**VKSZ API/Admin client Dokumentáció**

**Tartalomjegyzék**

[Telepítés: 3](#_Toc83894297)

[API(Szerver): 3](#_Toc83894298)

[Api/.env fájl tartalma: 3](#_Toc83894299)

[Valamilyen Mysql szerver telepítése, beállítása 3](#_Toc83894300)

[API-hoz szükséges social App-ok létrehozása, push üzenetkhez Firebase projekt létrehozása 3](#_Toc83894301)

[I.Firebase(Push üzenetekért): 3](#_Toc83894302)

[1.Firebase projekt létrehozása: 3](#_Toc83894303)

[2.Android alkalmazásba a projeaktunk integrálása: 3](#_Toc83894304)

[3.Firebase projekt hozzáadása az API szerverhez: 4](#_Toc83894305)

[II.Facebook app: 5](#_Toc83894306)

[1.Facebook App projekt létrehozása: 5](#_Toc83894307)

[2.Facebook App beállítása hogy engedélyezze a call-okat 5](#_Toc83894308)

[3.API URIs hozzáadása az App-hoz hogy engedélyezzük a facebook logint 5](#_Toc83894309)

[4.Engedélyezzük hogy megszerezzük a bejelentkező publikus profilját,és email-jét 5](#_Toc83894310)

[5.Key, App Secret megszerzése 5](#_Toc83894311)

[6.Teszteléshez, felhasználók hozzáadása 5](#_Toc83894312)

[III.Twitter app: 7](#_Toc83894313)

[1.Twitter App projekt létrehozása, key és secret megszerzése 7](#_Toc83894314)

[2.Twitter App callback urls beállítása 7](#_Toc83894315)

[3.Website URL beállítása 7](#_Toc83894316)

[IV.Google app: 8](#_Toc83894317)

[1.Google projekt létrehozása, és elérhetőségek, engedélyek beállítása 8](#_Toc83894318)

[2.Google project URI beállítás, Client ID és Client Secret megszerzése 8](#_Toc83894319)

[V. API szerver elindítása: 9](#_Toc83894320)

[1.Szerver indítása 9](#_Toc83894321)

[2.Első futtatásnál az adatbázis feltöltése táblákkal, pár teszt adattal 9](#_Toc83894322)

[Client(Admin felület): 10](#_Toc83894323)

[client/.env fájl tartalma: 10](#_Toc83894324)

[Client elindítása 10](#_Toc83894325)

[Fájlok: 11](#_Toc83894326)

[API(Szerver): 11](#_Toc83894327)

[Routes 12](#_Toc83894328)

[admin.js 12](#_Toc83894329)

[auth.js 17](#_Toc83894330)

[user.js 22](#_Toc83894331)

[Client: 25](#_Toc83894332)

[Használt fájlok: 25](#_Toc83894333)

Telepítés:

API(Szerver):

