Környezet elindítása

- Clone-ozzuk a repot. Ha szeretnék https/SSL-t akkor a Ninjastic/certs mappába tegyünk bele a domainhez tartozó .crt és .key file-t (pl: ninjastic.pro.crt, ninjastic.pro.key), majd módosítsuk a Ninjastic/docker-compose.yaml file-t:

```
services:
  nginx-proxy:
    image: jwilder/nginx-proxy:1.5.1
    ports:
      - "80:80"
      #- "443:443"
    volumes:
      - /var/run/docker.sock:/tmp/docker.sock
      #- ./certs:/etc/nginx/certs
    environment:
      - HTTP_PORT=80
      #- HTTPS_PORT=443
      #- VIRTUAL_PROTO=https
```

Az nginx-proxy service-ben vegyük ki a ”#”-eket ports, volumes és az environment résznél. Ezzel elérjük hogy a reverse-proxy-nk https-t használjon és felolvassa a cert-eket.

- Lépjünk be a Ninjastic mappába és a következő parancsal elindítjuk a database-t, api-t, admin, reverse-proxy-t és a frontendet:

```
sudo docker-compose up -d
```

- Miután a környezet elindult be kell exec-elni az egyik backend containerbe:

```
docker exec -it ninjastic-api bash
```

- Ezután pedig lefuttatjuk a migrációt:

```
php bin/console --no-interaction doctrine:migrations:migrate
```

- Majd a dirty words commandot ami az adatbázis elmenti a csúnya szavakat:

```
php bin/console app:process-dirty-words-xml
```

- A host fájlba vegyük fel a következőt:
  - 127.0.01 api.ninjastic.pro admin.ninjastic.pro ninjastic.pro
- Windows hosts fájl: C:\Windows\System32\drivers\etc\hosts
- Linux hosts fájl: /etc/hosts

- Ezekután a következő url-eken érhetjük el a weboldalkat:
  - API: api.ninjastic.pro
  - Admin: admin.ninjastic.pro
  - Frontend: ninjastic.pro

Fejlesztői Dokumentáció - API

Általános információk:

- Nyelv: PHP 8.2
- Keretrendszer: Symfony 7
- Swagger dokumentáció endpointja: /api/v1/swagger
- Swagger dokumentáció endpointja JSON formátumba: /api/v1/swagger/json
- Docker verzió:
- Docker-compose verzió:

Biztonság:

- Bearer Token: JWT (JSON Web Token)

Működés/Komponensek leírása

Docker-compose:

Docker Compose egy eszköz a többkonténeres alkalmazások meghatározására és futtatására. Ez az eszköz az egyszerű és hatékony fejlesztési és telepítési tapasztalat kulcsa.

A Compose egyszerűsíti teljes alkalmazás vezérlését, egyszerűvé téve a szolgáltatások, hálózatok és kötetek kezelését egyetlen, áttekinthető YAML konfigurációs fájlban. Ezután egyetlen parancs segítségével létrehozhatod és elindíthatod az összes szolgáltatást a konfigurációs fájlból.

Swagger:

- Swagger, Swagger UI és OpenAPI 3 mind olyan eszközök és specifikációk, amelyek segítenek API-k tervezésében, dokumentálásában és tesztelésében.
- Swagger: Ez egy nyílt forráskódú keretrendszer és specifikáció, amely lehetővé teszi fejlesztők számára az API-k tervezését, dokumentálását és tesztelését. A Swagger specifikáció egy egyszerű, de hatékony módszer arra, hogy leírják az API-kat, beleértve az elérhető végpontokat, azok paramétereit és válaszait.
- Swagger UI: Ez egy interaktív, webes felhasználói felület, amely a Swagger specifikációk alapján generál dokumentációt az API-ról. A Swagger UI lehetővé teszi a fejlesztők és más érdeklődők számára, hogy könnyen navigáljanak az API dokumentációjában, kipróbálják az egyes végpontokat és megismerjék azok működését.
- OpenAPI 3: Ez egy specifikáció, amely az API-k leírását és dokumentálását szolgálja, korábban ismert Swagger specifikációként. Az OpenAPI 3 a Swagger specifikáció egy továbbfejlesztett változata, amely több funkciót és lehetőséget biztosít az API-k részletes leírásához és dokumentálásához. Az OpenAPI 3 a RESTful API-k tervezésére és dokumentálására kínál standardizált módszert.

