# Adatbázis dokumentáció:

## ’actings’ táblázat:

Adatkapcsolótáblázat az ’actors’ és ’games’ táblázathoz. Azt mutatja, hogy melyik játékokban milyen színészek játszottak, kiegészítve ezt a színész által eljátszott szereppel.

CREATE TABLE `actings` (

  `id` int(10) UNSIGNED NOT NULL,

  `role` varchar(100) DEFAULT NULL COMMENT 'A színész által eljátszott karakter',

  `GameId` int(10) UNSIGNED DEFAULT NULL,

  `ActorId` int(10) UNSIGNED DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='A szénészek és játékokat kapcsolja össze kiegészítve a színész által eljátszott szereppel';

Az id az azonosító, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A role a színész által eljátszott karakter, karakterlánc, maximum 100 karakter, lehet üres is, mert van, olyan eset mikor a színész fel van tűntetve, de nincs tisztázva a szerepe. A GameId játék azonosítója a games táblázatból, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. Az ActorId színész azonosítója az actors táblázatból, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat.

ALTER TABLE `actings`

  ADD PRIMARY KEY (`id`),

  ADD UNIQUE KEY `Actings\_ActorId\_GameId\_unique` (`GameId`,`ActorId`),

  ADD KEY `ActorId` (`ActorId`);

Az id oszlopot elsődleges kulccsá tesszük. Megadjuk, hogy egy színész játék kapcsolat egyedi legyen, így nem ismétlődhet. Az ActorId-hez hozzárendeljük, hogy kulcs legyen.

ALTER TABLE `actings`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=0;

ALTER TABLE `actings`

  ADD CONSTRAINT `actings\_ibfk\_1` FOREIGN KEY (`GameId`) REFERENCES `games` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

  ADD CONSTRAINT `actings\_ibfk\_2` FOREIGN KEY (`ActorId`) REFERENCES `actors` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

Az id-nek megadjuk, hogy az értékét automatikusan növelje. A GameId-hez és ActorId-hez hozzárendeljük, hogy hogy idegen kulcsként működjenek, előbbinél a games táblázat id oszlopának, az utóbbi esetén az actors tábla id oszlopához. Megadjuk, hogy ha bármely két szülő rekord elsődleges kulcsát megváltoztatják, a változtatást ez a tábla kövesse le, és változtassa meg az idegen kulcs értékét az új értékre. Ha egy szülő rekord törlésre kerül, akkor az összekötő rekord is kerüljön törlésre.

## ’actors’ táblázat:

Tárolja a játékokban megjelenő színészek nevét, illetve egy képet róluk.

CREATE TABLE `actors` (

  `id` int(10) UNSIGNED NOT NULL,

  `firstName` varchar(100) DEFAULT NULL COMMENT 'A színész keresztneve',

  `lastName` varchar(100) DEFAULT NULL COMMENT 'A színész vezetékneve',

  `profilePicture` varchar(255) DEFAULT NULL COMMENT 'A színészről tárolt kép elérési útvonala'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='A színészek adatait tárolja';

Az id az azonosító, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A firstName a színész keresztneve, ami karakterlánc, maximum 100 karakter, lehet üres. A lastName a színész vezetékneve, ami karakterlánc, maximum 100 karakter, lehet üres. A profilePicture a színész profilképének az elérési útvonala, ami, karakterlánc, maximum 255 karakter, lehet üres.

ALTER TABLE `actors`

  ADD PRIMARY KEY (`id`);

Megadjuk az id-nek, hogy elsődleges kulcs legyen.

ALTER TABLE `actors`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=62;

Az id-nek megadjuk, hogy az értékét automatikusan növelje.

## ‘agerating’ táblázat:

Korhatár besorolás a játékokhoz, mivel több intézet foglalkozik ezzel, ezért egy játéknak lehet több, akár különböző besorolása az intézményektől.

