SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM

MAROSVÁSÁRHELYI KAR, TÁVKÖZLÉS SZAK

A green and black logo

Description automatically generated

**Szoftver rendszerek tervezése projekt**

**Email szerver tesztelő Pythonban**

Készítette: Jakab Eduárd és Kocsis Lenke

Témavezető: Dr. Szántó Zoltán

**2024**

**Tartalomjegyzék**

[1. Bevezető 4](#_Toc187363703)

[1.1. A projekt célja 4](#_Toc187363704)

[1.2. Célkitűzések 5](#_Toc187363705)

[2. Funkcionalitások 5](#_Toc187363706)

[2.1. A domain és az email-címek kezeléséhez 5](#_Toc187363707)

[2.1.1. Az api.py fájlban implementált funkciók: 6](#_Toc187363708)

[2.2. Tesztelési folyamat 8](#_Toc187363709)

[2.2.1. Új domain és email fiókok létrehozása 8](#_Toc187363710)

[2.2.2. Paragrafusok maximális számának meghatározása 9](#_Toc187363711)

[2.2.3. Billentyű figyelő aktiválása 9](#_Toc187363712)

[2.2.4. Szerver leterhelés 9](#_Toc187363713)

[2.2.5. A szerver leterhelése 9](#_Toc187363714)

[2.3. Rendszermonitorizálás 14](#_Toc187363715)

[2.3.1. Monitorizálási folyamat 14](#_Toc187363716)

[2.3.2. Lekérdezések 15](#_Toc187363717)

[2.3.3. A monitorizálás gyakorlati alkalmazása 16](#_Toc187363718)

[3. A Python kód struktúrája 17](#_Toc187363719)

[4. Technikai részletek 18](#_Toc187363720)

[5. Használati forgatókönyv 18](#_Toc187363721)

[5.1. Konfigurálás 19](#_Toc187363722)

[5.2. Indítás 19](#_Toc187363723)

[5.3. Leállítás 20](#_Toc187363724)

[5.4. Eredmények kiértékelése 20](#_Toc187363725)

[6. Előnyök 21](#_Toc187363726)

[6.1. Automatizált tesztelés 21](#_Toc187363727)

[6.2. Testreszabhatóság 21](#_Toc187363728)

[6.3. Átfogó elemzés 21](#_Toc187363729)

[6.4. Skálázhatóság és rugalmasság 22](#_Toc187363730)

[6.5. Proaktív hibakeresés és rendszeroptimalizálás 22](#_Toc187363731)

[6.6. Rugalmasság különböző környezetekben 22](#_Toc187363732)

[7. Összefoglaló 23](#_Toc187363733)

[7.1. Továbbfejlesztési lehetőségek 23](#_Toc187363734)

[7.1.1. Grafikus felhasználói felület (GUI) 23](#_Toc187363735)

[7.1.2. Adatvizualizáció a tesztelési eredményekhez 24](#_Toc187363736)

[7.1.3. Támogatás további protokollokhoz 24](#_Toc187363737)

[7.1.4. Automatikus skálázás, adaptív terheléskezelés 24](#_Toc187363738)

[7.1.5. Támogatás különböző operációs rendszerekhez és e-mail-szerverekhez 25](#_Toc187363739)

1. Bevezető

Az email szerverek az egyik legfontosabb infrastruktúrát képezik a digitális kommunikáció terén. Mivel ezek a rendszerek napi szinten rengeteg adatot küldenek és fogadnak, kritikus fontosságú, hogy az email küldési és fogadási folyamatok megbízhatóak és gyorsak legyenek. A világ számos vállalkozása és egyéb intézményei az emailt használják alapvető kommunikációs eszközként, így az email szerverek rendelkezésre állásának biztosítása nélkülözhetetlen. Ezen rendszerek működésének zavartalan biztosítása fontos a nagy terhelés alatt működő környezetekben, mint például a pénzügyi szektor, a közszolgáltatások, a hírközlési szolgáltatók, vagy bármely más, folyamatos adatkommunikációt igénylő szektorban.

A szerverek stabilitása és skálázhatósága az email kommunikáció megbízhatóságát alapvetően meghatározza, ezért alapvető, hogy az email szerverek minden esetben képesek legyenek kezelni a növekvő adatforgalmat, különösen csúcsidőszakokban. A különféle terhelési tesztek elvégzése és az erőforrások folyamatos figyelemmel kísérése lehetővé teszi a rendszer teljesítményének előrejelzését és finomhangolását, mielőtt azok éles környezetben esetleg problémákat okozhatnának.

* 1. A projekt célja

A projekt célja, hogy egy olyan automatizált tesztelő eszközt hozzunk létre, amely képes valós időben figyelni a szerver terhelés alatti működését, illetve mérni a rendszer teljesítményét a különböző terhelési szinteken. A Python programozási nyelvet választottuk, mivel az ideális eszközként szolgál a különböző protokollok (IMAP, SMTP) kezeléséhez, valamint az erőforrások monitorozásához. A Python egyszerűsége és rugalmassága lehetővé teszi, hogy a fejlesztés gyorsan haladjon, míg a széleskörű könyvtártámogatás biztosítja a szükséges funkciók könnyű elérhetőségét. A projektben alkalmazni kívánt automatizált megközelítés segít csökkenteni az emberi beavatkozás szükségességét, így hatékonyabbá téve a tesztelési folyamatokat, valamint minimalizálva a hibák előfordulását.