Összességében a Swagger, Swagger UI és OpenAPI 3 olyan eszközök és specifikációk, amelyek segítik a fejlesztőket az API-k hatékonyabb és átláthatóbb tervezésében, dokumentálásában és tesztelésében, ami hozzájárul az alkalmazások jobb minőségű és könnyebben használható integrációjához.

Swagger endpointok:

- Swagger UI: /api/v1/swagger endpointon
- OpenAPI 3 JSON: /api/v1/swagger/jons

JWT (JSON Web Token):

JWT (JSON Web Token): Ez egy nyílt szabvány (RFC 7519), amely szöveges adatot tartalmaz és biztonságos módon küldhető egy JSON formátumú objektumban. A JWT-k gyakran használatosak az autentikáció során, mint például a felhasználó azonosítása egy alkalmazásban vagy egy webhelyen. Egy JWT három részből áll: fejléc, terhelés és aláírás. A fejléc tartalmazza a token típusát és az alkalmazott algoritmust, a terhelés tartalmazza az adatokat (például felhasználói információkat), míg az aláírás pedig egy titkos kulcs használatával készül, és a token valódiságát ellenőrzi.

Bearer:

Bearer: Ez egy hitelesítési sémának azon típusa, amely azt jelzi, hogy a kliens (általában egy alkalmazás vagy felhasználó) rendelkezik egy hitelesítő token-nel, amelyet egy szolgáltatás vagy erőforrás hitelesítésére használ. A Bearer token általában a HTTP kérés fejlécében található meg, és az "Authorization" mezőben kerül átvitelre. A Bearer token azonosítja és hitelesíti a klienst a szolgáltatás vagy erőforrás felé.

Ezeket az eszközöket gyakran kombinálják az alkalmazások és webszolgáltatások biztonságosabbá tételére. A JWT-k segítségével a felhasználói adatok biztonságosan továbbíthatók és tárolhatók, míg a Bearer tokenek lehetővé teszik a hitelesített hozzáférést az erőforrásokhoz vagy szolgáltatásokhoz a kliensek számára.

Symfony API-hoz használt bundle-ök a JWT Authentikációhoz és refresh tokenhez:

- Symfony JWT bundle
- Symfony Refresh Token bundle

Nginx és Reverse-Proxy:

Az NGINX egy kiválóan teljesítő, könnyű és nagyon rugalmas webkiszolgáló szoftver, amely a kiszolgáló oldali feladatokra specializálódik. Az NGINX-t széles körben használják olyan célokra, mint a statikus tartalmak szolgáltatása, terheléselosztás, proxy szolgáltatások, SSL és TLS titkosítás, valamint sok más feladat megoldására.

A "Reverse Proxy" egy olyan szolgáltatás, amely lehetővé teszi a bejövő kérések átirányítását egy belső hálózaton belül található vagy más távoli szerverekre. Az NGINX Reverse Proxy lehetővé teszi, hogy egy központi helyen kezeljük a beérkező kéréseket, majd ezeket továbbítsuk a megfelelő helyre.

