](#_Toc199074607)

# Az alkalmazás funkciói

Felhasználó:

* Regisztrálni/ bejelentkezni tud
* tud berakni a kosarába hamburgereket
* meg tudja nézni a kosarát
* tudja módosítani / törölni a kosarát
* tud felvenni / törölni címeket
* tud leadni rendelést
* tudja listázni a rendeléseit státusz alapján

Van alapból két felhasználó regisztrálva a rendszeren és van 11 burger felvéve a menüre

Admin:

- Email : admin@example.com

- Pw : 123321

User:

- Email : joska@example.com

- Pw : 123321

Az adminok ha elkészült volna felület (backenden kész vannak a funkciók)

* menü kezelése: módosítás, törlés, hozzáadás
* rendelések kezelése

# Telepítési útmutató

Huzzuk le a a repot github-ról

<https://github.com/greengamerhu/Vizsgaremek>

Ahhoz hogy működőn az alkalmazás kelleni fog egy Docker mivel az alkalmazás teljesen konténerizálva van

<https://www.docker.com/>

Következő lépésként ha csak a Developer környezetet akarjuk elindítani akkor elég a projektnek a gyökérmappájában nyitni egy terminált és begépelni a ezt:

az env file-okon nem kell állítgatni

docker compose up –build -d

<http://localhost:3000> – frontend

<http://localhost:3001> - backend

Leállításhoz:

docker compose down

De ha a „production ready” környezetet szeretnénk elindítani akkor viszont a következőket kell beírni a terminálba:

(itt valamiért a docker nem igazán akarta beadni a build containernek az env file-t szóval létrehoztam egyet a frontendben is és úgy már müködik)

docker compose -f docker-compose.prod.yml --env-file .env.prod up --build -d

<http://localhost> – frontend

<http://localhost>/api/ - backend

<http://localhost:5341> – log szerver

docker compose -f docker-compose.prod.yml down

# Technikai megvalósítás

## Alkalmazás programozás

Backend:

Nestjs-ben készült a teljes backend része az alkalmazásnak nagyon sok minden megtalálható benne alapból és nagyon sok package-re van támogatás

Amiket én használtam:

* TypeORM: Ami nekem abban segített hogy teljesen megszabadultam az sql lekérdezésektől és egyben megkönnyítette a programozás részét is mivel az adatbázisban a táblákat ő hozza létre indításkor, a kódban is egyszerűen objektumként tudok rá hivatkozni
* Pino: Loggoláshoz/naplózáshoz használtam
* Throttler: A ddos támadások elkerülése érdekében
* Swagger Api dokumentáció
* képek kiszolgálása beépített csomaggal történik

Frontend:

React + Typscript kombót azért válaszottatam mert viszonylag egyszerű benne jól kinéző dolgokat alkotni kevés idő alatt:

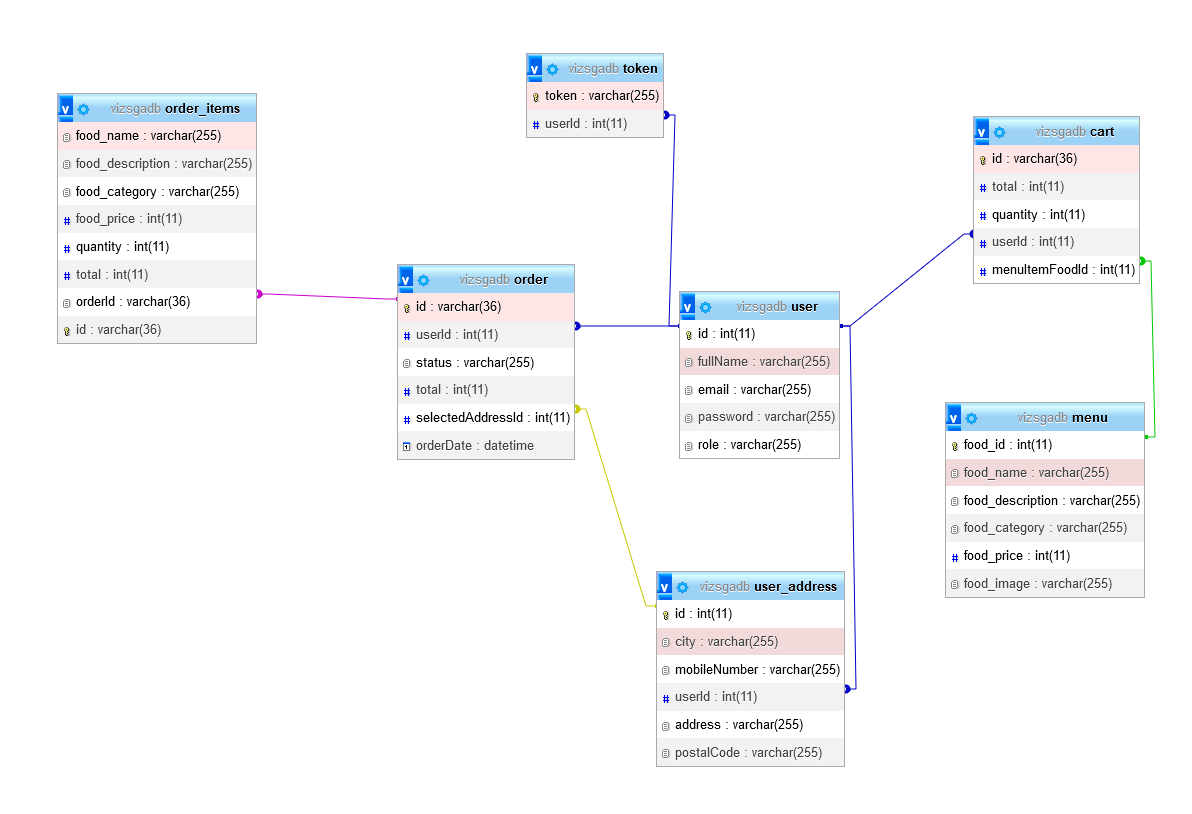
* React/MUI: Különböző kész reszponzív komponens könyvtár
* Axios: Az api kérések intézésére mert egyszerűbb is használni mint a sima fetch-et és még biztonságosabb is
* react-toastify: a különböző visszajelzések megjelenítésére használtam szervertől kapott üzenetek, egyéb visszajelzésre szoruló dolgok
* bootstrap: hasonló mint az MUI volt ahol inkább ebből a libary-ből választottam komponenseket

## Infrastruktúra kialakítása

Mivel az egész alkalmazás dockerizálva van ezért viszonylag egyszerűen le tudom írni a szerkezetét:

* nginx: ez szolgálja ki a frontend-et és rak egy proxy-t a backend és a frontend közé ami annyiban segít hogy nem közvetlenűl a localhostra küldi a frontend a kéréseket hanem átírányítódik /api/ végponton és onnan küldödik a localhostra tehát:
  + Statikusan kiszolgálja a buildelt frontend alkalmazást
  + Továbbítja az /api/ végpontokra irányuló kéréseket a NestJS backendhez
  + Biztonsági fejlécekkel és konfigurációval védi a rendszert
* nestjs: Egy node konténeren belül fut
* mysql: sok vitát láttam hogy érdemes e egyáltalán konténerizálni az adatbázist stabilitás / biztonsági kérdések is feljöttek de úgy gondoltam hogy ennek a projektnek a scope-ja alapján hogy konténerizálom
* mysql\_backup: ez meg szerintem elég leíró név de fél óránként csinál egy backup dumpot az adatbázisról a gyökér mappában
* seq: egy log/napló gyűjtő rendszer

## Adatbázis struktúra:



# Tesztek és sérülékenység vizsgálati eredmények:

## Api végpont tesztek:

A backendnek a mappáján belül megtalálhatóak:

BiteMe\_Burger\_Végpont\_tesztek\_final.json ezen a file néven

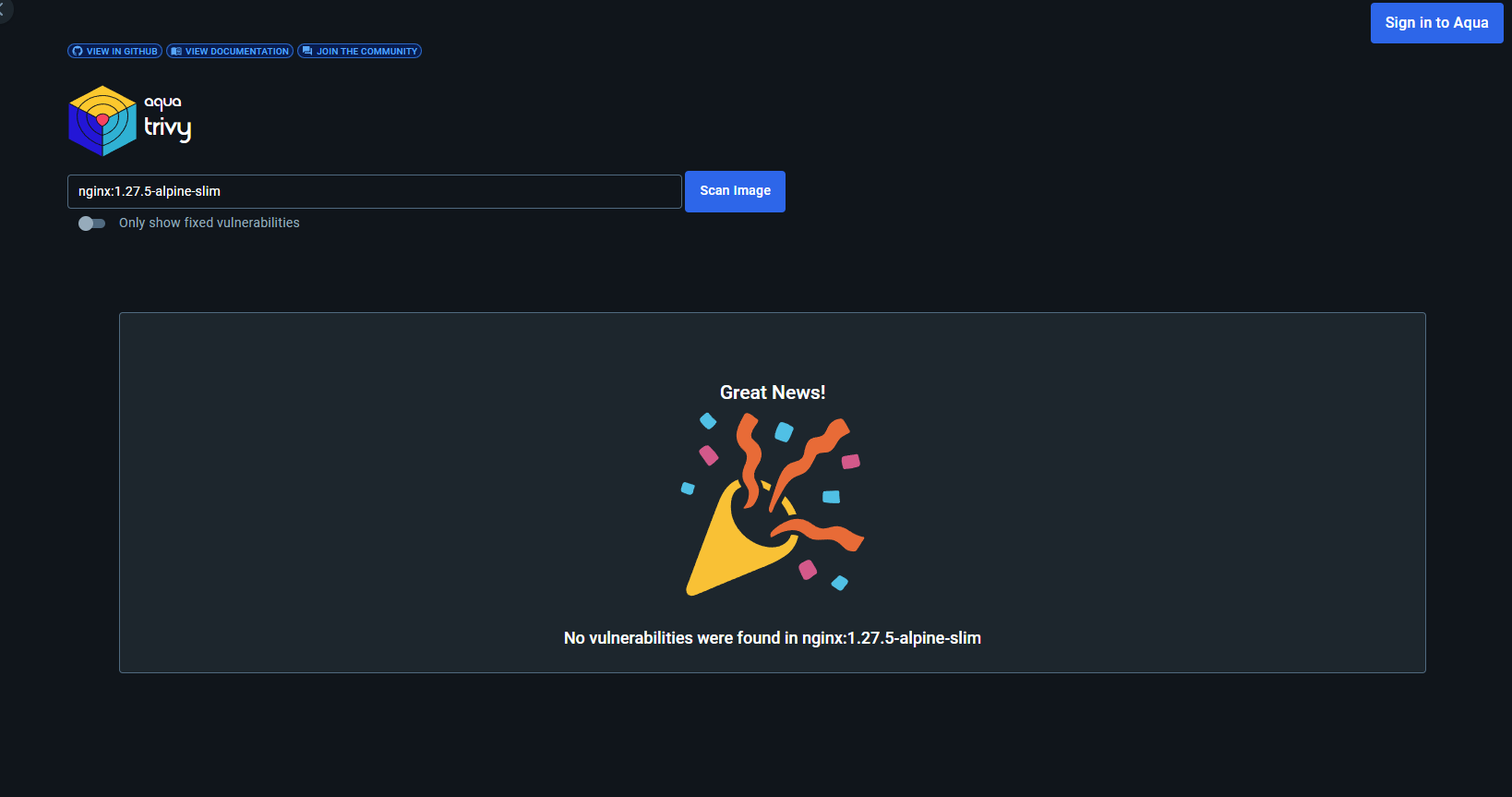
## Api dokumentáció:

dev környezetben: localhost:3001/docs

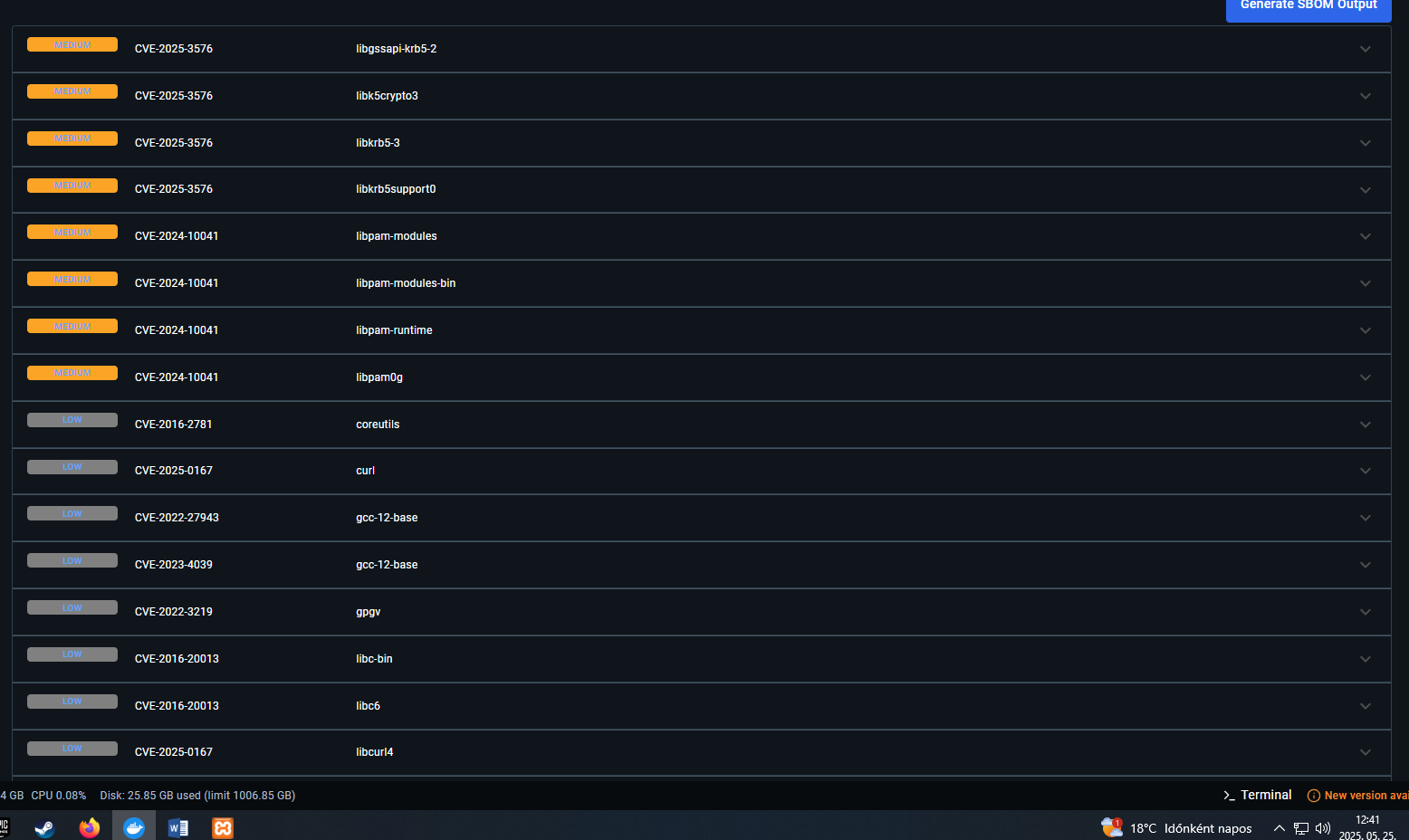
prod-ban (tudom nem kéne prodban elérhetőnek lennie) localhost/api/docs

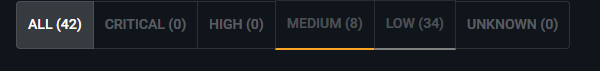
## Trivy-vel image-eken végezett tesztelési eredmények:

### nginx:1.27.5-alpine-slim:



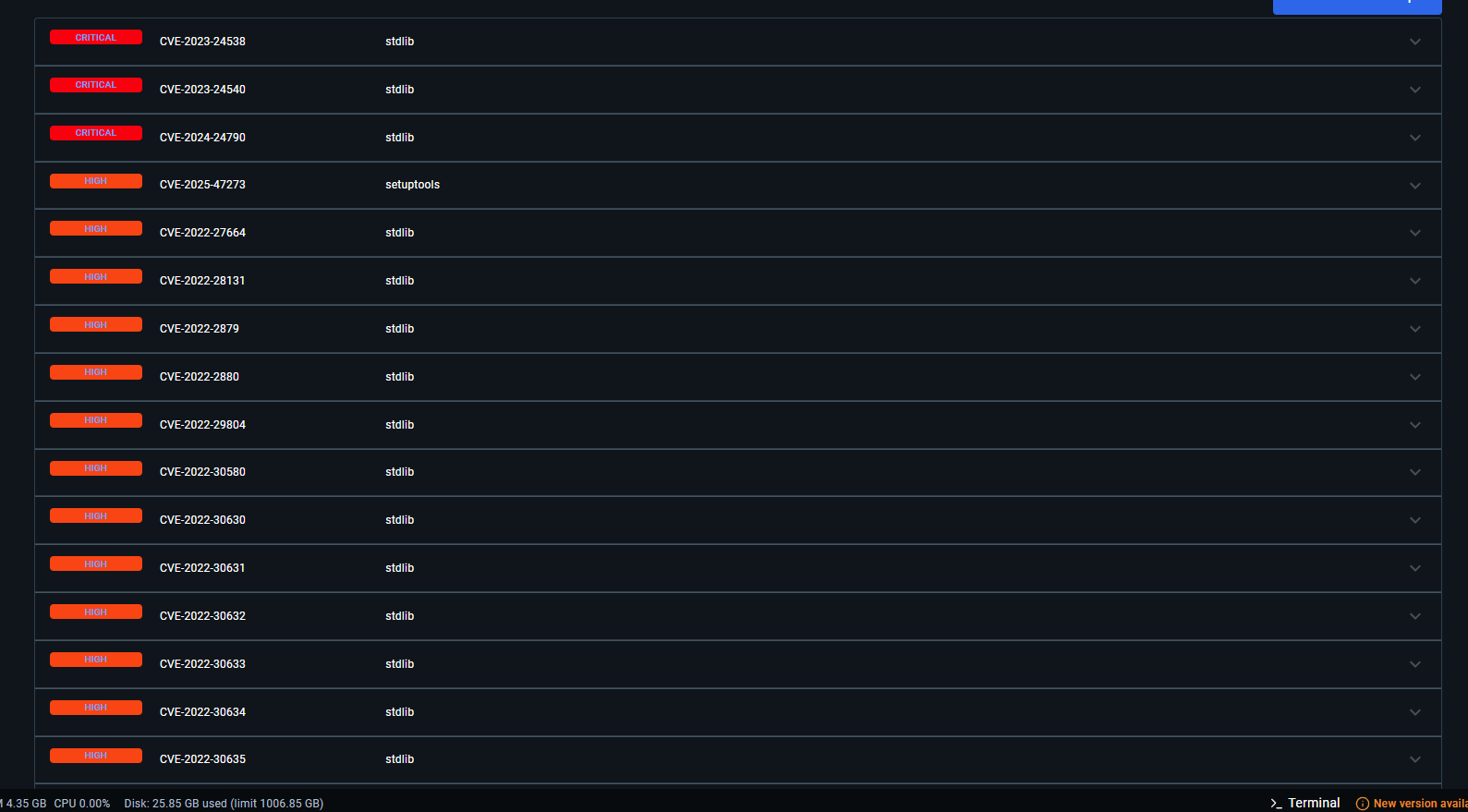
### datalust/seq:latest:

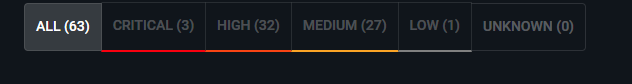




Ezekre még nincsenek elérhető frissítések még jelenleg is dolgoznak rajtuk és mivel kis besorolásúak ezért elfogadtam a kockázatot