CREATE TABLE `ageratings` (

  `id` int(10) UNSIGNED NOT NULL,

  `rating` varchar(10) NOT NULL COMMENT 'A korhatár besorolás',

  `institution` varchar(4) NOT NULL COMMENT 'A korhatár besorolást intéző intézmény',

  `url` varchar(255) NOT NULL COMMENT 'A korhatár besorolás képének az elérési útvonala'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='Korhatár besorolások adatait tartalmazó tábla';

Az id az azonosító, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A rating a korhatár besorolás, ami karakterlánc, maximum 10 karakter, nem lehet üres, mert kötelező adat. Az institution a korhatár besorolást adó szervezet, ami karakterlánc, maximum 4 karakter, nem lehet üres, mert kötelező adat. A GameId a játék azonosítója a ’games’ táblázatból, ami egész szám, nem lehet üres, mert kötelező adat.

ALTER TABLE `ageratings`

  ADD PRIMARY KEY (`id`);

Megadjuk, hogy az id legyen elsődleges kulcs.

ALTER TABLE `ageratings`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=0;

Az azonosítónak megadjuk, hogy az értékét automatikusan növelje.

## ’creations’ táblázat:

A ’creators’ és ’games’ táblázatot köti össze kiegészítve ezt azzal, hogy a fejlesztő milyen területen járult hozzá a játék elkészítéséhez.

CREATE TABLE `creations` (

  `id` int(10) UNSIGNED NOT NULL,

  `field` varchar(50) DEFAULT NULL COMMENT 'A személy munkaterületét írja le',

  `GameId` int(10) UNSIGNED DEFAULT NULL,

  `CreatorId` int(10) UNSIGNED DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='Összeköti a készítőket az általuk részt vett játékokkal és hogy milyen téren vett részt';

Az id az azonosító, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A field azt mutatja, hogy a készítő milyen területen dolgozott az adott játékon, karakterlánc, maximum 50 karakter, lehet üres az olyan esetek miatt, ahol nem tisztázott a személy szerepe. A GameId a játék azonosítója a ’games’ táblázatból, egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A CreatorId a készítő azonosítója a ’creator’ táblázatból, egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, kötelező adat.

ALTER TABLE `creations`

  ADD PRIMARY KEY (`id`),

  ADD UNIQUE KEY `Creations\_CreatorId\_GameId\_unique` (`GameId`,`CreatorId`),

  ADD KEY `CreatorId` (`CreatorId`);

Megadjuk, hogy az azonosító legyen az elsődleges kulcs. Egy adott játék és fejlesztő kapcsolat egyedi, így nem ismétlődhet. Végül kulcsokká alakítjuk a GameId és CreatorId oszlopokat.

ALTER TABLE `creations`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=0;

ALTER TABLE `creations`

  ADD CONSTRAINT `creations\_ibfk\_1` FOREIGN KEY (`GameId`) REFERENCES `games` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

  ADD CONSTRAINT `creations\_ibfk\_2` FOREIGN KEY (`CreatorId`) REFERENCES `creators` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

Az id-nek megadjuk, hogy az értékét automatikusan növelje. Megkötéseket hozunk létre, hogy a GameId oszlop a games tábla id oszlopának az idegen kulcsa legyen. Hasonlóan a CreatorId oszlop pedig a creators tábla id oszlop idegen kulcsa legyen. Megadjuk, hogy ha bármely két szülő rekord elsődleges kulcsát megváltoztatják, a változtatást ez a tábla kövesse le, és változtassa meg az idegen kulcs értékét az új értékre. Ha egy szülő rekord törlésre kerül, akkor az összekötő rekord is kerüljön törlésre.

## ’creators’ táblázat:

A játék fejlesztőit tartalmazza, és hogy milyen területen dolgoznak, bele értve a producereket, rendezőket és zeneszerzőket.

CREATE TABLE `creators` (

  `id` int(10) UNSIGNED NOT NULL,

  `firstName` varchar(100) DEFAULT NULL COMMENT 'A készítő keresztneve',

  `lastName` varchar(100) DEFAULT NULL COMMENT 'A készítő vezetékneve'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='A játékokban részvevő készítőket tárolja';

Az id az azonosító, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A firstName a személy keresztneve, ami karakterlánc, maximum 100 karakter, lehet üres. A lastName a személy vezetékneve, ami karakterlánc, maximum 100 karakter, lehet üres.

ALTER TABLE `creators`

  ADD PRIMARY KEY (`id`);

ALTER TABLE `creators`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=0;

Az id oszlopot elsődleges kulcsra alakítjuk, majd megadjuk, hogy az értékét automatikusan növelje.

