1. Backend
2. API-dokumentáció

2.1. userController.js

* Elérési útvonal: /player

2.1.1 registerPlayer()

* Kérés típusa: POST
* Paraméterek:
  + email: string
  + username: string
  + password: string
  + confirmPassword: string
* Feladat: az új játékosok jelszavának hashelése, ezek után az adatok eltárolása az adatbázisban
* Válasz: egy üzenet a sikeres regisztrációról, valamint az adatbázis válasza a sikeres adattárolásról
* Elérési útvonal: /register

2.1.2 loginPlayer()

* Kérés típusa: POST
* Paraméterek:
  + username: string
  + password: string
* Feladat: a megadott jelszó és hashelt jelszó **„bcrypt”** package compare függvényének használatával a jelszavak összehasonlítása, amennyiben ezek egyeznek saját tokent generál, amibe különböző adatokat encryptelünk JWT-hez hasonló módon
* Válasz: a request headerjében megadott **„x-app-type”** kulcs értéke alapján kétfajta választ adhat vissza a backend erre a route-ra:
  + „web”: username és a generált token (később ennek összessége jelszóval elválasztva authCode vagy authToken-ként kerül megnevezésre), valamint egy összegzett statisztikát a játékos eddigi teljesítményei alapján, amit a webes felületünkön megjelenítünk, ezek mellett egy **„tips”** tömböt ez a játék által oktatott három webfejlesztési nyelvekből 1-1 tippet tartalmaz
  + érték nélkül: username és generált token kerül visszaküldésre, valamint egy **„stats”** tömb, amelyben a webes felületre adott statisztikákhoz számolandó adatok kerülnek bele a válaszba
* Elérési útvonal: /login

2.1.3 forgotPassword()

* Kérés típusa: POST
* Paraméter:
  + email: string
* Feladat: a megadott email cím alapján lekéri az email címhez tartozó adatokat, amelyekből egy tokent generálunk (szintén saját tokent) 60 perces lejárati idővel, majd egy emailt küldd a felhasználó emailjére, amiben egy password-reset hivatkozás kerül kiküldésre **„nodemailer”** csomag segítségével GOOGLE SMTP szerveren keresztül
* Válasz: sikeres email küldés esetén egy üzenet, ami értesíti a felhasználót a sikeres email küldésről
* Elérési útvonal: /forgotPassword

2.1.4 validateResetToken()

* Kérés típusa: POST
* Paraméter:
  + token: string
* Feladat: token validálása a password-reset link megnyitása után, hogy megfelelő-e a token és vagy érvényes-e
* Válasz: amennyiben a felhasználó egy érvényes tokennel nyitotta meg a hivatkozását, a válasz szimplán a token önmaga, valamint egy üzenet, hogy sikeres volt az ellenőrzés
* Elérési útvonal: /validateResetToken

2.1.5 resetPassword()

* Kérés típusa: POST
* Paraméterek:
  + token: string
  + password: string
  + confirmPassword: string
* Feladat: token alapján érvényesített felhasználó jelszavának hash-elése és a régi jelszó felülírása az adatbázisban, ennek az eredményéről szintúgy email küldése a felhasználónak sikeres jelszóváltoztatás esetén
* Válasz: üzenet a sikeres jelszóváltoztatásról és az email-es értesítésről
* Elérési útvonal: /resetPassword

2.1.6 getStatistics()

* Kérés típusa: POST
* Paraméter:
  + authCode: string (username és token egy space-el elválasztva)
* Feladat: a weboldalra kiírt statisztikák lekérése, amennyiben lefrissül az oldal újrageneráljuk a statisztikákat, ha változtak volna
* Válasz: egy objektum, amelyben a statisztikák, valamint a tippek vannak
* Elérési útvonal: /getStatistics

2.2 gameController.js

* Elérési útvonal: /game

2.2.1 getHardwareElements()

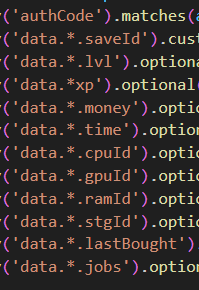
* Kérés típusa: GET
* Paraméter: nincs
* Feladat: az összes hardver elem lekérése az adatbázisból esetleges változtatás miatt
* Válasz: a hardver elemek típus szerint csoportosítva
* Elérési útvonal: /getHardwareElements

2.2.2 getSaves()

* Kérés típusa: POST
* Paraméter:
  + authCode: string
* Feladat: a token validálása után, a token alapján lekéri a játékos adatbázisban tárolt összes saveFile-ját
* Válasz: üzenet a sikeres lekérésről, valamint az összes saveFile adata
* Elérési útvonal: /getPlayerSaves

2.2.3 setSavesOrUpdate()

* Kérés típusa: POST
* Lehetséges paraméterek:



* Feladat: a headerben szereplő **„x-save-type”** kulcs értéke alapján három feladata lehet:
  + érték nélkül: saveFile létrehozása, amennyiben még nem létezik a megadott saveId-val a játékosnál
  + „update”: felülírja a megadott saveId-hoz tartozó adatokat, valamint a statisztikákat és az encryptelt munkát is
  + „override”: ebben az esetben az adott saveId-hoz tartozó adatokat alapértelmezett értékre állítja
* Válasz: üzenet a saveFile sikeres mentéséről/felülírásáról
* Elérési útvonal: /savePlayerData

2.2.4 deleteSave()

* Kérés típusa: POST
* Paraméterek:
  + authCode: string
  + saveId: string
* Feladat: a megadott saveId-hoz tartozó adatok törlése az adott játékos id-ja alapján
* Válasz: üzenet a sikeres saveFile törlésről
* Elérési útvonal: /deleteSave

2.3 adminController.js

* Elérési útvonal: /admin

2.3.1 isAdmin()

* Kérés típusa: POST
* Paraméter:
  + authCode: string
* Feladat: az authCode-ból kiolvasott token alapján a felhasználó adatainak lekérése és ellenőrzése, hogy admin-e a felhasználó
* Válasz: üzenet a sikeres ellenőrzésről és egy objektum, ami tartalmazza az isAdmin kulcs boolean értékét
* Elérési útvonal: /isAdmin

2.3.2 getTableNames()

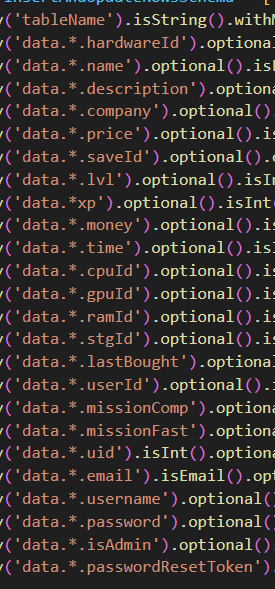
* Kérés típusa: GET
* Paraméter: nincs
* Feladat: az adatbázisban tárolt táblák a táblák neveinek lekérése, a tábla mezőinek a nevét és adattípusát kéri le
* Válasz: egy összesített adathalmaz az adatok alapján csoportosítva
* Elérési útvonal: /getTableNames

2.3.3 getRowsByTableName()

* Kérés típusa: POST
* Paraméter:
  + tableName: string
* Feladat: az adott táblához tartozó összes rekord lekérése az adatbázisból
* Válasz: üzenet a sikeres lekérésről és a rekordok csoportosítva
* Elérési útvonal: /getTableRows

2.3.4 insertRows()

* Kérés típusa: POST
* Lehetséges paraméterek:



* Feladat: a megadott formátumban beilleszteni az adatbázisba a megadott táblához az adatokat, jelszó hashelve kerül minden esetben az adatbázisba
* Válasz: üzenet a sikeres beillesztésről
* Elérési útvonal: /insertRows

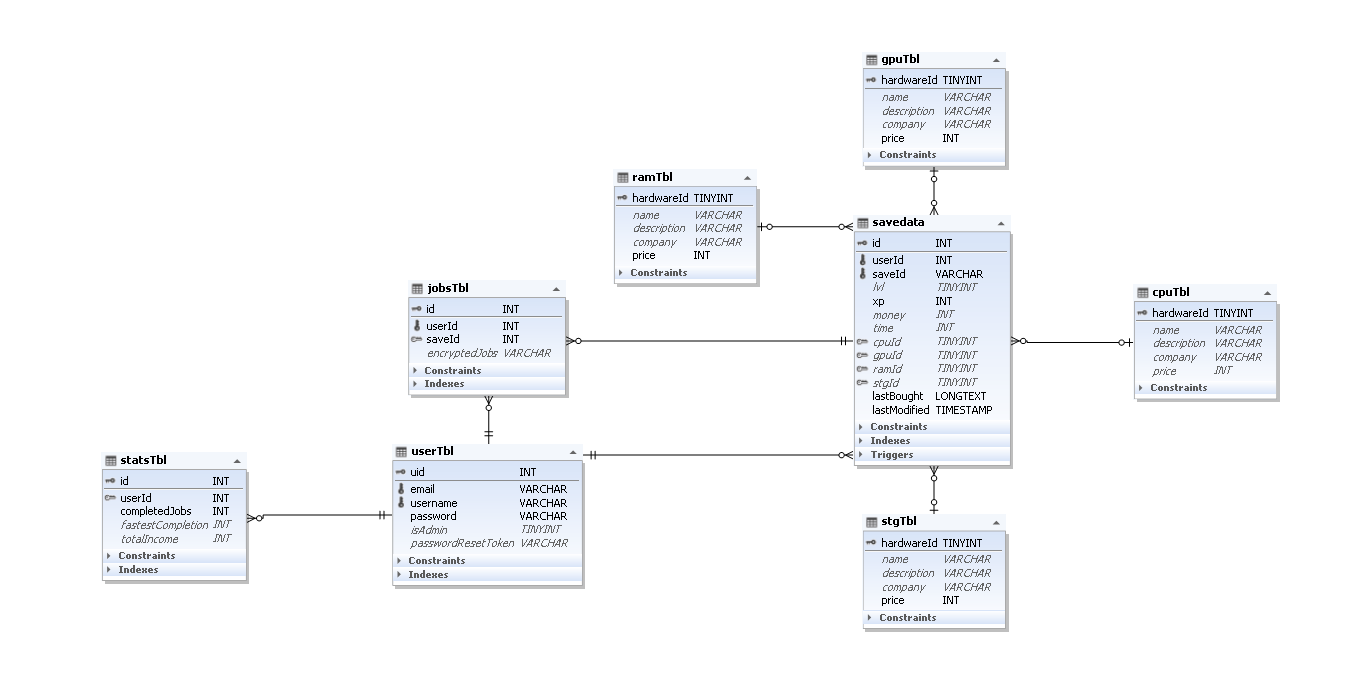
2.3.5 updateRows()