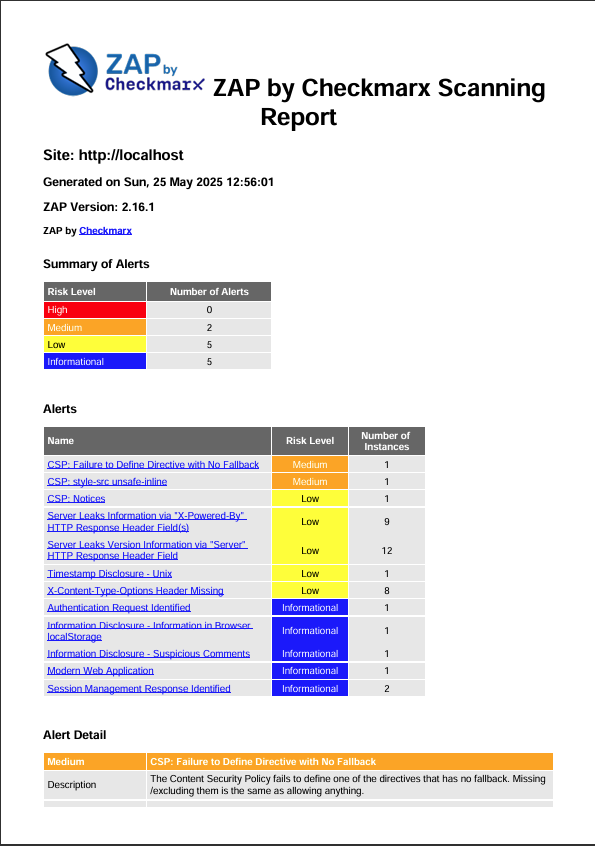
### mysql:latest:





Itt rengeteg hiba van de mivel kívülről sehogy sem elérhető mivel a dockernek a belső hálózatáról működik ezért úgy ítéltem meg hogy ezeknél is elfogadom a kockázatot

## Zap report:



Többnyire csp probléma van, amit őszintén megvallva annyira sokat szenvedtem vele hogy működőn hogy inkább elengedtem

# Biztonsági funkciók illetve azok hiányai

## Az alkalmazás biztonsági funciói

* rate limiting (1 perc alatt száz kérést lehet intézni egy helyről egy végpontra)
* naplózás megvalósítása, felhasználói/admin tevékenység naplózása, jól átláthatósága
* jelszavak hash-elése
* Jogosultság kezelés a backend-en
* az adatbázis és a backend izolációja
* Az adatok elérhetőségének részleges megvalósítása a backup készítéssel
* reverse proxy
* A hibaüzenetek backenden történő feldolgozása ( pl: rossz email cím)

## Az alkalmazás hiányossága és fejlesztési potenciáljai :

* egy egyszerű token alapú bejelentkezési rendszert használok elég elavult és nem is biztonságos: Megoldás:
  + Akár kiszervezni egy AD/B2C Ms, firebase, Auth0 rendszerbe (legjobb megoldás mert akkor nem nálunk vannak a felhasználói adatok)
  + szimplán kiépíteni egy accestoken/refreshtoken rendszert
  + http-only Cookie-k használata
* naplózási felület nincsen túlságosan jól beállítva hiányzik róla egy login rendszer, a kategorizálása hiánya, illetve az se segít, hogy a nagy nyilvánosságnak elérhető megoldás:
  + Kiszervezni szintén valamelyik nagy cloud szolgáltatónak
  + beállítani egy erős user + jelszó védelmet
  + Részletes kategóriákat hozzáadni pl request kimenetel alapján szűrés 401,404,400 stb
* Adatbázisról a jelszó hiányzik a megoldás vagy beállítani vagy kiszervezni úgy ahogy már fent is említettem cloudba
* backup automatikus betöltése ha esetleg leállna vagy korruptálódnának az adatok mert most jelenleg csak kézzel történhetne ez meg
* Admin felület teljes hiánya, a backend fel van készítve erre(api végpontok készek) de viszont a frontendhez már nem volt elég jó az időbeosztásom hogy megtörténjen meg elvittek idő különböző bugok illetve fejlesztési nehézségek
* csp headerek beállítása
* frontend input validáció hiánya minden a backenden történik de lehetne levenni terhet és levédeni támadási lehetőségeket
* https protokol hiánya
* CI (Continuous Integration) hiánya amivel folyamatosan lehetett volna tesztelni az alkalmazást a sérülékenységeket és clean code meglétét
* a Unit illetve a webfelületi tesztek hiánya
* repoban .env.example-öknek kéne fent lenni de az alkalmazás nagyon kényes az env beállításokra ezért feltöltöttem őket
* stressz teszt hiánya
  + locust-al vagy Ddosify

És még biztosan vannak olyan hibák amikre tudáshiány miatt nem látok rá….