* 1. Célkitűzések

A projekt elkészítésének célkitűzései:

* email-szerver működésének szimulálása, vagyis egy saját, virtuális szerver létrehozása és leterhelése;
* rendszererőforrások monitorozása, vagyis a CPU terhelés, a memóriakihasználtság és a hőmérséklet adatainak naplózása;
* tesztelési adatok elemzése, vagyis az email-szerver skálázhatóságának és teljesítményének részletes kiértékelése;
* automatizáció és testreszabhatóság, vagyis könnyen paraméterezhető tesztelési folyamatok létrehozása (pl. teszt email címek száma, üzenetek hossza és küldési gyakorisága).

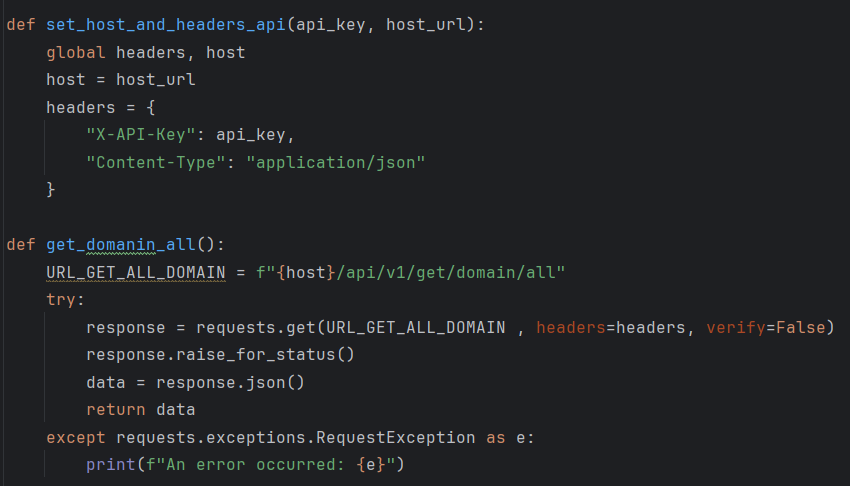
1. Funkcionalitások
   1. A domain és az email-címek kezeléséhez

A domain-ek és email-címek kezelése kulcsfontosságú a tesztelési rendszer dinamikus működéséhez. A projekt során alkalmazott API interfész biztosítja a szükséges kommunikációs kapcsolatot a rendszerünk és a domain kezelő szolgáltatás között. Az API használatával az email-címek és domain-ek kezelése automatizáltan történik, és a tesztelési folyamatok során a szükséges adatokat könnyen létrehozhatjuk, módosíthatjuk és törölhetjük. A requests könyvtár lehetővé teszi az egyszerű és hatékony HTTP-kérések küldését, az adatátvitelt JSON formátumban, valamint az API válaszok feldolgozását.

A rendszer az elkészített email-címeket JSON fájlokba menti, hogy azok később könnyen hozzáférhetők és újra felhasználhatók legyenek a tesztelési folyamatokban. A JSON formátum könnyen kezelhető és jól integrálható különböző eszközökkel, amely egyszerűsíti a tesztelési adatok értelmezését és feldolgozását. Mivel a tesztelési folyamatok során a rendszer számos dinamikusan generált email-címet használ, az adatok könnyű tárolása és kereshetősége rendkívül fontos.

Az api.py fájlban az API funkciók gondosan felépített struktúrában kezelik a REST végpontokat. A modul a domain-ek és email-fiókok dinamikus létrehozását, lekérdezését, és törlését valósítja meg, amely támogatja az automatizált tesztelési folyamatokat. A funkcionalitást Python requests könyvtár használatával, JSON formátumú adatcserével valósítottuk meg, amely segíti és megkönnyíti a következő, tesztelési folyamatot és a teszt adatok értelmezését, feldolgozását.

* + 1. Az api.py fájlban implementált funkciók:
* set\_host\_and\_headers\_api: Ez a funkció alapvetően előkészíti az összes API-hívást. Feladata, hogy minden API kérés tartalmazza az API-kulcsot és a megfelelő tartalomtípust. A REST API-k működése során elengedhetetlen, hogy minden kérés megfelelő autentikációval és tartalom típusokkal rendelkezzen, hogy a rendszer biztonságosan és szabványos módon működjön. Ez a funkció tehát előkészíti a kérések környezetét, és biztosítja a további API-hívások sima végrehajtását.
* get\_doman\_all: A get\_domain\_all funkció egy GET típusú HTTP-kérést küld, amely az összes elérhető domain-t lekérdezi. Ez a funkció lehetővé teszi, hogy a rendszer automatikusan lekérdezze az összes domain-t, amelyek már létre vannak hozva. Az API válaszában JSON formátumban kapjuk vissza a domain-ek listáját, amelyet a tesztelési folyamatokban felhasználhatunk, hogy ellenőrizzük, hogy a szükséges domainek rendelkezésre állnak-e a teszteléshez.
* get\_mailbox\_all: Ez a funkció hasonlóan működik, mint a get\_domain\_all, de ezúttal a domain-en belüli email-fiókok lekérdezésére szolgál. A get\_mailbox\_all segítségével lekérdezhetjük a meglévő email-fiókokat, és biztosíthatjuk, hogy a teszteléshez szükséges fiókok rendelkezésre állnak. A válasz JSON formátumban tartalmazza a fiók adatait, amelyeket a tesztelés során felhasználhatunk.
* create\_domain: A create\_domain egy POST típusú HTTP-kérés, amely egy új domain létrehozására szolgál. Az új domain létrehozásához szükséges adatokat JSON formátumban küldjük el, és az API válaszában visszakapjuk a létrehozott domain részletes adatait. Ez a funkció alapvető fontosságú, mivel lehetővé teszi, hogy dinamikusan hozzunk létre új domaineket a tesztelési ciklusok során, amelyeket a rendszer a későbbi tesztekhez felhasználhat.
* create\_mailbox: A create\_mailbox egy új email-fiók létrehozását teszi lehetővé egy adott domain-en belül, a create\_domain függvényhez hasonló módon.
* delete\_domains és delete\_mailboxes: A delete\_domains és a delete\_mailboxes funkciók POST típusú HTTP-kérésekkel végzik a domain-ek és email-címek törlését. Ezek a funkciók lehetővé teszik, hogy a tesztelés befejeztével eltávolítsuk a már nem szükséges domain-eket és fiókokat. A törléshez a szükséges adatokat JSON formátumban adjuk meg, amely egy listát tartalmaz a törlendő domain-ek vagy email-címek neveiről. Ezzel biztosítjuk, hogy a rendszer mindig naprakész és rendezett maradjon.