## ’favourites’ táblázat:

A felhasználók által kedvencekbe elmentett játékok.

CREATE TABLE `favourites` (

  `id` int(10) UNSIGNED NOT NULL,

  `GameId` int(10) UNSIGNED DEFAULT NULL,

  `UserId` int(10) UNSIGNED DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='A felhasználók által a kedvencekbe mentett játékaikat tárolja';

Az id az azonosító, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A UserId a felhasználó azonosítója a users tábla id oszlopából, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A GameId a játék azonosítója a games tábla id oszlopából, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat.

ALTER TABLE `favourites`

  ADD PRIMARY KEY (`id`),

  ADD KEY `GameId` (`GameId`),

  ADD KEY `UserId` (`UserId`);

Az id-nek megadjuk, hogy elsődleges kulcs. A UserId és GameId oszlopokat kulcsokká alakítjuk.

ALTER TABLE `favourites`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=0;

ALTER TABLE `favourites`

  ADD CONSTRAINT `favourites\_ibfk\_1` FOREIGN KEY (`GameId`) REFERENCES `games` (`id`) ON DELETE SET NULL ON UPDATE CASCADE,

  ADD CONSTRAINT `favourites\_ibfk\_2` FOREIGN KEY (`UserId`) REFERENCES `users` (`id`) ON DELETE SET NULL ON UPDATE CASCADE;

Az id-nek megadjuk, hogy az értékét automatikusan növelje. Megkötésekben megadjuk, hogy a UserId idegen kulcs legyen a users tábla id oszlopának, illetve a GameId is idegen kulcs legyen a games tábla id oszlopának. Megadjuk, hogy ha bármely két szülő rekord elsődleges kulcsát megváltoztatják, a változtatást ez a tábla kövesse le, és változtassa meg az idegen kulcs értékét az új értékre. Ha egy szülő rekord törlésre kerül, akkor az összekötő rekord helyén az érték legyen NULL.

## ’gamepictures’ táblázat:

Ez a tábla a játékokhoz tartozó képeknek az elérési útját tárolják játék betekintésére szánt képeket.

CREATE TABLE `gamepictures` (

  `id` int(10) UNSIGNED NOT NULL,

  `url` varchar(255) NOT NULL COMMENT 'A képernyőkép elérési útvonala',

  `GameId` int(10) UNSIGNED DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='A játékok képernyőképeinek az elérési útvonalait tárolja';

Az id az azonosító, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. Az url a kép elérési útvonala, ami karakterlánc, maximum 255 karakter, nem lehet üres, mert kötelező adat. A GameId a játék azonosítója a ’games’ táblából, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat.

ALTER TABLE `gamepictures`

  ADD PRIMARY KEY (`id`),

  ADD KEY `GameId` (`GameId`);

Az id-nek megadjuk, hogy elsődleges kulcs. A GameId oszlopot kulccsá alakítjuk.

ALTER TABLE `gamepictures`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=0;

ALTER TABLE `gamepictures`

  ADD CONSTRAINT `gamepictures\_ibfk\_1` FOREIGN KEY (`GameId`) REFERENCES `games` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

Az id-nek megadjuk, hogy az értékét automatikusan növelje. Megkötésekben megadjuk, hogy a GameId idegen kulcs legyen a games tábla id oszlopának. Megadjuk, hogy ha a szülő rekord rekord törlésre kerül, akkor az összekötő rekord is kerüljön törlésre, ha a szülő rekord elsődleges azonosítója megváltozik, az összekötő rekord kövesse le a változást.

## ’games’ táblázat:

Ez a fő táblázat, itt tárolódnak a játék egyedi adatai.

CREATE TABLE `games` (

  `id` int(10) UNSIGNED NOT NULL,

  `gameTitle` varchar(350) NOT NULL COMMENT 'A játék címe',

  `altGameTitle` varchar(350) DEFAULT NULL COMMENT 'A játék esetleges alternatív címe',

  `description` varchar(255) NOT NULL COMMENT 'Rövid leírás a játékról',

  `release` date NOT NULL COMMENT 'A játék legelső megjelenési dátuma',

  `boxart` varchar(255) DEFAULT NULL COMMENT 'A játék borító képének az elérési útvonala',

  `promoArt` varchar(255) DEFAULT NULL COMMENT 'Egy olyan széles felbontású kép a játékról',