Néhány fő előnye az NGINX Reverse Proxy-nak:

- Terheléselosztás: Az NGINX képes elosztani a bejövő kéréseket több háttérszerver között, így csökkentve a szerverterhelést és javítva a teljesítményt.
- Titkosítás és biztonság: Az NGINX segítségével titkosíthatjuk a kommunikációt a kliens és a háttérszerver között SSL vagy TLS segítségével.
- Címzés átírása: Az NGINX Reverse Proxy képes átírni a bejövő kérések címzését, így lehetővé teszi a belső infrastruktúra számára, hogy ne kelljen közvetlenül exponálva lennie a külvilág felé.
- Cache kezelés: Az NGINX képes cache-elni a gyakran használt tartalmakat, így csökkentve a háttérszerver terhelését és növelve a teljesítményt.
- Webalkalmazás tűzfal: Az NGINX lehetővé teszi a beérkező kérések ellenőrzését és szűrését, így megvédve a háttérszervereket a rosszindulatú támadásoktól.

Ezek az NGINX Reverse Proxy funkciói segítenek optimalizálni az infrastruktúrát, javítani a teljesítményt és biztosítani a biztonságot az internetes alkalmazások számára.

Cloudflare:

Az demó oldalunk cloudflare mögött van.
Cloudflare egy olyan szolgáltató, amely globális tartalomelosztó hálózatot (CDN-t), webbiztonsági megoldásokat és egyéb teljesítményjavító szolgáltatásokat kínál webhelyek és alkalmazások számára. Itt van néhány kulcsfontosságú jellemzője:

- Tartalomelosztó hálózat (CDN): Cloudflare rendelkezik egy világszerte elosztott tartalomelosztó hálózattal, amely lehetővé teszi a webes tartalmak gyors és hatékony szállítását a felhasználók felé. Ennek eredményeként a weboldalak gyorsabban töltődnek be és kevesebb terhelést generálnak a szerverekre.
- Webbiztonság: Cloudflare biztonsági szolgáltatásai közé tartozik a DDoS támadások elleni védelem, a webalkalmazás-tűzfal (WAF), a botvédelem és a SSL/TLS titkosítás. Ezek a megoldások segítenek védeni a webhelyeket a kibertámadásoktól és biztosítani a felhasználók biztonságát.
- Teljesítményjavító szolgáltatások: A Cloudflare számos olyan funkciót kínál, amelyek segítenek optimalizálni a webhelyek teljesítményét, például a tartalom gyorsítótár (cache) szolgáltatást, az automatikus kép- és fájl optimalizációt, valamint a HTTP/2 és HTTP/3 támogatást.
- DNS szolgáltatás: Cloudflare egy vezető DNS szolgáltató is. A DNS szolgáltatásának segítségével lehetőség van a domain név feloldására és átirányítására a webes szerverek felé.
- Analitikai eszközök: Cloudflare lehetőséget biztosít az ügyfeleknek arra, hogy nyomon kövessék a webhelyük teljesítményét és biztonságát, valamint részletes adatokat kapjanak a forgalomról és a kibertámadásokról.

Apache2:

Az API és az admin Apach2-t használ hogy kívülről elérhető legyen.

Apache HTTP szerver egy ingyenes és nyílt forráskódú webszerver szoftver, amelyet széles körben használnak a világ minden táján. Az Apache szoftver egyike a legelterjedtebb webszervereknek az interneten, és sok webhely és alkalmazás alapja.

Apache HTTP szerver fő jellemzői és funkciói:

- Nyílt forráskódú: Az Apache HTTP szerver ingyenesen elérhető, és nyílt forráskódú, ami azt jelenti, hogy bárki számára hozzáférhető és módosítható a forráskódja a felhasználók által.
- Platformfüggetlen: Az Apache több platformon is futtatható, beleértve a Linuxot, Unixot, Windowst és más operációs rendszereket is, ami lehetővé teszi a széles körű alkalmazást és elérhetőséget.
- Modularitás: Az Apache HTTP szerver moduláris architektúrával rendelkezik, ami azt jelenti, hogy különböző funkciókat és bővítményeket lehet hozzáadni vagy eltávolítani az igényeknek megfelelően. Ezek a modulok lehetővé teszik különböző funkciók, mint például az SSL/TLS titkosítás, az URL átirányítás, a terheléselosztás, a kompresszió és még sok más.
- Teljesítmény: Az Apache HTTP szerver kiváló teljesítményt nyújt, különösen alacsony és közepes terhelésű webhelyek esetén. Nagyon hatékonyan kezeli a statikus tartalmakat, valamint dinamikus tartalmakat, mint például a PHP vagy más szerveroldali szkriptek által generált tartalmak.
- Közösség és támogatás: Az Apache szoftver egy nagy és aktív fejlesztői és felhasználói közösséggel rendelkezik, ami jelentős segítséget és támogatást nyújt az új felhasználóknak és a fejlesztőknek.