Kódrészletek az API funkciók implementálására:

* 1. A computer screen shot of a program code

     Description automatically generatedTesztelési folyamat

A tesztelési folyamat kulcsfontosságú szerepet játszik a rendszer teljesítményének és stabilitásának értékelésében. A cél az, hogy alaposan validáljuk az API által biztosított domain- és email-kezelési műveleteket, valamint teszteljük az e-mailek küldését és fogadását különböző terhelési szinteken. A tesztelési folyamat segítségével biztosíthatjuk, hogy a rendszer képes megfelelően kezelni a valós terhelési körülményeket, és biztosítani tudja az email szolgáltatások zökkenőmentes működését magas terhelés alatt is.

* + 1. Új domain és email fiókok létrehozása

Az első lépésben a create\_testers függvény segítségével új domain-t és email fiókokat hozunk létre. A teszteléshez szükséges domain-ek és fiókok létrehozása kritikus az automatikus teszteléshez, mivel minden tesztelési esethez új, dinamikusan generált címek szükségesek:

* + 1. Paragrafusok maximális számának meghatározása

Az email-ekben levő paragrafusok maximális számának megadása segít a tesztelési adataink szabályozásában. Ennek az értéknek az optimalizálása biztosítja, hogy minden email elég változatos legyen ahhoz, hogy megfelelően reprezentálja a valódi használati környezetet, de ne legyenek túlzottan hosszúak vagy túlságosan bonyolultak

* + 1. Billentyű figyelő aktiválása

A billentyű figyelő segítségével a felhasználók a tesztelés közben bármikor leállíthatják a folyamatot. Ez segít a tesztelés kézi megszakításában, ha hibákra, problémákra derül fény, vagy ha valamilyen nem várt esemény történik. A tesztelés irányításának ilyen módja biztosítja a rendszergazdák számára a szükséges kontrollt a tesztelési folyamatok felett:

* + 1. Szerver leterhelés

A load\_server\_capacity függvény meghívásával a rendszer terhelése alatt történő tesztelés céljából a tesztelési környezetet szimuláljuk. A szerver kapacitásának tesztelése azt célozza, hogy kiderüljön, milyen mértékig képes a rendszer fenntartani a stabilitást nagy mennyiségű email kezelésével, valamint, hogy mennyi ideig bírja el a növekvő terhelést.

* + 1. A szerver leterhelése

A terheléses tesztelés során a cél, hogy a rendszer stabilitását és skálázhatóságát a lehető legnagyobb terhelés mellett is validáljuk. A load\_server\_capacity függvény végzi el a terhelés generálását, és lehetővé teszi a különböző paraméterek megadását a teszteléshez. A paraméterek közé tartozik a küldés közötti várakozási idő, az email-címek száma és egyéb változók, amelyekkel finomhangolhatjuk a tesztelés intenzitását.

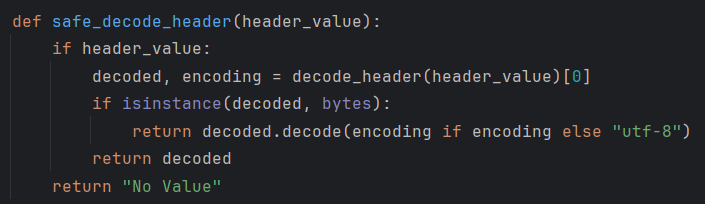
A folyamat lépései az alábbiak szerint alakulnak:

* A computer screen shot of a program code

  Description automatically generatedVárakozási idő és email-címek kezelése: A load\_server\_capacity függvény beállítja, hogy egy-egy email küldése között mennyi idő teljen el, illetve, hogy hány email-címmel dolgozzunk a tesztelés során. A várakozási idő beállítása fontos, mert segít szimulálni a terhelést a valós környezethez hasonlóan, ahol az email-ek nem azonnal kerülnek kiküldésre, hanem egy kis időeltolódással. A több email-cím használata pedig biztosítja, hogy a rendszer képes kezelni a több különböző fiókról történő párhuzamos üzenetküldést is.
* Küldő és fogadó email-címek elkülönítése: A rendszer különválasztja a küldő email-címeket a fogadó email-címektől. Ez fontos, hogy elkerüljük a körkörös levélküldést (amikor ugyanaz az email-cím küld és fogad ugyanabban a tesztben). Az elkülönítés biztosítja, hogy a tesztelés során az email-ek megfelelően legyenek kézbesítve és fogadva.
* Email-ek generálása és küldése: A send\_n\_per\_2\_email függvény segítségével az email-eket generáljuk és küldjük.
  + A screen shot of a computer program