  `controllerSupport` int(10) UNSIGNED DEFAULT NULL COMMENT 'A játék kontroller támogatásának száma, a NULL azt jelenti hogy nem releváns, a 0 hogy nincs támogatás, az 1 hogy csak xbox típusú kontrollert támogat, a 2 hogy az xbox mellet a playstation részleges támogatást kap, a 3 hogy xbox és playstation kontrollert is teljesen támogat',

  `crossplatform` int(10) UNSIGNED DEFAULT NULL COMMENT 'A crossplay támogatás száma, a NULL azt jelenti hogy nem releváns, a 0 hogy nincs, az 1 hogy részleges támogatás, a 2 hogy teljes támogatás',

  `crossPlatformException` varchar(255) DEFAULT NULL COMMENT 'Azt magyarázza hogy a részleges crossplay támogatást mit jelent az adott játék esetén'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='A játékok egyedi adatait tárolja';

Az id az azonosító, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A gameTitle a játék neve, ami karakterlánc, maximum 350 karakter, nem lehet üres, mert kötelező adat. Az altGameTitle a játék alternatív neve, ami jöhet onnan, hogy más régióban más nevet vagy számozást kapott, illetve egy kisegítő, ha az eredeti neve megegyezik egy másik játék nevével, karakterlánc típusú, maximum 350 karakter, lehet üres. A description a játék rövid leírása, ami karakterlánc, maximum 255 karakter, nem lehet üres, mert kötelező adat. A release a játék legelső megjelenési ideje, ami dátum típusú, nem lehet üres, mert kötelező adat. A boxart a játék borítóképének az elérési útvonala, ami karakterlánc, maximum 255 karakter, lehet üres. A promoart a játékhoz tartozó nagyobb méretű promóciós képe, ami karakterlánc, maximum 255 karakter, lehet üres. A controllerSupport azt mutatja, hogy támogatja egy kontroller használatot számítógépen, előjel nélküli, mert nem lehet negatív, egész szám, ami ezeket az eseteket jelöli: a NULL, hogy nem releváns, mert pl PC-re nem jelent meg vagy hiányzik, 0, vagyis nincs natív kontroller támogatás. 1, vagyis csak Xbox kontroller támogatása van. 2, vagyis Playstation kontroller esetén nincs teljes támogatás, ami során a kontroller működik, de nem a megfelelő gombkiosztás jelenik meg. 3, vagyis Xbox és Playstation kontrollert egyaránt támogat. A crossplatform azt mutatja meg hogy támogatja e, hogy a játékokat különböző platformokon egymással tudjanak-e játszani, előjel nélküli, mert nem lehet negatív, egész szám, ami ezeket az eseteket tárolja: NULL, nem releváns, mert a játéknak nincs több játékos módja, 0, vagyis nincs crossplatfrom. 1, részleges támogatás van. 2, vagyis konzolok és számítógépek között egyaránt. A crossPlatformException az elöbbi esetet bővití ki, ha szükséges, a félreértéseket elkerülve, karakterlánc típusú, maximum 255 karakter, lehet üres.

ALTER TABLE `games`

  ADD PRIMARY KEY (`id`);

ALTER TABLE `games`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=0;

A games táblának az id oszlopát elsődleges kulcsnak adjuk meg. Majd megadjuk, hogy értékét automatikusan növelje.

## ’gamesawards’ táblázat:

Összekötő tábla a games és awards tábla között, ami a díjakat a játékokkal összeköti, hozzáadva hogy melyik évben jelölték a díjra és hogy megnyerte-e.

CREATE TABLE `gamesawards` (

  `id` int(10) UNSIGNED NOT NULL,

  `year` int(10) UNSIGNED NOT NULL COMMENT 'Az év amikor a játékot a díjra jelölték',

  `result` tinyint(1) NOT NULL DEFAULT 0 COMMENT 'Az eredmény, ahol a 0 azt jelenti hogy a játékot csak jelölték a díjra, az 1 hogy megnyerte a díjat',

  `GameId` int(10) UNSIGNED DEFAULT NULL,

  `AwardId` int(10) UNSIGNED DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='A játékokat köti össze azokkal a díjakkal, amelyekre jeölték és az adott jelölésre specifikus adatokkal kiegészítve';

Az id az azonosító, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A year a díjazás éve, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A result a díjazás eredménye, ami logikai, ha igaz, akkor a játék megnyerte a díjat, ha hamis, akkor csak jelölve volt, alapértelmezetten hamis. A GameId a játék azonosítója a ’games’ táblából, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. Az AwardId a díj azonosítója az ’awards’ táblából, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat.