RESTful API:

Egy RESTful API (Application Programming Interface) egy olyan API, amely a REST (Representational State Transfer) architektúráján alapul.

Doctrine:

Doctrine egy PHP nyelven írt objektum-relációs leképző (ORM) eszköz, amely segítségével könnyen és hatékonyan lehet adatbázisokat kezelni a PHP alkalmazásokban. Az ORM technika lehetővé teszi az objektumok és az adatbázisok közötti átjárást, így a fejlesztőknek nem kell közvetlenül SQL lekérdezéseket írniuk.
A mi rendszerünk is a Doctrine ORM-et használja.

Doctrine fő jellemzői:

- Objektum-relációs leképzés (ORM): A Doctrine segítségével a PHP objektumokat lehet leképezni és tárolni az adatbázisban. Ez azt jelenti, hogy az adatbázistáblákat PHP objektumokkal lehet modellezni, és az ezekre vonatkozó műveleteket objektumorientált módon lehet végrehajtani.
- Entitások és Repository-k: A Doctrine az adatbázis táblákat entitásokként kezeli, amelyeknek megfelelően reprezentálják az adatbázis struktúráját. Minden entitáshoz tartozik egy Repository osztály, amely különböző adatbázis műveleteket biztosít az entitásokkal kapcsolatban.
- Adatbázis-független: A Doctrine támogatja különböző adatbázisrendszereket, beleértve az SQL-alapú adatbázisokat, mint például a MySQL, PostgreSQL és SQLite, valamint NoSQL adatbázisokat is.
- Query Builder: A Doctrine lehetővé teszi a SQL lekérdezések objektumorientált módon történő összeállítását a Query Builder segítségével, ami segít elkerülni a hagyományos, karakterláncokon alapuló SQL lekérdezésekkel járó hibákat és biztonsági problémákat.
- Teljesítmény optimalizáció: A Doctrine számos funkciót és lehetőséget kínál a teljesítmény optimalizálására, beleértve az adatbázis indexelést, a lazy loading és eager loading lehetőségeket, valamint a caching funkciókat.

Endpointok

Témák/Topics:

- GET /api/v1/topics: Témák listázása.
- POST /api/v1/topics: Új téma létrehozása.
- PATCH /api/v1/topics/{topicId}: Téma frissítése azonosító alapján.
- DELETE /api/v1/topics/{topicId}: Téma törlése azonosító alapján.

Hozzászólások/Comments:

- GET /api/v1/topics/{topicId}/comments: Hozzászólások listázása egy témához.
- POST /api/v1/topics/{topicId}/comments: Új hozzászólás létrehozása egy témához.
- DELETE /api/v1/topics/{topicId}/comments/{id}: Egy komment törlése az azonosító alapján. Csak a komment tulajdonosa törölheti.
- PATH /api/v1/topics/{topicId}/comments/{id}: Egy komment módosítása az azonosító alapján. Csak a komment tulajdonosa törölheti.

Felhasználók/Users:

- POST /api/v1/users: Új felhasználó létrehozása.
- GET /api/v1/users/{id}: Felhasználó lekérése azonosító alapján.
- PATCH /api/v1/users/{id}: Felhasználó frissítése azonosító alapján.
- DELETE /api/v1/users/{id}: Felhasználó törlése azonosító alapján.

Felhasználó Bejelentkezés:

- POST /api/v1/login: Felhasználó JWT Token létrehozása.