    Description automatically generated A generálás során a rendszer véletlenszerű mondatokat generál a téma számára, és véletlenszerű paragrafusokat állít össze az email tartalmához. A csatolmányok elérési útját is megadjuk, ha szükséges. A randomizálás során külön figyelmet fordítunk arra, hogy az email-ek ne legyenek minden esetben ugyanolyanok. Az email-ek változatossága fontos, hogy a tesztelés a valós környezethez hasonlóan működjön.
  + A generált mondatok és paragrafusok előre meghatározott szókészletből kerülnek kiválasztásra. A program nemcsak a szavakat válogatja véletlenszerűen, hanem a mondatok hosszát, a paragrafusok számát és a paragrafusok hosszát is dinamikusan változtatja. Ez biztosítja, hogy a generált email-ek különböző hosszúságúak és szerkezetűek legyenek, ami reális terhelést jelent a rendszer számára.
  + Az email-ek tényleges összeállítása MIME-formátumban történik, amely az email-küldés egyik alapvető szabványa. A send\_email\_with\_attachment függvény az SMTP protokollt használja a levelek elküldéséhez, és biztosítja, hogy az email titkosítása is megtörténjen a server.starttls() hívással. Ez a biztonsági lépés szükséges a titkosított kommunikációhoz, amely védelmet biztosít az email-ek adatainak integritásához.
* Email-ek fogadása és mentése: A recieve\_n\_per\_2\_email függvény segítségével az email-eket fogadjuk és mentjük el a megadott helyre. Ehhez a rendszer az IMAP protokollt használja, amely lehetővé teszi az email-ek letöltését és feldolgozását a szerverről. A fetch\_and\_save\_email függvény felelős azért, hogy az IMAP szerverhez csatlakozva letöltse az e-maileket, azok tartalmát és mellékleteit, majd rendszerezve elmentse őket egy helyi könyvtárba.
  + Célja, hogy egy megadott e-mail fiókból letöltse az e-mailek tartalmát és mellékleteit, majd ezeket rendszerezve elmentse egy helyi könyvtárba.
  + A folyamat azzal kezdődik, hogy a függvény biztonságosan csatlakozik az IMAP szerverhez a felhasználó e-mail címével és jelszavával.
  + Az egyes e-mailekhez tartozó adatok letöltése után a program rendszerezve menti azokat egy helyi könyvtárba.
  + A program automatikusan felismeri, hogy egy e-mail multipart tartalmú-e (pl. többféle adat, mint szöveg és mellékletek is vannak benne), és ennek megfelelően dolgozza fel az üzenetet.
  + Ha egy e-mail szövege több részletben van kódolva, azt dekódolja és összefűzi.
  + A mellékletek is dekódolásra kerülnek, majd az eredeti fájlnevük alapján mentődnek el.
  + Ha bármelyik lépés során hiba történik, például, ha egy mappa nem érhető el, vagy egy e-mail nem tölthető le, a program azt figyelmeztető üzenetként naplózza.
  + A folyamat végén a függvény bontja a kapcsolatot az IMAP szerverrel, és jelzi a felhasználónak, hogy a letöltés sikeresen befejeződött.
  + A letöltött adatok így teljes mértékben elérhetőek offline, és biztonságosan mentve vannak a megadott könyvtárban.

A computer screen with text and images

Description automatically generatedAhogy a leírás is szemlélteti, az IMAP protokoll használata egy, valamivel hosszabb és bonyolultabb, kód megírását igényli. Íme néhány részlet a fetch\_and\_save\_email függvényből:

* 1. Rendszermonitorizálás

A rendszermonitorizálás az informatikai rendszerek karbantartásának és optimalizálásának egyik elengedhetetlen része. A szerverek folyamatos figyelemmel kísérése lehetővé teszi a potenciális problémák korai felismerését és gyors beavatkozást igénylő helyzetek elkerülését.

A getting\_server\_info függvény egy olyan folyamatos monitorozási folyamatot indít el, amely periódikusan, a megadott frissítési gyakoriság szerint, lekérdezi és megjeleníti a szerver aktuális teljesítmény- és állapotadatait. Ezen információk figyelése rendkívül fontos, mivel segítséget nyújtanak a rendszergazdák számára a szerverek állapotának valós idejű követésében, különösen nagy terhelés alatt, vagy kritikus időszakokban.

A rendszer ezen része lehetőséget biztosít arra, hogy a szerverek működését folyamatosan nyomon követhessük, és amennyiben bármilyen problémát észlelünk (például magas CPU használat, memória-túlterhelés, vagy a processzor hőmérsékletének emelkedése), gyorsan reagálhassunk, így minimalizálhatjuk a rendszer leállásának esélyét.

* + 1. Monitorizálási folyamat

A getting\_server\_info függvény folyamatosan gyűjti a szerver adatait, miközben valós időben biztosít visszajelzést az aktuális állapotról. Az információk frissítése minden egyes ciklusban megtörténik, és az adatok a képernyőre kerülnek megjelenítésre. A frissítési ciklus frekvenciája a rendszerbeállításoktól függően 20 Hz-en történik, vagyis a szerver állapotát másodpercenként 20 alkalommal lekérdezi a rendszer, és az új adatokat azonnal megjeleníti.