ALTER TABLE `gamesawards`

  ADD PRIMARY KEY (`id`),

  ADD UNIQUE KEY `Gamesawards\_AwardId\_GameId\_unique` (`GameId`,`AwardId`),

  ADD KEY `AwardId` (`AwardId`);

A games táblának az id oszlopát elsődleges kulcsnak adjuk meg. Megadjuk, hogy egy játék és díj összekötés egyedi legyen, így nem ismételhető. Az AwardId oszlopot kulcsá alakítjuk.

ALTER TABLE `gamesawards`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=0;

ALTER TABLE `gamesawards`

  ADD CONSTRAINT `gamesawards\_ibfk\_1` FOREIGN KEY (`GameId`) REFERENCES `games` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

  ADD CONSTRAINT `gamesawards\_ibfk\_2` FOREIGN KEY (`AwardId`) REFERENCES `awards` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

Az id-nek megadjuk, hogy az értékét automatikusan növelje. Megkötésekben megadjuk, hogy a GameId idegen kulcs legyen a games tábla id oszlopának, illetve az AwardId legyen az award tábla id oszlopának az idegen kulcsa. Megadjuk, hogy ha bármely két szülő rekord elsődleges kulcsát megváltoztatják, a változtatást ez a tábla kövesse le, és változtassa meg az idegen kulcs értékét az új értékre. Ha egy szülő rekord törlésre kerül, akkor az összekötő rekord is kerüljön törlésre.

## ’gameslanguages’ táblázat:

Összekötő tábla a ’games’ és ’languages’ között, hozzá adva hogy a játékhoz tartozik e szinkron az adott nyelven.

CREATE TABLE `gameslanguages` (

  `id` int(10) UNSIGNED NOT NULL,

  `dub` tinyint(1) NOT NULL DEFAULT 0 COMMENT 'Az adott nyelvhez tartozik-e szinkron, a 0 nemet jelent, az 1 az igent',

  `GameId` int(10) UNSIGNED DEFAULT NULL,

  `LanguageId` int(10) UNSIGNED DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='A játékokat köti össze a játék által támogatott nyelvekkel kiegészítve, hogy tartozik-e hozzá szinkron';

Az id az azonosító, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A dub azt mutatja, hogy tartozik-e a játékhoz szinkron is, ami logikai, ha igaz, akkor a játék az adott nyelven szinkronizálva is van, ha hamis, akkor csak felirat érhető el, nem lehet üres, kötelező adat, alapvetően hamis. A GameId a játék azonosítója a ’games’ táblázatból, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A LanguageId a nyelv azonosítója a ’languages’ táblázatból, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat.

ALTER TABLE `gameslanguages`

  ADD PRIMARY KEY (`id`),

  ADD UNIQUE KEY `Gameslanguages\_LanguageId\_GameId\_unique` (`GameId`,`LanguageId`),

  ADD KEY `LanguageId` (`LanguageId`);

Az id-nek megadjuk, hogy elsődleges kulcs. Megadjuk, hogy egy játék és nyelv összekötés egyedi eset legyen, így nem ismétlődhet. A LanguageId oszlopot kulcsá alakítjuk.

ALTER TABLE `gameslanguages`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=0;

ALTER TABLE `gameslanguages`

  ADD CONSTRAINT `gameslanguages\_ibfk\_1` FOREIGN KEY (`GameId`) REFERENCES `games` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

  ADD CONSTRAINT `gameslanguages\_ibfk\_2` FOREIGN KEY (`LanguageId`) REFERENCES `languages` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

Az id-nek megadjuk, hogy az értékét automatikusan növelje. Megkötésekben megadjuk, hogy a GameId idegen kulcs legyen a games tábla id oszlopának, illetve az LanguageId legyen az languages tábla id oszlopának az idegen kulcsa. Megadjuk, hogy ha bármely két szülő rekord elsődleges kulcsát megváltoztatják, a változtatást ez a tábla kövesse le, és változtassa meg az idegen kulcs értékét az új értékre. Ha egy szülő rekord törlésre kerül, akkor az összekötő rekord is kerüljön törlésre.