Felhasználó JWT token frissítése:

- POST /api/v1/token/refresh: Felhasználó JWT Token frissítése.

Endpointok részletezése

Témák/Topics:

GET /api/v1/topics: Témák listázása:

- Query paraméterek:
  - page: int, oldalszám, default: 1
  - limit: int, maximális témák száma oldalanként, maximum: 100, default: 10
- Responses:
  - 200: visszaadja a témákat a megadott paraméterek szerint.
    Példa:


```
{
    "message": "",
    "errors": [],
    "result": [
        {
            "id": 1,
            "name": "Első topic 2",
            "description": "Az első topic leírásom nézzzük csak meg 2",
            "created_at": "2024-03-20T19:58:06+00:00",
            "user_id": 1,
            "username": "JBZ",
            "comment_count": 7
        }
    ]
}
```


  - 403: Hozzáférés megtagadva.
    Példa:

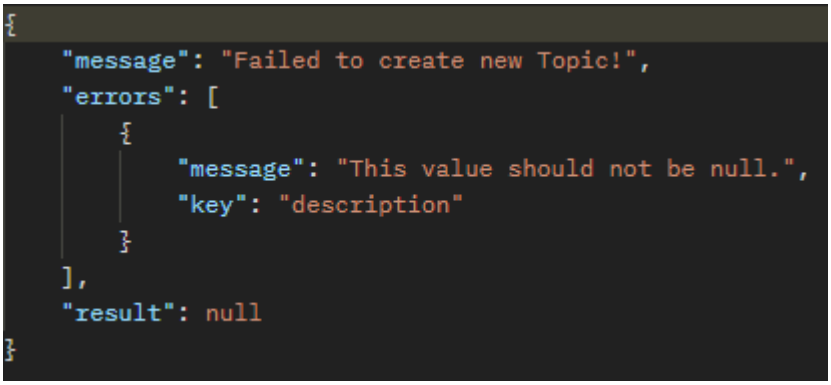

```
{
    "code": 401,
    "message": "JWT Token not found"
}
```


POST /api/v1/topics: Új téma létrehozása:

- Body (JSON):
  - name: string, téma neve
  - description: string, téma leírása
- Responses:
  - 200: Sikeres téma létrehozás
    Példa:

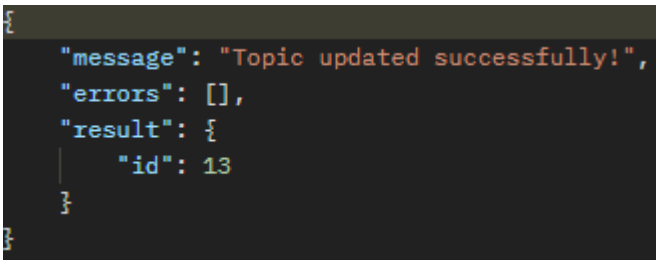

```
{
    "message": "Topic created successfully!",
    "errors": [],
    "result": {
        "id": 13
    }
}
```

- 400: Hibás kérés
  Példa:

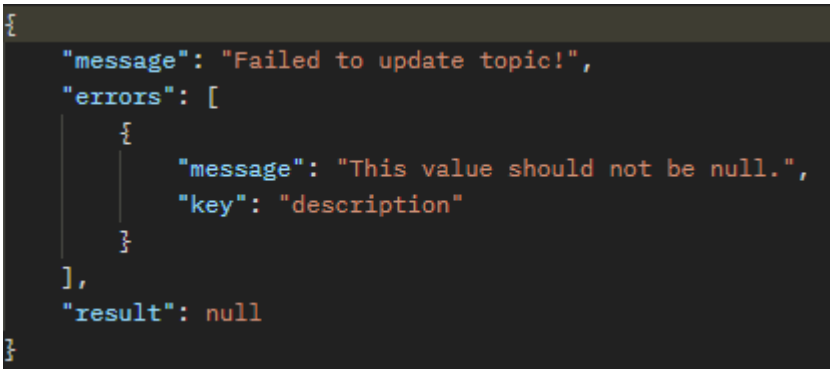


PATCH /api/v1/topics/{topicId}: Téma frissítése azonosító alapján:

- URL paraméter:
  - topicId: int, téma azonosítója
- Body (JSON):
  - name: string, téma neve
  - description: string, téma leírása
- Responses:
  - 200: Sikeres téma módosítás
    Példa:



  - 400: Hibás kérés
    Példa:



DELETE /api/v1/topics/{topicId}: Téma törlése azonosító alapján:

- URL paraméter:
  - topicId: int, téma azonosítója
- Responses:
  - 200: Sikeres törlés.
    Példa:

```
{
    "message": "Topic deleted successfully!",
    "errors": [],
    "result": null
}
```

  - 400: Hibás kérés:
    Példa:

```
{
    "message": "Topic not found!",
    "errors": {
        "topicId": "Invalid topic id!"
    },
    "result": null
}
```

Hozzászólások/Comments:

GET /api/v1/topics/{topicId}/comments: Hozzászólások listázása egy témához.

- URL paraméter:
  - topicId: int, téma azonosítója
- Responses:
  - 200: Sikeres kérés
    Példa:


```
{
    "message": "",
    "errors": [],
    "result": [
        {
            "id": 1,
            "user_id": 1,
            "user_name": "Erdős Attila",
            "message": "testtt",
            "created_at": "2024-04-23T14:40:01+00:00",
            "updated_at": "2024-04-23T14:40:01+00:00"
        }
    ]
}
```


POST /api/v1/topics/{topicId}/comments: Új hozzászólás létrehozása egy témához.

- URL paraméter:
  - topicId: int, téma azonosítója
- Body (JSON):
  - orignal: string, hozzászolás
- Responses:
  - 200: Sikeres kérés
    Példa:


```
{
    "message": "Comment created successfully!",
    "errors": [],
    "result": {
        "id": 7
    }
}
```

- 400: Hibás kérés
  Példa:

```
{
    "message": "Failed to create new comment!",
    "errors": [
        {
            "message": "This value should not be null.",
            "key": "original"
        }
    ],
    "result": null
}
```

-

DELETE /api/v1/topics/{topicId}/comments/{id}: Egy komment törlése az azonosító alapján. Csak a komment tulajdonosa törölheti.

- URL paraméter:
  - topicId: int, téma azonosítója
  - Id: int, hozzászólás azonosítója
- Responses:
  - 200: Sikeres törlés.
    Példa:

```
{
    "message": "Comment deleted successfully!",
    "errors": [],
    "result": null
}
```

  - 400: Hibás kérés
    példa:

```
{
    "message": "Comment not found!",
    "errors": {
        "topicId": "Invalid comment id!"
    },
    "result": null
}
```

PATH /api/v1/topics/{topicId}/comments/{id}: Egy komment módosítása az azonosító alapján. Csak a komment tulajdonosa törölheti.

- URL paraméter:
  - topicId: int, téma azonosítója
  - Id: int, hozzászólás azonosítója
- Body (JSON):
  - orignal: string, hozzászolás
- Responses:
  - 200: Sikeres kérés
    Példa:

```
{
    "message": "Comment updated successfully!",
    "errors": [],
    "result": {
        "id": 7
    }
}
```

- 400: Hibás Kérés
  Példa:


```
{
    "message": "Comment not found!",
    "errors": {
        "topicId": "Invalid comment id!"
    },
    "result": null
}
```