A monitorozási program egy interaktív felületet biztosít, amely lehetővé teszi a felhasználó számára, hogy a figyelési folyamatot bármikor leállítsa. A tesztelés vagy ellenőrzés végeztével a felhasználó a q billentyű lenyomásával vagy a program megszakításával (például a **Ctrl+C** billentyűkombinációval) szüntetheti meg a monitorozást. Ezzel biztosítva van, hogy a figyelési folyamat irányítása mindig a felhasználó kezében maradjon, és a szükséges adatokat akkor lehessen gyűjteni, amikor valóban szükség van rájuk.

A get\_server\_info függvény az alapértelmezett paramiko könyvtár segítségével csatlakozik egy távoli szerverhez. A paramiko egy erőteljes Python könyvtár, amely SSH kapcsolatokat hoz létre, és képes távoli parancsokat futtatni a csatlakoztatott gépen. Ez a könyvtár ideális eszköz a szerverek távoli kezelésére és monitorozására, mivel biztosítja a biztonságos, titkosított kommunikációt a szerverek és az adminisztrátorok között.

Miután a csatlakozás létrejön, a get\_server\_info függvény három kulcsfontosságú Linux parancsot futtat le a szerveren, hogy a szükséges információkat begyűjtse a szerver állapotáról.

* + 1. Lekérdezések

CPU információk lekérdezése (lscpu parancs): Az első parancs, amelyet a függvény végrehajt, a lscpu. Ez a parancs alapvető információkat szolgáltat a szerver CPU-járól, beleértve annak típusát, a rendelkezésre álló magok számát, a processzor frekvenciáját és egyéb jellemzőit. A CPU terhelése és működési jellemzői alapvető információk ahhoz, hogy meghatározzuk, vajon a rendszer képes-e kezelni az adott terhelést anélkül, hogy túlzottan lassulna vagy leállna.

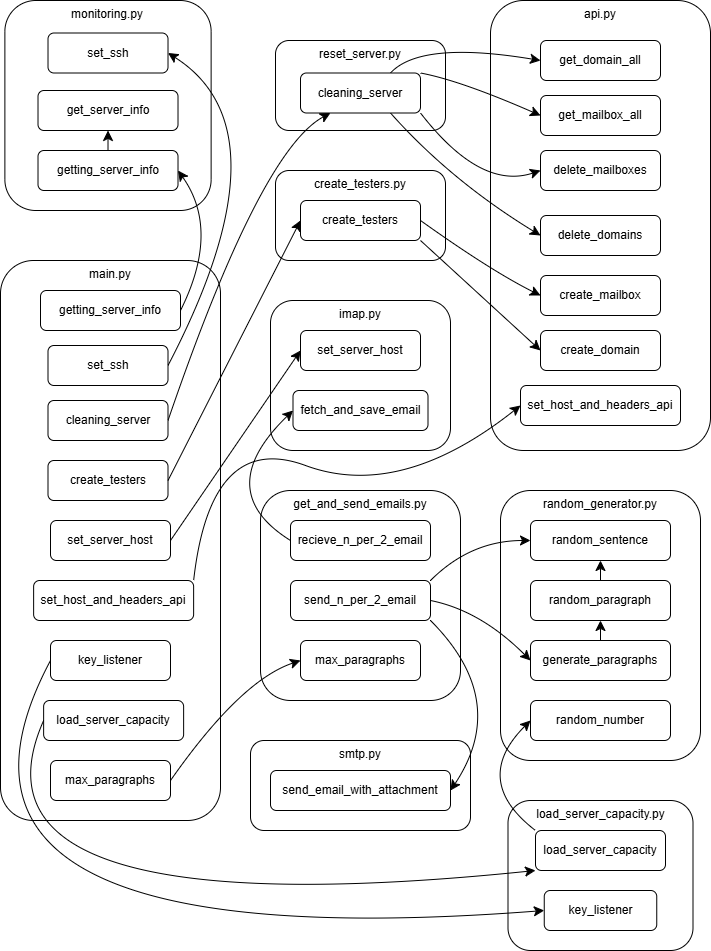
**Processzor hőmérsékletének lekérdezése (**sensors **parancs)**: A második parancs, amit a függvény futtat, a sensors. Ez a parancs a rendszer hőmérsékleti szenzorainak állapotát kérdezi le, így biztosítva a processzor hőmérsékletének nyomon követését. A **lm-sensors** csomag telepítése szükséges ahhoz, hogy ez a parancs megfelelően működjön. A rendszer hőmérséklete kritikus tényező lehet a szerver teljesítményének és megbízhatóságának fenntartásában. Ha a hőmérséklet túl magasra emelkedik, akkor ez jelezheti, hogy a processzor túlterhelt vagy hogy hűtési problémák léptek fel. Ebben az esetben fontos, hogy a rendszergazda gyorsan intézkedjen. Ha a hőmérsékleti adatok nem érhetők el (például, ha a megfelelő csomagok nincsenek telepítve, vagy ha a szerver nem rendelkezik hőmérséklet-érzékelőkkel), a függvény figyelmeztető üzenetet ad a felhasználónak.