## ’gamesplatforms’ táblázat:

Összekötő tábla a ’games’ és az ’platforms’ között, további információt nem tartalmaz.

CREATE TABLE `gamesplatforms` (

  `id` int(10) UNSIGNED NOT NULL,

  `GameId` int(10) UNSIGNED DEFAULT NULL,

  `PlatformId` int(10) UNSIGNED DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='A játékokat köti össze azokkal a platformokkal amelyeken megjelentek';

Az id az azonosító, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A GameId a játék azonosítója a ’games’ táblázatból, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A PlatformId a platform azonosítója a ’platforms’ táblázatból, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat.

ALTER TABLE `gamesplatforms`

  ADD PRIMARY KEY (`id`),

  ADD UNIQUE KEY `Gamesplatforms\_PlatformId\_GameId\_unique` (`GameId`,`PlatformId`),

  ADD KEY `PlatformId` (`PlatformId`);

Az id-nek megadjuk, hogy elsődleges kulcs. Megadjuk, hogy a játék és platform összeköttetés egyedi legyen, így nem ismétlődhet. A PlatformId oszlopot kulccsá alakítjuk.

ALTER TABLE `gamesplatforms`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=0;

ALTER TABLE `gamesplatforms`

  ADD CONSTRAINT `gamesplatforms\_ibfk\_1` FOREIGN KEY (`GameId`) REFERENCES `games` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

  ADD CONSTRAINT `gamesplatforms\_ibfk\_2` FOREIGN KEY (`PlatformId`) REFERENCES `platforms` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

Az id-nek megadjuk, hogy az értékét automatikusan növelje. Megkötésekben megadjuk, hogy a GameId idegen kulcs legyen a games tábla id oszlopának, illetve az PlatformId legyen az platforms tábla id oszlopának az idegen kulcsa. Megadjuk, hogy ha bármely két szülő rekord elsődleges kulcsát megváltoztatják, a változtatást ez a tábla kövesse le. Ha egy szülő rekord törlésre kerül, akkor az összekötő rekord is kerüljön törlésre.

## ’gametags’ táblázat:

Összekötő tábla a ’games’ és az ’tags’ között, további információt nem tartalmaz

CREATE TABLE `gamestags` (

  `id` int(10) UNSIGNED NOT NULL,

  `GameId` int(10) UNSIGNED DEFAULT NULL,

  `TagId` int(10) UNSIGNED DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='A játékokat köti azokkal a címkékkel amelyek a játékra illenek';

Az id az azonosító, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A GameId a játék azonosítója a ’games’ táblázatból, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A TagId a tag azonosítója a ’tags’ táblázatból, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat.

ALTER TABLE `gamestags`

  ADD PRIMARY KEY (`id`),

  ADD UNIQUE KEY `Gamestags\_TagId\_GameId\_unique` (`GameId`,`TagId`),

  ADD KEY `TagId` (`TagId`);

Az id-nek megadjuk, hogy elsődleges kulcs. Megadjuk, hogy egy játék és címke kapcsolat egyedi legyen, így nem ismétlődhet. A TagId oszlopot kulccsá alakítjuk.

ALTER TABLE `gamestags`

  ADD CONSTRAINT `gamestags\_ibfk\_1` FOREIGN KEY (`GameId`) REFERENCES `games` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

  ADD CONSTRAINT `gamestags\_ibfk\_2` FOREIGN KEY (`TagId`) REFERENCES `tags` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

Az id-nek megadjuk, hogy az értékét automatikusan növelje. Megkötésekben megadjuk, hogy a GameId idegen kulcs legyen a games tábla id oszlopának, illetve az TagId legyen az tags tábla id oszlopának az idegen kulcsa. Megadjuk, hogy ha bármely két szülő rekord elsődleges kulcsát megváltoztatják, a változtatást ez a tábla kövesse le, és változtassa meg az idegen kulcs értékét az új értékre. Ha egy szülő rekord törlésre kerül, akkor az összekötő rekord is kerüljön törlésre.

## ’languages’ táblázat:

Ez a tábla a különböző nyelveket tartalmaz, amelyek játékok esetén előfordulnak.