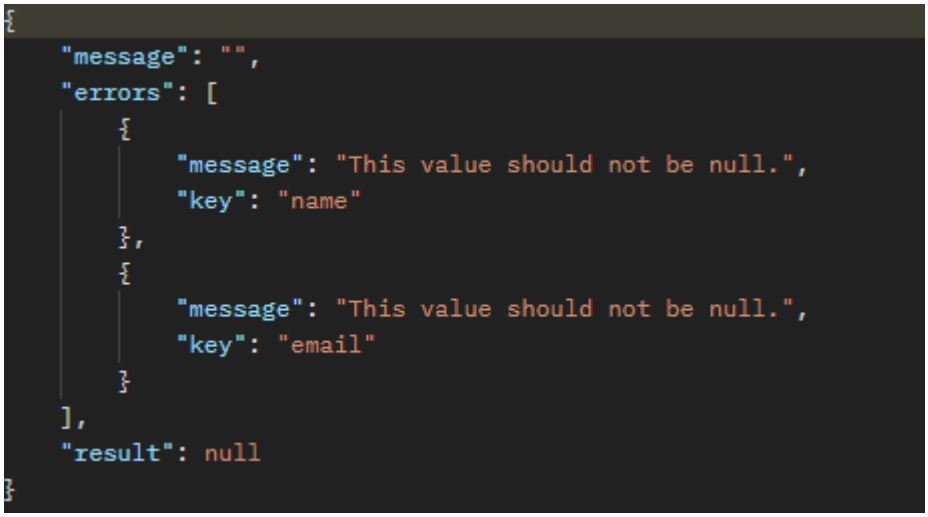

Felhasználók/Users:

POST /api/v1/users: Új felhasználó létrehozása.

- Body (JSON):
  - name: stirng, név
  - email: string, email cím
  - password: string, jelszó
- Responses:
  - 200: Sikeres kérés
    Példa:


```
{
    "message": "User created successfully",
    "user": {
        "id": 4,
        "name": "test",
        "email": "test@test.com",
        "status": 1,
        "createdAt": "2024-04-23T16:39:47+00:00",
        "updatedAt": "2024-04-23T16:39:47+00:00",
        "topics": [],
        "comments": [],
        "password": "d74008962777b1c27dac077194d5e2d883d198b0f3ac94168e907fc7178e2716",
        "roles": [
            "ROLE_USER"
        ],
        "userIdentifier": "test@test.com"
    }
}
```

- 400: Hibás kérés
  Példa:



GET /api/v1/users/{id}: Felhasználó lekérése azonosító alapján.

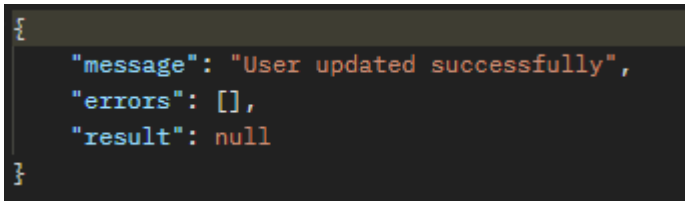
- URL paraméter:
  - id: int, felhasználó azonosítója
- Responses:
  - 200: Sikeres kérés
    Példa:

  - 404: Nem található

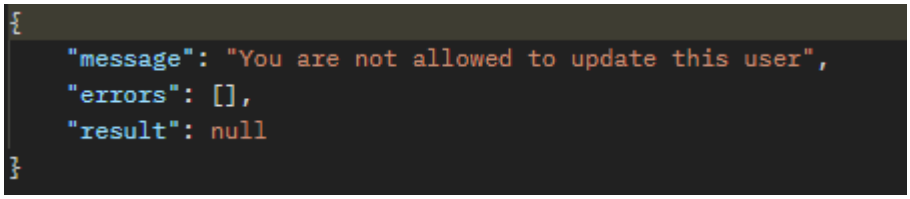
PATCH /api/v1/users/{id}: Felhasználó frissítése azonosító alapján.

- URL paraméter:
  - id: int, felhasználó azonosítója
- Body (JSON):
  - name: stirng, név, opcionális
  - email: string, email cím, opcionális
  - password: string, jelszó, opcionális
- Responses:

- 200: Sikeres kérés
  Példa:



- 403: Hozzáférés megtagadva
  Példa



DELETE /api/v1/users/{id}: Felhasználó törlése azonosító alapján.