|  |  |
| --- | --- |
| **Dependencies feltelepítése:**   |  | | --- | | Api/ mappában npm install | |
| Api/.env fájl tartalma:   |  | | --- | | VARIABLE\_NAME=Description(Fejlesztésnél használt értékek)  PORT= Api Szerver Portja(3001) SECRET\_KEY=JWT-hez titkosító kulcs(lamaosecretkey) CLIENT\_URL= <http://client_ip:client_port> (http://localhost:3011) API\_URL=http://api\_ip:api\_port (http://localhost:3001) EMAIL\_HOST=Email szerver(smtp.ethereal.email) EMAIL\_PORT=Email szerver portja(587) [EMAIL\_USERNAME=](mailto:EMAIL_USERNAME=isabella.daugherty58@ethereal.email)Email( [isabella.daugherty58@ethereal.email](mailto:isabella.daugherty58@ethereal.email) )  EMAIL\_PASSWORD=Email password(bCMKFutjxc9MVXxe77) DEFAULT\_AVATAR\_LOCATION=./src/media/blank.png DB\_HOST=Mysql host(localhost) DB\_USER=Mysql username(root) DB\_PASSWORD=Mysql user password() DB\_DATABASE=Mysql database(vksz) FACEBOOK\_APP\_ID= II.Pontban(137181788589630) FACEBOOK\_APP\_SECRET= II.Pontban(e81c2806f11912d1bf3d38b3a17e1bea) TWITTER\_API\_KEY= III.Pontban(bwQdHUKJJaMSCUOkSxfu3iTNL) TWITTER\_API\_KEY\_SECRET= III.Pontban(ZCviRbz3NvemCLQ8fe9Xr0GccBTjtq01ZiSKVt7XBOeDxgX1Vn) GOOGLE\_CLIENT\_ID= IV.Pontban(270328200769-7kmjk88qvufnjl94pupm77g9holefd0n.apps.googleusercontent.com) GOOGLE\_CLIENT\_SECRET=IV.Pontban(y0J6A8ACR62HIfeXegoES4gj) | |
| Valamilyen Mysql szerver telepítése, beállítása |

API-hoz szükséges social App-ok létrehozása, push üzenetkhez Firebase projekt létrehozása

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| I.Firebase(Push üzenetekért):   |  |  | | --- | --- | | 1.Firebase projekt létrehozása:   |  | | --- | | Firebase → Go to console → Add Project → Nevezd el a projektek → Enable Google Analytics → Continue → Meglévő, vagy új analytic account kiválasztása → Create Project | | | 2.Android alkalmazásba a projektunk integrálása:   |  | | --- | | Project settings → Add Android App → Írjunk be egy package nevet, App nevet → Register app → töltsd le a google-services.json -t → Mobil alkalmazás app/ mappájába kell tenni → FCM integrálása az alkalmazásba <https://firebase.google.com/docs/cloud-messaging/android/client> | | | 3.Firebase projekt hozzáadása az API szerverhez:   |  | | --- | | Project settings → Service accounts → Firebase Admin SDK → Generate new private key → letöltött JSON-t az api/src/ mappába kell tenni → az api/src/routes/admin.js fájlban a var serviceAccount = require('../pushup-test-vksz-firebase-adminsdk-hy1hr-b70f68fdc3.json'); írd át →  var serviceAccount = require('../A\_Letöltött\_JSON\_Neve.json'); -ra | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| II.Facebook app:   |  |  | | --- | --- | | 1.Facebook App projekt létrehozása:   |  | | --- | | Facebook Developer Dashboard → Create App → Consumer(Vagy ami kell) → App elnevezése, contact email megadása → Facebook login kártya → Setup | | | 2.Facebook App beállítása hogy engedélyezze a call-okat   |  | | --- | | Settings → Basic → App Domains kitöltése(localhost) → Privacy Policy, User Data Deletion url megadása Category → Kategória kiválasztása(Business and pages)App purpose → App purpose kiválasztása(Yourself or your own business)Oldal alján → Add platform → Website → Site URL(https://localhost:8443/) website os teszteléshez, majd az android os implementációhoz Android platform ot is addolni kell | | | 3.API URIs hozzáadása az App-hoz hogy engedélyezzük a facebook logint   |  | | --- | | Facebook Login → Settings → Valid OAuth Redirect URIs →  https://localhost:8443/  https://localhost:8443/auth/facebook  https://localhost:8443/auth/facebook/callback hozzáadása | | | 4.Engedélyezzük hogy megszerezzük a bejelentkező publikus profilját,és email-jét   |  | | --- | | App Review → Permissions and Features → public profile → Get advanced access email → Get advanced access | | | 5.Key, App Secret megszerzése   |  | | --- | | Key/Secret: Settings → Basic → App ID = app/.env-ben FACEBOOK\_APP\_ID  App Secret = app/.env-ben FACEBOOK\_APP\_SECRET | | | 6.Teszteléshez, felhasználók hozzáadása   |  | | --- | | Roles→ Roles → Testers → Add Testers | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| III.Twitter app:   |  |  | | --- | --- | | 1.Twitter App projekt létrehozása, key és secret megszerzése   |  | | --- | | Twitter Developer Dashboard → Create App → App elnevezése →  API Key = app/.env-ben TWITTER\_API\_KEY API Key Secret = app/.env-ben TWITTER\_API\_KEY\_SECRET | | | 2.Twitter App callback urls beállítása   |  | | --- | | App → Authentication Settings → Edit →  Enable 3-legged OAuth  Enable Request email address from users Callback URLs: <https://127.0.0.1:8443/auth/twitter/callback> <https://127.0.0.1:8443/auth/twitter> <https://127.0.0.1:8443/api/auth/twitter/token> <https://127.0.0.1:8443> | | | 3.Website URL beállítása   |  | | --- | | Website URL: https://www.localhost:8443/ | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IV.Google app:   |  |  | | --- | --- | | 1.Google projekt létrehozása, és elérhetőségek, engedélyek beállítása   |  | | --- | | Google Cloud Platform → Create Project(vagy a Firebase projekt is jó) → APIs & Services→ OAuth consent screen → External → Create → App name, support email, developer contant megadása → Save and continue →Add or remove scopes →   .../auth/userinfo.email  .../auth/userinfo.profile scope kijelölése és update/save → Continue → Test usereket hozzá adhatunk → Save and continue | | | 2.Google project URI beállítás, Client ID és Client Secret megszerzése   |  | | --- | | Credentials → Create Credentials → OAuth Client ID → App Type: Web Application (Majd androidos implementálásnál android kell) → Elnevezés → Authorized Redirect URIs:   <https://localhost:8443/auth/google/callback>  <https://localhost:8443/auth/google>  <https://localhost:8443> Create →  Your Client ID = api/.env-ben GOOGLE\_CLIENT\_ID  Your Client Secret = api/.env-ben GOOGLE\_CLIENT\_SECRET | | |