* Kérés típusa: POST
* Lehetséges paraméterek: azonosak az insert paramétereivel
* Feladat: az adatbázishoz képest követi a változásokat és csak a megváltozott adatokat írja felül az adatbázisban
* Válasz: üzenet a sikeres felülírásról
* Elérési útvonal: /updateRows

2.3.6 deleteRows()

* Kérés típusa: POST
* Paraméterek:
  + tableName: string
  + fieldName: string
  + fieldValue: string
* Feladat: a megadott táblából, az adott fieldName-hez tartozó fieldValue rekordját törölni
* Válasz: üzenet a sikeres törlésről
* Elérési útvonal: /deleteRows

1. Adatbázis



3.1 cpuTbl

* hadrdwareId A hardver azonosítója (tinyint(4)), elsődleges kulcs (PK), AUTO\_INCREMENT
* name A hardver neve (varchar(50))
* description A hardver leírása (varchar(255))
* company A hardvert gyártó cég neve (varchar(50))
* price A hardver ára (int(11))

3.2 gpuTbl

* hadrdwareId A hardver azonosítója (tinyint(4)), elsődleges kulcs (PK), AUTO\_INCREMENT
* name A hardver neve (varchar(50))
* description A hardver leírása (varchar(255))
* company A hardvert gyártó cég neve (varchar(50))
* price A hardver ára (int(11))

3.3 jobsTbl

* id A job azonosítója (int(11)), elsődleges kulcs (PK), AUTO\_INCREMENT
* userId A user azonosítója, akihez a job tartozik (int(11)), FK
* saveId A saveFile azonosítója, amelyhez a job tartozik (int(11)), FK
* encryptedJobs A jobok encryptelt formája (varchar(255))

3.4 ramTbl

* hadrdwareId A hardver azonosítója (tinyint(4)), elsődleges kulcs (PK), AUTO\_INCREMENT
* name A hardver neve (varchar(50))
* description A hardver leírása (varchar(255))
* company A hardvert gyártó cég neve (varchar(50))
* price A hardver ára (int(11))

3.5 savedata

* id A saveFile azonosítója (int(11)), elsődleges kulcs (PK), AUTO\_INCREMENT
* userId A saveFile-hoz tartozó user azonosítója (int(11)), FK
* saveId A saveFile neve (varchar(255)), userenként egyedi
* lvl A save-ben a user tárolt szintje (tinyint(4)), alapértlemezett „-1”
* xp A save-ben tárolt xp szint a lvl számolására (int(11))
* money Az összegyűjtött pénz összege (int(11))
* time A save-ben töltött idő másodperces értéke (int(11))
* cpuId A save-ben jelenlegi használt cpu id-ja (tinyint(4)), FK
* gpuId A save-ben jelenlegi használt gpu id-ja (tinyint(4)), FK
* ramId A save-ben jelenlegi használt ram id-ja (tinyint(4)), FK
* stgId A save-ben jelenlegi használt storage id-ja (tinyint(4)), FK
* lastBought A legnagyobb megvásárolt hardver elemek id-ja egy objektumként tárolva (longtext)
* lastModified A saveFile módosításának utolsó időpontja, ez alapján rendezzük a userekhez tartozó saveFile-okat csökkenő sorrendben (timestamp), alapértelmezett érték:
  + - * + current\_timestamp()

3.6 statsTbl

* id A userhez tartozó statisztika azonosítója (int(11)), elsődleges kulcs(PK), AUTO\_INCREMENT
* userId A user azonosítója, amelyhez a statisztika tartozik (int(11))
* completedJobs A user által összes elvégzett jobok összege (int(11)), alapértelmezett értéke 0
* fastestCompletion A user leggyorsabban elvégzett munkájának ideje, demo verzióban nem számolunk vele (int(11)), alapértelmezett érték NULL
* totalIncome A user összes bevétele a játékai során (int(11)), alapértelmezett értéke 0

3.7 stgTbl

* hadrdwareId A hardver azonosítója (tinyint(4)), elsődleges kulcs (PK), AUTO\_INCREMENT
* name A hardver neve (varchar(50))
* description A hardver leírása (varchar(255))
* company A hardvert gyártó cég neve (varchar(50))
* price A hardver ára (int(11))

3.8 userTbl