- URL paraméter:
  - id: int, felhasználó azonosítója
- Responses:
  - 200
    Példa:

```
{
    "message": "User deleted successfully",
    "errors": [],
    "result": null
}
```

  - 403: Hozzáférés megtagadva
    Példa:

```
{
    "message": "You are not allowed to delete this user",
    "errors": [],
    "result": null
}
```

Felhasználó Bejelentkezés:

POST /api/v1/login: Felhasználó JWT Token létrehozása.

- Body (JSON):
  - username: string, email cím
  - password: string, jelszó
- Responses:
  - 200: Sikeres kérés
    Példa:


```
{
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.
        eyJpYXQiOjE3MTM4OTAwMTMsImV4cCI6MTcxMzg5MzYxMywicm9sZXMiOlsiUk9MRV9VU0VSIl0sInVzZXJuYW1lIjoiZWF0dHkxOTk2QGdtYWlsLmNvbSIsIm5hbWUiOiJFcmTFkXMgQXR0aWxhIiwidXNlcl9pZCI6MX0.
        AjJtQ9l7bwnJ4q9JVhStLL3log7D3GCbfHqj1xpXBey4G4rBQNEZktJMyY1M_ohItSX8y5v9LB7aydtuIE9OrvV5iLSTchcFLjw5sU34dYeuVkISKCN3AdGY5bZBtRFB8aJojymGirCuzdkJVwsnrzTgGgj
        3JbEiz-ejkopbwUw7mWuXCS17poZPBWpQdc86pWUZpZleMV6ToM9a18BsHHGisYpu660ZAshsnGXM0dVpjvt5BHQNgdNTWpyjVQhv0X-vSISA5DjFH4NJSlQUG8W1PVEDIQ1gwCm5pV6EWgaJQSJshNTd_T
        YXJl6Lhnpv_jnUk8-dcg-5ArV5e4ZLNQ",
    "refresh_token": "b1bdbaebc60624a294ac08a0319071bfa9baa7df64915987b0eb10d9b35748655e5e552d68715137401673b52dab781e769b7f8496cfa82ba04767ca44bbb024"
}
```


POST /api/v1/token/refresh: Felhasználó JWT Token frissítése.

- Body (JSON):
  - refresh_token: string, loginkor visszajött refresh_token.
- Responses:
  - 200: Sikeres kérés
    Példa:

```
{
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.
        eyJpYXQiOjE3MTM4OTA5MzEsImV4cCI6MTcxMzg5NDUzMSwicm9sZXMiOlsiUk9MRV9VU0VSIl0sInVzZXJuYW1lIjoidGVzdEB0ZXN0LmNvbSIsIm5hbWUiOiJ0ZXN0IiwidXNlcl9pZCI6NH0.
        IHoxJz7x5Dd6xEmq8KoHrg7NGadCOxzV6q31paos42GCk8rEtiiIje8FNCKhlH4SYZrhp6-_YdqSNrfa0-C5ZTzLrwzygtaB8o36WFrcCwkR0NLgTeozID3fg8HuFoAWSPNOBG_5spwMlc2WQZBJzJ1cFKx
        0YtYfrFYfFRlti61EprsthnsTDBZZyCRfnukRRDzAaANLR8hu4HhyDoX3aMaeTpcQRBAY6oh_WixNmUsIcJctG2qVa6mDtjxr0Ud0QIn6LPDYAiFlp_tOa3WTTgto93coK8Sze6EBMipRA92s_oaLfEtoFh
        RSkmICZiDIM5mQYKm-DJk2BpsZjeHlyQ",
    "refresh_token": "5926a44ab4539c856ca57959c5ce2cff0add16b2a86c9242d1ecfa93fe3e815c47aa6b45a09f90cdf73ca6ff29ed1faa20ad5ca40d9d0b61c05a49c4b01007e1"
}
```