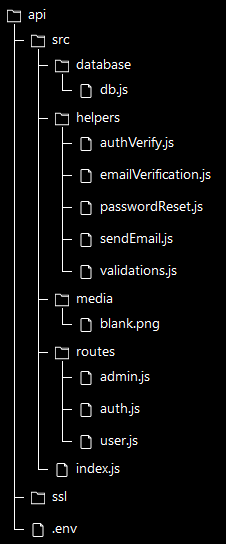
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| V. API szerver elindítása:   |  |  | | --- | --- | | 1.Szerver indítása   |  | | --- | | api/ mappából nodemon(test) vagy node src/index.js | | | 2.Első futtatásnál az adatbázis feltöltése táblákkal, pár teszt adattal   |  | | --- | | Ha fut a mysql szerver és az environment variables-k is bevannak állítva első futtatásnál a <http://Ip_cím:PORT/createdb> oldal megtekintésével a beállított adatbázisba létrehozza a táblákat, és feltölti a notif\_type, news\_services táblát pár teszt értékkel. | | |

Client(Admin felület):

|  |  |
| --- | --- |
| **Dependencies feltelepítése:**   |  | | --- | | Client/ mappában npm install | |
| client/.env fájl tartalma:   |  | | --- | | VARIABLE\_NAME=Description(Fejlesztésnél használt értékek)  PORT=Milyen porton fusson a client(3011) REACT\_APP\_BASE\_LAYOUT\_CONFIG\_KEY='metronic-react-demo100' REACT\_APP\_API\_URL=Az API szervernek base URL -je ([http://72.99.421.](http://80.99.143.2)8:3001 (Saját ip cimem)) REACT\_APP\_VERSION=v8.0.19 REACT\_APP\_THEME\_NAME=Metronic REACT\_APP\_BOOTSTRAP\_DOCS\_LINK=https://getbootstrap.com/docs/5.0 REACT\_APP\_SASS\_PATH=src/\_metronic/assets/sass/core/components REACT\_APP\_SASS\_VARIABLES\_PATH=src/\_metronic/assets/sass/core/components/\_variables.scss PUBLIC\_URL= Client public URL je (<http://72.99.421.>8 (Saját ip cimem)) //A következő változókat nem használjuk de nem töröltem ki őket hátha elront valamit REACT\_APP\_PURCHASE\_URL=https://themeforest.net/item/metronic-responsive-admin-dashboard-template/4021469 REACT\_APP\_PREVIEW\_URL=https://preview.keenthemes.com/metronic8/react/demo1/ REACT\_APP\_PREVIEW\_REACT\_URL=https://preview.keenthemes.com/metronic8/react REACT\_APP\_PREVIEW\_DOCS\_URL=https://preview.keenthemes.com/metronic8/react/demo1-docs | |
| Client elindítása   |  | | --- | | client/ -mappában npm start | |

Fájlok:

API(Szerver):



|  |  |
| --- | --- |
| **Fájlnév/function név** | **Szerep** |
| **db.js** | Létrehozza a kapcsolatot az adatbázissal |
| **authVerify.js/verify(req, res, next)** | A bejövő API call oknak ellenőrzni a headerjében lévő Authorization JWT tokent hogy érvényes-e, és blacklistelve van-e a küldő |
| **emailVerification.js/sendEmailVerification(email):** | Elküld egy verification e-mailt a megadott címre |
| **passwordReset.js/passwordReset(email)** | Elküld egy password reset e-mailt a megadott címre |
| **sendEmail.js/sendEmail(email, subject, message)** | Elküld egy e-mailt a megadott címre, a megadott tárgyal, és a megadott üzenettel |
| **validations.js** | JOI validations objekteket tartalmaz amikkel lehet ellenőrizni az eggyes request body tartalmát |
| **media/blank.png** | Az alap avatar a vksz alapon történő regisztráció esetén |