Memória állapotának lekérdezése (free -h parancs): A harmadik parancs a free -h, amely a rendszer memóriahasználatáról szolgáltat információkat. Ez megmutatja a szabad és használt memória mennyiségét, valamint az elérhető swap területet. A memóriahasználat magas szintje jelezheti, hogy a rendszer több erőforrást használ, mint amit valójában képes kezelni, vagy hogy memóriahiányos helyzetek alakulhatnak ki. Az ilyen problémák hatással lehetnek a rendszer teljesítményére, és szükség lehet további memória hozzáadására vagy a memória kezelésének optimalizálására.

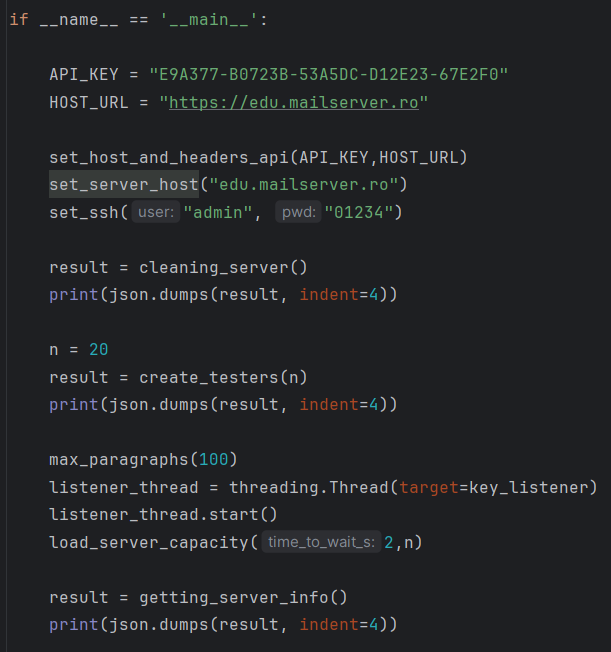
* + 1. A monitorizálás gyakorlati alkalmazása

A monitorozás folyamata segíti a rendszergazdákat abban, hogy proaktívan beavatkozhassanak, ha a rendszer erőforrásai (pl. CPU vagy memória) túlzottan leterheltek, vagy ha a szerver hőmérséklete veszélyes szintre emelkedik. Ezenkívül az adatok elemzése segíthet abban, hogy meghatározzák a rendszer optimális működési paramétereit és észleljék azokat a terhelési mintákat, amelyek hatással lehetnek a rendszer hosszú távú stabilitására.

A rendszer tehát nem csupán a jelenlegi állapot monitorozására szolgál, hanem eszközként is, amely segít a rendszer folyamatos karbantartásában és a potenciális problémák gyors kezelésében. Az automatikus és folyamatos monitorozás révén a rendszergazdák biztosíthatják, hogy a szerverek mindig optimális környezetben működjenek, és gyorsan reagálhatnak, ha a rendszer bármely területén eltérést tapasztalnak.

1. A Python kód struktúrája
2. Technikai részletek

* Fejlesztési nyelv: Python.
* Használt protokollok és eszközök: IMAP és SMTP, az email-ek küldéséhez és fogadásához az imaplib és smtplib Python könyvtárak.
* API kezelés: a requests könyvtár az API hívásokhoz.
* Rendszerfigyelés: a psutil könyvtár a CPU, memória és hőmérséklet monitorozásához.
* Email cím generálás: véletlenszerű címek létrehozása a domain alapján, például: [tester1@test.com](mailto:tester1@test.com), [tester2@test.com](mailto:tester2@test.com).
* Az email címek ideiglenes eltárolása JSON fájlban.

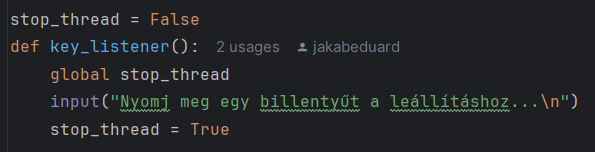
1. Használati forgatókönyv
   1. Konfigurálás

A felhasználó az alábbi paraméterek megadásával konfigurálja a tesztelést:

* **n**: A létrehozandó teszt e-mail címek száma, amelyeket a rendszer automatikusan generál és regisztrál a teszt előtt.
* **k**: A véletlenszerűen generált e-mailek szövegeinek hosszát szabályozó paraméter, amely befolyásolja a mondatok és paragrafusok számát az egyes üzenetekben.
* **t**: Az e-mailek közötti időköz, amely a küldési sebességet szabályozza és biztosítja, hogy a terhelés fokozatosan épüljön fel.
  1. Indítás

Miután a felhasználó megadta az említett paramétereket, elindul a tesztelési folyamat, amely a következő lépésekből áll:

* Szerver alaphelyzetbe állítása: A tesztelés előtt a rendszer automatikusan végrehajt egy resetelési folyamatot, amely magában foglalja a korábban létrehozott domain-ek és e-mail címek törlését. Ez biztosítja, hogy a teszt a legfrissebb, tiszta környezetben kezdődjön.
* E-mailek küldése: A rendszer a konfigurált paraméterek alapján véletlenszerűen generál e-maileket, majd azokat a létrehozott címekre elküldi. A küldési folyamat során az időközöket figyelembe véve, az üzenetek terjedelme és tartalma dinamikusan változik.
* Szerver teljesítményének figyelése: A tesztelés alatt a szerver állapotát folyamatosan monitorozza a rendszer. Az erőforrások, mint a CPU használat, memóriakihasználtság és hőmérséklet, valós időben kerülnek lekérdezésre és kiértékelésre, biztosítva, hogy a terhelés alatt a rendszer ne szenvedjen teljesítményproblémákban.
* E-mailek fogadása és elmentése: A küldött e-mailek érkezése és feldolgozása is része a tesztnek. A rendszer minden fogadott e-mailt letölt és rendszerezi a megadott könyvtárban, lehetővé téve a teszt során felmerült hibák gyors azonosítását.
  1. Leállítás

 A tesztelés folyamatosan fut, és a felhasználó bármikor leállíthatja a folyamatot egy billentyű lenyomásával:

A teszt során összegyűjtött adatok, beleértve az e-mailek küldésére és fogadására vonatkozó statisztikákat, valamint a rendszer teljesítményét, részletes naplózásra kerülnek, amely később elemzés céljából felhasználható. Az így nyert eredmények lehetőséget biztosítanak a rendszer optimalizálására és a potenciális problémák gyors felderítésére.

