🔧 Plágium Ellenőrző Rendszer - Technikai Áttekintés

# ROBOT FRAMEWORK PLÁGIUM ELLENŐRZŐ - TECHNIKAI ÁTTEKINTÉS

## RENDSZER CÉLJA

Automatizált DOCX dokumentum plágium ellenőrzés hash-alapú algoritmussal, Excel jelentéskészítéssel és automatikus email értesítéssel.

## ARCHITEKTÚRA

### Robot Framework Modulok

* PLG-00-main.robot: Főfolyamat vezérlő
* PLG-02-read\_docx.robot: DOCX beolvasás és hash generálás
* PLG-03-write-excel.robot: Excel export funkció

### Python Backend

* DocxReader.py: DOCX tartalom kinyerés
* send\_email.py: Outlook COM email automatizálás
* excelexportsimple.py: openpyxl alapú Excel generálás
* plagium\_config.py: Konfigurációs osztály

### Adatbázis Réteg

* SQLite: Helyi adatbázis (test\_database.db)
* 3 tábla: redundancia, hashCodes, repeat
* Relációs kapcsolatok: Foreign key integritás

## PLÁGIUM ALGORITMUS

### 1️⃣ Hash Generálás

import hashlib  
hash\_value = hashlib.sha256(line\_content.encode('utf-8')).hexdigest()

### 2️⃣ Összehasonlítás

SELECT file\_name, line\_content FROM hashCodes  
WHERE hash\_value = ? AND file\_name != ?

### 3️⃣ Kategorizálás

* 🟢 Rendben: < 300 karakter redundancia
* 🟡 Gyanús: 300-1200 karakter redundancia
* 🔴 Másolt: > 1200 karakter redundancia

## EMAIL AUTOMATIZÁLÁS

### Outlook COM Integráció

outlook = win32com.client.Dispatch("Outlook.Application")  
mail = outlook.CreateItem(0)  
mail.Send() # Automatikus küldés

### Hibakezelés

* ✅ 3 próbálkozás exponenciális várakozással
* ✅ Piszkozat mentés fallback megoldás
* ✅ Részletes hibanaplózás

## KIMENETEK

### Excel Jelentés

* Formátum: .xlsx (openpyxl)
* Tartalom: Státusz, fájlnév, redundancia metrikák
* Stílus: Színkódolt státusz, formázott táblázat

### Email Template

* HTML formátum: Modern, reszponzív design
* Melléklet: Excel fájl automatikus csatolás
* Tartalom: Összesítő statisztikák + részletes lista

## KONFIGURÁLÁS

### Plagium.config

email=lovasz.otto@clarity.hu  
input\_folder=d:\tmp  
output\_folder=d:\tmp  
email\_subject=Plagium Ellenorzes - Eredmenyek  
excel\_prefix=plagium\_eredmenyek

### Robot Framework változók

${SQLITE\_DB\_FILE} test\_database.db  
${PYTHON\_EXEC} rf\_env/Scripts/python.exe  
${DOCUMENT\_PATH} ${CONFIG\_INPUT\_FOLDER}

## FUTTATÁS

### Egyszerű futtatás

rf\_env\Scripts\robot.exe PLG-00-main.robot

### Batch futtatás

FOR ${docx\_file} IN @{docx\_files}  
 Fájladatok Rögzítése Redundancia Táblába ${docx\_file}  
 DOCX Beolvasás Teszt  
END

## TELJESÍTMÉNY

### Feldolgozási sebesség

* Kis fájl (<1MB): ~10-15 másodperc
* Közepes fájl (1-5MB): ~30-60 másodperc
* Nagy fájl (>5MB): ~2-5 perc

### Memória használat

* Alapfolyamat: ~50-100 MB
* Nagy dokumentum: ~200-500 MB
* Batch feldolgozás: Lineáris skálázódás

## BIZTONSÁG

### Adatvédelem

* ✅ Helyi adatkezelés (nincs cloud)
* ✅ SHA-256 kriptográfiai hash
* ✅ Windows biztonsági modell

### Hibakezelés

* ✅ Try-except blokkok minden kritikus ponton
* ✅ Részletes error logging
* ✅ Graceful degradation

## FÜGGŐSÉGEK

### Python csomagok

robotframework==6.1.1  
robotframework-databaselibrary==1.2.4  
python-docx==0.8.11  
openpyxl==3.1.2  
pywin32==306

### Rendszerkövetelmények

* OS: Windows 10/11
* Python: 3.8+
* Outlook: Microsoft Outlook telepítve és konfigurálva
* Memória: Min. 4GB RAM
* Tárhely: Min. 1GB szabad hely

## KARBANTARTÁS

### Adatbázis tisztítás

DELETE FROM redundancia WHERE record\_date < date('now', '-90 days');  
VACUUM;

### Log fájlok rotáció

Get-ChildItem results\ -Name "\*.html" | Where-Object {$\_.LastWriteTime -lt (Get-Date).AddDays(-30)} | Remove-Item

## HIBAELHÁRÍTÁS

### Gyakori hibák

1. Email küldés hiba: Outlook újraindítás
2. DOCX olvasási hiba: Fájl integritás ellenőrzés
3. Adatbázis lock: Kapcsolatok bezárása

### Debug mód

Log To Console ${variable\_value}  
Log Detailed information DEBUG

---

🔧 Technikai dokumentáció - v2.1.0

📅 Frissítve: 2025.08.25