Routes

|  |
| --- |
| admin.js |
| |  | | --- | | **router.post(‘/get-user-by-id’, verify, (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{  Authorization:Bearer JWT\_TOKEN  }  Body:{  {  user\_id:int  }  } | | **Mit ad vissza:** | user:{  id(int),  username(string),  email(string),  firstname(string),  lastname(string),  phone(string),  roles(string array),  pic(medium blob),  zip(string),  city(string),  house\_number(string),  street(string),  communication(array)[{  name(string)  email(bool)  sms(bool)  phone(bool)  service\_id(int)  }]  } | | **Mit csinál:** | Megkeresi a request body-ban megadott id-jú user-t, ha van vissza adja az adatait, ha nincs errort ad | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **router.get(‘/get-users’, verify, (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{  Authorization:Bearer JWT\_TOKEN  }  Body:{} | | **Mit ad vissza:** | users(array):[{  id(int),  email(string),  firstname(string),  lastname(string),  pic(medium blob),  phone(string),  provider(string),  blacklisted(bool),  zip(string),  city(string),  house\_number(string),  street(string),  }] | | **Mit csinál:** | Visszadja az összes user-t a user táblából | | |
| |  | | --- | | **router.post(‘/block-user’, verify, (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{  Authorization:Bearer JWT\_TOKEN  }  Body:{  {  user\_id:int  email:string  }  } | | **Mit ad vissza:** | result(bool) | | **Mit csinál:** | Beteszi a blacklist táblába az adott ID-jú user-t, és a megadott email alapján(ami a user\_id hoz kéne tartoznia) megkeresi a többi user-t ami arról az email-ről regisztrált(Social loginon keresztül) és őket is beteszi a blacklist táblába. Vagy ha már bent van a a blacklist táblában a user, kitörli. | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **router.post(‘/change/password’, verify, (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{  Authorization:Bearer JWT\_TOKEN  }  Body:{  {  new\_pass:string,  new\_pass2:string,  user\_id:int  }  } | | **Mit ad vissza:** | result(bool) | | **Mit csinál:** | Megnézi hogy a megadott 2 jelszó egyezik, és megváltoztatja arra a body-ban megadott user\_id-jú user-nek a jelszavát | | |
| |  | | --- | | **router.post(‘/change/email’, verify, (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{  Authorization:Bearer JWT\_TOKEN  }  Body:{  {  email:string,  user\_id:int  }  } | | **Mit ad vissza:** | result(bool) | | **Mit csinál:** | Megváltoztatja a megadott user\_id-jú user-nek az email-jét ha a user-nek a providere vksz | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **router.post(‘/send/users/notification’, verify, (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{  Authorization:Bearer JWT\_TOKEN  }  Body:{  {  service\_id:int,  notif\_id:int,  title:string,  message:string  }  } | | **Mit ad vissza:** | result(bool) | | **Mit csinál:** | Megadott notif\_id-val megeggyező módon üzenetet küld minden user-nek aki fel van iratkozva a megadott service\_id-jú szolgáltatásra a megadott title, message-el | | |
| |  | | --- | | **router.post(‘/send/user/notification’, verify, (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{  Authorization:Bearer JWT\_TOKEN  }  Body:{  {  user\_id:int,  notif\_id:int,  title:string,  message:string  }  } | | **Mit ad vissza:** | result(bool) | | **Mit csinál:** | Megadott notif\_id-val megeggyező módon üzenetet küld a megadott user\_id-jú user nek a megadott title, message-el | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **router.get(‘/get-news-notifs’, verify, (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{  Authorization:Bearer JWT\_TOKEN  }  Body:{} | | **Mit ad vissza:** | news\_notifs:{  notif\_types(array)[{notif\_id, notif\_name}]  news\_services(array)[{service\_id, service\_name}]  } | | **Mit csinál:** | Vissza adja milyen üzenet küldési módszerek vannak(notif\_type tábla), és milyen szolgáltatások(news\_services tábla) vannak | | |