* 1. Eredmények kiértékelése

A teszt befejeztével az összegyűjtött adatokat kiértékelhetjük, beleértve a szerver terhelésére vonatkozó teljesítmény mérését és az e-mail küldési/fogadási statisztikákat. A rögzített naplók és adatpontok segítségével könnyen azonosíthatók a rendszerben esetlegesen felmerülő hibák, amelyeket a fejlesztők a jövőbeni optimalizálás során felhasználhatnak.

1. Előnyök

A projekt számos előnyt kínál mind a tesztelési folyamatok automatizálásában, mind a rendszer teljesítményének és stabilitásának optimalizálásában. Az alábbiakban bemutatjuk a legfontosabb előnyöket, amelyek a fejlesztett eszköz használatával érhetők el:

* 1. Automatizált tesztelés

A rendszer automatizálja az e-mail küldést és fogadást, valamint a szerver terhelés alatti teljesítményének figyelését. Ennek eredményeként a tesztelési folyamat gyorsan és hatékonyan elvégezhető anélkül, hogy manuálisan kellene végrehajtani minden lépést. Ez jelentős időmegtakarítást jelent a tesztelés és a hibakeresés során, mivel az automatizált eszköz a teljes tesztelési ciklus alatt folyamatosan képes figyelemmel kísérni és rögzíteni az adatokat és gyorsabban képes reagálni a terhelésre és a környezeti tényezőkre.

* 1. Testreszabhatóság

Az eszköz testreszabható paraméterezési lehetőségeket biztosít, amelyek lehetővé teszik a felhasználó számára, hogy a tesztelési környezetet a saját igényeihez igazítsa. A felhasználók pontosan meghatározhatják a tesztelés paramétereit, mint például az e-mail címek számát, az egyes e-mailek szövegének hosszát és a küldési időközöket. Ez lehetővé teszi a különböző terhelési szintek és e-mail forgalmi minták tesztelését, amelyek segítenek megérteni, hogyan reagál a szerver különböző körülmények között. A testreszabhatóság növeli az eszköz rugalmasságát és alkalmazhatóságát különböző típusú e-mail-szerverek és vállalati infrastruktúrák tesztelésére.

* 1. Átfogó elemzés

Az eszköz részletes és átfogó adatokat gyűjt és biztosít az e-mail-szerver teljesítményéről, amelyeket később részletesen elemezhetünk. A szerver állapotának folyamatos monitorozása az erőforrások felhasználását is figyelemmel kíséri. A rendszer rögzíti a CPU használatot, memóriahasználatot és a hőmérsékleti adatokat, és lehetővé teszi az erőforrások optimalizálását a tesztelési eredmények alapján. A felhasználók könnyen azonosíthatják a szerver teljesítményének gyenge pontjait, ami segít a konfiguráció finomhangolásában és a problémák előrejelzésében.

* 1. Skálázhatóság és rugalmasság

Az eszköz képes különböző terhelési szintek kezelésére, azaz lehetőséget biztosít arra, hogy a tesztelést különböző mértékű e-mail forgalom mellett végezzük el. Az eszköz képes alkalmazkodni a változó igényekhez, és a tesztelési környezetet dinamikusan állíthatjuk a kívánt paramétereknek megfelelően. Az eszköz képes különböző terhelési forgatókönyveket szimulálni, így lehetőség van a szerver maximális teljesítményének és stabilitásának tesztelésére éles környezethez hasonló körülmények között.

* 1. Proaktív hibakeresés és rendszeroptimalizálás

Az eszköz folyamatosan figyeli a rendszert és a teljesítményt, és képes azonnal jelezni a problémákat. Az e-mailek küldésének és fogadásának sebessége, a rendszer erőforrásainak kihasználtsága és az esetleges hibák azonnali rögzítése segíti a rendszergazdákat a gyors hibaelhárításban és a rendszer optimalizálásában. A tesztelés során gyűjtött adatok segítenek abban, hogy a rendszer a legjobb teljesítményt nyújtsa a legmagasabb terhelés mellett is, ezáltal csökkentve a jövőbeni működési problémák kockázatát.

* 1. Rugalmasság különböző környezetekben

A tesztelő eszköz kompatibilis különböző operációs rendszerekkel és e-mail szerverekkel, így lehetővé teszi a különböző vállalati vagy fejlesztési környezetekben történő alkalmazását. A különböző beállításoknak köszönhetően a rendszer képes alkalmazkodni a specifikus igényekhez, és képes hatékonyan működni mind a kis-, mind a nagyvállalati infrastruktúrákban, figyelembe véve azok sajátos terhelési és teljesítményi elvárásait.