|  |
| --- |
| auth.js |
| |  | | --- | | **router.post('/login', (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{}  Body:{  {  email:string,  password:string  }  } | | **Mit ad vissza:** | accessToken(JWT object) | | **Mit csinál:** | Megnézi hogy megvan-e a megadott email az adatbázisban és megegyezik-e a jelszó és a provider = vksz.  Egy JWT-t ad vissza amiben benne van az email, role, user\_id, provider | | |
| |  | | --- | | **router.post('/register', (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{}  Body:{  {  email:string,  lastname:string,  firstname:string,  password:string,  zip:string,  city:string,  street:string,  house\_number:string,  phone:string,  device\_token:string  }  } | | **Mit ad vissza:** | result(bool) | | **Mit csinál:** | Ellenőrzi az adatok érvényességét, és ha nincs ilyen user az adatbázisban vksz providerrel, akkor beteszi az adatbázisba, és küld egy megerősítő email-t a megadott email-re | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **router.get('/confirmation/:token', (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{}  Body:{} | | **Mit ad vissza:** | Redirectel a client weboldalra | | **Mit csinál:** | Ellenőrzi a url ben lévő JWT-t és frissíti a JWT alapján a user-nek a confirmed értékét, hogy betudjon lépni. | | |
| |  | | --- | | **router.post('/confirmation', (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{}  Body:{  {  email:string  }  } | | **Mit ad vissza:** | result(bool) | | **Mit csinál:** | Megadott email-re küld egy verification emailt. | | |
| |  | | --- | | **router.get('/reset-password/:token', (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{}  Body:{} | | **Mit ad vissza:** | result(bool) | | **Mit csinál:** | Ellenőrzi a url ben lévő JWT-t és general egy új jelszót a JWT-ben lévő user-nek és küld neki egy email-t | | |
| |  | | --- | | **router.post('/reset-password', (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{}  Body:{  {  email:string  }  } | | **Mit ad vissza:** | result(bool) | | **Mit csinál:** | Ellenőrzi az adatok érvényességét, és küld egy password reset emailt. | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **router.get('/get-user', verify, (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{  Authorization:Bearer JWT\_TOKEN  }  Body:{} | | **Mit ad vissza:** | user:{  id(int),  username(string),  email(string),  firstname(string),  lastname(string),  phone(string),  roles(string array),  provider(string),  pic(medium blob),  zip(string),  city(string),  house\_number(string),  street(string),  communication(array)[{  name(string)  email(bool)  sms(bool)  phone(bool)  service\_id(int)  }]  } | | **Mit csinál:** | Ellenőrzi a JWT, adatok érvényességét és vissza adja a JWT tokenben lévő id-jú user adatait | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **router.get('/facebook/token', (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{  Authorization:Bearer facebook\_loginnál\_kapott\_accessToken  }  Body:{  {  device\_token:string  }  } | | **Mit ad vissza:** | accessToken(JWT object) | | **Mit csinál:** | Ellenőrzi a facebook accessToken érvényességét, ha nincs még ilyen user aki facebookal lépett be volna, lekér pár adatot a user facebook public profile-járól és beteszi az adatbázisba, ha már van ilyen JWT-t ad vissza.  A facebook bejelentkezést a https://localhost:8443/auth/facebook -al az index.js ben lévő route al lehet tesztelni JSON-ben kapjuk meg az adatokat amik kellenek a szerveren való ellenőrzéshez  accessToken = accessToken | | |
| |  | | --- | | **router.get('/twitter/token', (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Paraméterek:  oauth\_token: twitter\_loginnál\_token  oauth\_token\_secret: twitter\_loginnál\_tokenSecret  user\_id: twitter\_loginnál\_profil\_id  Body:{  {  device\_token:string  }  } | | **Mit ad vissza:** | accessToken(JWT object) | | **Mit csinál:** | Ellenőrzi a twitter token, token\_secret, user\_id érvényességét, ha nincs még ilyen user aki twitterrel lépett be volna, lekér pár adatot a user twitter public profile-járól és beteszi az adatbázisba, ha már van ilyen JWT-t ad vissza.  A twitter bejelentkezést a https://127.0.0.1:8443/auth/twitter -al az index.js ben lévő route al lehet tesztelni, JSON-ben kapjuk meg az adatokat amik kellenek a szerveren való ellenőrzéshez  oauth\_token = token,  oauth\_token\_secret = tokenSecret,  user\_id = id | | |
| |  | | --- | | **router.get('/google/token', (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{  access\_token:GOOGLE\_accessToken  }  Body:{  {  device\_token:string  }  } | | **Mit ad vissza:** | accessToken(JWT object) | | **Mit csinál:** | Ellenőrzi a google accessToken érvényességét, ha nincs még ilyen user aki google-val lépett be volna, lekér pár adatot a user google profile-járól és beteszi az adatbázisba, ha már van ilyen JWT-t ad vissza.  A google bejelentkezést a https://localhost:8443/auth/google -al az index.js ben lévő route al lehet tesztelni, JSON-ben kapjuk meg az adatokat amik kellenek a szerveren való ellenőrzéshez  accessToken = accessToken | | |

|  |
| --- |
| user.js |
| |  | | --- | | **router.post('/change/details', verify, (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{  Authorization:Bearer JWT\_TOKEN  }  Body:{  {  user\_id:int,  lastname:string,  firstname:string,  zip:string,  city:string,  street:string,  house\_number:string,  phone:string  }  } | | **Mit ad vissza:** | result(bool) | | **Mit csinál:** | Ellenőrzi a JWT, adatok érvényességét megváltoztatja a body-ban megadott user\_id értéke alapján a user adatait | | |
| |  | | --- | | **router.post('/change/avatar', verify, (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{  Authorization:Bearer JWT\_TOKEN  }  Body:{  {  avatar:string,  user\_id:int  }  } | | **Mit ad vissza:** | result(bool) | | **Mit csinál:** | Ellenőrzi a JWT, adatok érvényességét megváltoztatja a body-ban megadott user\_id értéke alapján frissíti az avatar-t az adatbázisban. | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **router.put('/change/notifications', verify, (req,res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{  Authorization:Bearer JWT\_TOKEN  }  Body:{  {  communication(array)[{  name(string)  email(bool)  sms(bool)  phone(bool)  service\_id(int)  }],  user\_id:int  }  } | | **Mit ad vissza:** | result(bool) | | **Mit csinál:** | Ellenőrzi a JWT, adatok érvényességét megváltoztatja a body-ban megadott user\_id értéke alapján frissíti a user\_notifs-ban lévő értékeket. | | |
| |  | | --- | | **router.post('/change/password', verify, (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{  Authorization:Bearer JWT\_TOKEN  }  Body:{  {  current\_pass:string,  new\_pass:string,  new\_pass2:string,  user\_id:int  }  } | | **Mit ad vissza:** | result(bool) | | **Mit csinál:** | Ellenőrzi a JWT, adatok érvényességét megváltoztatja a body-ban megadott user\_id értéke alapján a user jelszavát | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **router.post('/change/email', verify, (req, res)):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{  Authorization:Bearer JWT\_TOKEN  }  Body:{  {  email:string,  current\_pass:string,  user\_id:int  }  } | | **Mit ad vissza:** | result(bool) | | **Mit csinál:** | Ellenőrzi a JWT, adatok érvényességét, megváltoztatja az adott user-nek az email-ját ha még nincs ilyen email az adatbázisban vksz providerrel | | |
| |  | | --- | | **router.post('/update/device-token', verify, (req, res):** | | |  |  | | --- | --- | | **Mi kell a requestbe:** | Header:{  Authorization:Bearer JWT\_TOKEN  }  Body:{  {  email:string,  user\_id:int,  device\_token:string  }  } | | **Mit ad vissza:** | result(bool) | | **Mit csinál:** | Ellenőrzi a JWT, adatok érvényességét, megváltoztatja az adott user-nek a device\_tokenjét. | | |

Client:

Használt fájlok:

|  |  |
| --- | --- |
| **Létrehozott fájlok** | |
| **Fájlnév** | **Szerep** |
| **UsersTableWidget.tsx** | /admin-dashboard -URL-en az adatbázisban lévő összes felhasználót jeleníti meg, email re kattintva megnyit egy részletesebb oldalt ahol a felhasználónak az adatait tudja változtatni, notificationt tud küldeni |
| **AdminBoard.tsx** | /admin-dashboard -URL-en a UsersTableWidget.tsx-et jeleníti meg ha a bejelentkezett felhasználó admin, ha nem vissza küldi a felhasználót a főoldalra |
| **UserProfile.tsx** | /user/:id -URL-en lekéri az url-ben szereplő id-jú user adatait, majd megjeleníti UserProfileDetails.tsx, UserNotifications.tsx SendUserNotification.tsx UserPasswordEmail.tsx komponenseket |
| **ProfilePic.tsx** | Egy function ami egy stringet vár és az elejére hozzáfűzi hogy: „data:image/jpeg;base64,” |
| **SendUsersNotification.tsx** | /send-users-notification -URL-en lekéri az adatbázisban lévő szolgáltatásokat, értesítés típusokat, címet, üzenetet lehet megadni és a Send out notifications gombra kattintva csinál egy post requestet az api hoz |
| **UserRole.tsx** | Egy function ami vissza adja a most bejelentkezett felhasználó szerepkörét |
| **ConfirmEmail.tsx** | /auth/forgot-password -URL-en megadott email-re lehet kérni új verification emailt |
| **UserNotifications.tsx** | Megjeleníti a Propsban átadott felhasználónak az értesítés beállításait, és meg lehet változtatni azokat |
| **UserPasswordEmail.tsx** | Megjeleníti a jelszó, email változtató felületet |
| **UserProfileDetails.tsx** | Megjeleníti a a Propsban átadott id-jú felhasználónak az adatait, amiket lehet változtatni |
| **NewsNotifsModel.ts** | Adatmodell ami a notif\_types, és news\_services tömbben tárol NotifTypeModel és NewsServiceModel-eket |
| **NewsServiceModel.ts** | Adatmodell ami egy szolgáltatás id-ját és nevét adja meg |
| **NotifTypeModel.ts** | Adatmodell ami egy értesítés módnak az id-ját és nevét adja meg |
| **SendUserNotification.tsx** | A Propsban megadott id-jú felhasználónak lehet értesítést küldeni a kiválasztott értesítés típus alapján, a megadott címmel és üzenettel |