1. Összefoglaló

Ez a projekt egy rendkívül hasznos eszközt biztosít az e-mail-szerverek skálázhatóságának és stabilitásának tesztelésére, miközben folyamatosan figyelemmel kíséri és rögzíti a rendszererőforrások használatát, segítve ezzel a teljesítményoptimalizálást. Az automatizált tesztelési eszköz képes szimulálni különböző terhelési forgatókönyveket, így lehetőség nyílik arra, hogy a szerverek valódi használati környezethez hasonló terhelés mellett kerüljenek tesztelésre. A tesztelési folyamatok automatizálása és a rendszerfigyelés részletes monitorozása nagyban hozzájárul a megbízhatóság, a stabilitás és a teljesítmény javításához.

A projekt kiemelkedő erőssége, hogy testreszabható és skálázható, így különböző infrastruktúrák és tesztelési igények szerint is alkalmazható. Az e-mailek véletlenszerű generálásának és küldésének folyamata, a szerverek terhelése alatt végzett teljesítményfigyelés és az összegyűjtött adatok elemzése révén az eszköz segíti a rendszergazdákat abban, hogy az e-mail-szerverek és kapcsolódó rendszerek mindig optimálisan működjenek, és minden lehetséges problémát proaktívan felismerjenek.

* 1. Továbbfejlesztési lehetőségek

Bár a projekt jelenlegi verziója már funkcionalitásban gazdag és hatékony, számos fejlesztési irány kínálkozik, amelyek még inkább javíthatják az eszköz használhatóságát, és új lehetőségeket nyithatnak meg a tesztelés és monitorozás során.

* + 1. Grafikus felhasználói felület (GUI)

A jelenlegi rendszer parancssori eszközként működik, amely bár hatékony és rugalmas, egyes felhasználók számára kevésbé intuitív lehet. A grafikus felhasználói felület (GUI) bevezetésével a tesztelési folyamatok és eredmények könnyebben áttekinthetők lennének, így a felhasználók vizuálisan is követhetik a tesztelés előrehaladását. A GUI segítségével a paraméterek beállítása, a tesztelési folyamatok indítása és leállítása, valamint az eredmények értékelése egyszerűsödne, különösen azok számára, akik nem rendelkeznek mélyebb parancssori ismeretekkel. Ezzel a fejlesztéssel a rendszer még elérhetőbbé válna egy szélesebb felhasználói kör számára.

* + 1. Adatvizualizáció a tesztelési eredményekhez

A tesztelési eredmények részletes megjelenítése érdekében hasznos lenne egy adatvizualizációs eszközkészlet integrálása. A különböző mérési eredmények – mint a CPU terhelés, memóriahasználat, e-mailek küldésének és fogadásának sebessége – grafikus ábrázolása lehetővé tenné a felhasználók számára, hogy gyorsan és könnyedén értékeljék a rendszer teljesítményét. Az ilyen típusú vizualizációk, például vonaldiagramok, oszlopdiagramok és hőtérképek, segítenének a trendek gyors felismerésében és az esetleges teljesítményproblémák azonosításában, amelyek esetleg nem lenne olyan könnyen észrevehetők a nyers adatokban.

* + 1. Támogatás további protokollokhoz

Bár a jelenlegi verzió az SMTP protokollt használja az e-mailek küldésére és fogadására, a további protokollok, mint például a POP3 és IMAP4 hozzáadása lehetőséget biztosítana a rendszer számára, hogy még átfogóbb tesztelési eszközzé váljon. A POP3 és IMAP4 az e-mailek letöltésére és kezelésére szolgálnak, és széles körben használtak a levelező rendszerekben. A különböző protokollok támogatása lehetővé tenné a felhasználók számára, hogy különböző e-mail kommunikációs formákat teszteljenek, és a rendszer teljesítményét az e-mail szerverek más típusú forgalma alatt is figyelemmel kísérjék.

* + 1. Automatikus skálázás, adaptív terheléskezelés

A jövőben a rendszer képes lehetne dinamikusan alkalmazkodni a tesztelt szerver aktuális teljesítményéhez. Az automatikus skálázás és adaptív terheléskezelés lehetővé tenné, hogy a tesztelés a szerver erőforrásainak kihasználtságát figyelembe véve állítsa be a tesztelési paramétereket, így elkerülhető lenne a túlzott terhelés, miközben folyamatosan biztosítható lenne a teljesítményhatárok tesztelése. A rendszer képes lenne figyelni a szerver válaszidejét, az erőforrások kihasználtságát, és ennek megfelelően állítaná be a tesztelési intenzitást.

* + 1. Támogatás különböző operációs rendszerekhez és e-mail-szerverekhez

A rendszer fejlesztésével további operációs rendszerek és e-mail-szerverek támogatása is fontolóra vehető. Az eszköz kiterjeszthető úgy, hogy kompatibilis legyen olyan különböző platformokkal, mint például a Microsoft Exchange, Zimbra vagy a Postfix, valamint különböző Linux-alapú disztribúciókkal és Windows környezetekkel. Ez a bővítés lehetővé tenné, hogy a tesztelés minden környezetben elvégezhető legyen, ahol e-mail-szerverek működnek, és biztosítaná, hogy a rendszer széleskörű használhatóságot nyújt.

A fent említett fejlesztések révén a rendszer nemcsak a tesztelés hatékonyságát és rugalmasságát növelhetné, hanem a felhasználói élményt is javíthatná, így a jövőbeni igényeknek és kihívásoknak is megfelelne.