|  |  |
| --- | --- |
| **Módosított fájlok** | |
| **Fájlnév** | **Szerep** |
| **DefaultLayoutConfig.ts** | Különböző layout elemeknek az alap viselkedését adja meg |
| **Footer.tsx** | Oldal alján pár információt tartalmaz |
| **MasterLayout.tsx** | Ez adja az oldalnak a felépítését, mi hol helyezkedik el. |
| **AuthCRUD.ts** | Az axios requesteket tartalmazza az API felé |
| **index.tsx** | A kliens egyik fő fájlja amiből kiindul az egész react applikáció |
| **MockAxios.ts** | Axios lekérdezéseket imitáló fájl |
| **UserModel.ts** | Adatmodell ami egy felhasználó adatait adja meg |
| **PrivateRoutes.tsx** | Azokat az url címeket/route okat tartalmazza amihez szükséges hogy legyen érvényes JWT-token |
| **DashboardWrapper.tsx** | Sima felhasználónak a fő oldala bejelentkezés után, példa komponenseket tartalmaz |
| **ErrorsPage.tsx** | Tartalmazza a különböző hiba komponenseket, és az azokhoz vezető utat |
| **Login.tsx** | Bejelentkezés oldal |
| **Mixed.tsx** | Példa adatokat tartalmazó komponens |
| **MixedWidget2.tsx** | Példa adatokat tartalmazó komponens |
| **PublicRoutes.tsx** | Azokat az url címeket/route okat tartalmazza amit bejelentkezés nélkül is ellehet érni |
| **Registration.tsx** | Regisztráció oldal |
| **Notifications.tsx** | Megjeleníti a bejelentkezett felhasználónak az értesítés beállításait, és meg lehet változtatni azokat |
| **UserAddressModel.ts** | Adatmodell ami egy felhasználónak a címét adja meg |
| **UserCommunicationModel.ts** | Adatmodell ami egy felhasználónak az értesítés beállításait adja meg |
| **AuthRedux.ts** | A React-redux hoz tartalmazz különböző acion öket |
| **ProfileDetails.tsx** | Megjeleníti a bejelentkezett felhasználónak az adatait, amiket lehet változtatni |
| **HeaderUserMenu.tsx** | Headerben jobb felül a profil négyzet |
| **MenuInner.tsx** | A Headerben a menüpontokat tartalmazza és azoknak a címét hogy hova irányítsanak át |
| **Topbar.tsx** | A HeaderWrapperben az a komponens ami a HeaderUserMenu komponenst tartalmazza |
| **SignInMethod.tsx** | Megjeleníti a jelszó, email változtató felületet |
| **AuthPage.tsx** | Authentikációhoz szükséges route okat tartalmazza(Bejelentkezés, regisztráció, stb…) |
| **AccountPage.tsx** | Bejelentkezett felhasználónak az account oldala, a Settings komponenst tartalmazza és a szükséges route-okat |
| **Settings.tsx** | Megjeleníti a ProfileDetails, SignInMethods és Notifications komponenst |