CREATE TABLE `languages` (

  `id` int(10) UNSIGNED NOT NULL,

  `language` varchar(25) NOT NULL COMMENT 'A nyelv neve'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='A játékokban előforduló nyelveket tárolja';

Az id az azonosító, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A language a nyelv neve, ami karakterlánc, maximum 25 karakter, nem lehet üres, mert kötelező adat.

ALTER TABLE `languages`

  ADD PRIMARY KEY (`id`),

  ADD UNIQUE KEY `language` (`language`);

ALTER TABLE `languages`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=20;

Az id-nek megadjuk, hogy elsődleges kulcs és hogy a language egyedi adat legyen, szóval két megegyező nevű rekord ne legyen lehetséges. Végül megadjuk, hogy az id az értékét automatikusan növelje.

## ’pcspecs’ táblázat:

A játék minimum és ajánlott gépigényét tárolja. Az eszközök általános rövidítéseit használja, és ami a minimumhoz tartozik, ott szerepel a „min” szó.

CREATE TABLE `pcspecs` (

  `id` int(10) UNSIGNED NOT NULL,

  `minOp` varchar(50) NOT NULL COMMENT 'A minimum rendszer követelményben meghatározott operációs rendszer',

  `minCpu` varchar(100) NOT NULL COMMENT 'A minimum rendszer követelményben meghatározott processzor',

  `minRam` varchar(10) NOT NULL COMMENT 'A minimum rendszer követelményben meghatározott memória',

  `minGpu` varchar(100) NOT NULL COMMENT 'A minimum rendszer követelményben meghatározott videókártya',

  `minDirectx` varchar(20) DEFAULT NULL COMMENT 'A minimum rendszer követelményben meghatározott DirectX verziója',

  `op` varchar(50) DEFAULT NULL COMMENT 'Az ajánlott rendszer követelményben meghatározott operációs rendszer',

  `cpu` varchar(100) DEFAULT NULL COMMENT 'Az ajánlott rendszer követelményben meghatározott processzor',

  `ram` varchar(10) DEFAULT NULL COMMENT 'Az ajánlott rendszer követelményben meghatározott memória',

  `gpu` varchar(100) DEFAULT NULL COMMENT 'Az ajánlott rendszer követelményben meghatározott videókártya',

  `directx` varchar(20) DEFAULT NULL COMMENT 'Az ajánlott rendszer követelményben meghatározott DirectX verzió',

  `storage` varchar(10) DEFAULT NULL COMMENT 'Az rendszer követelményben meghatározott háttártáron szükséges szabad hely',

  `sidenote` varchar(255) DEFAULT NULL COMMENT 'Egyéb rendszerkövetelmény',

  `GameId` int(10) UNSIGNED DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='A játékok rendszer követelményeit tárolja';

Az id, az azonosító, egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, kötelező adat. A minop az a minimum szükséglet operációsrendszer terén, karakterlánc, maximum 50 karakter, nem lehet üres, kötelező adat. A mincpu a minimum szükséglet a processzor terén, karakterlánc, maximum 100 karakter, nem lehet üres, kötelező adat. A minram a minimum szükséglet a memória terén, karakterlánc, maximum 10 karakter, nem lehet üres, kötelező adat. A mingpu a minimum szükséglet a videókártya terén, karakterlánc, maximum 100 karakter, nem lehet üres, kötelező adat. A mindirectx a minimum szükséglet a directx verzió terén, karakterlánc, maximum 20 karakter, nem lehet üres, kötelező adat. Az op az ajánlott szükséglet operációsrendszer terén, karakterlánc, maximum 50 karakter. A cpu az ajánlott szükséglet a processzor terén, karakterlánc, maximum 100 karakter. A ram az ajánlott szükséglet a memória terén, karakterlánc, maximum 10 karakter. A gpu az ajánlott szükséglet a videókártya terén, karakterlánc, maximum 100 karakter. A directx az ajánlott szükséglet a directx verzió terén, karakterlánc, maximum 20 karakter. A storage a szükséges szabad tárhely, karakterlánc, maximum 10 karakter, nem lehet üres, kötelező adat. A sidenote az egyéb megjegyzéseket tartalmazza, karakterlánc, maximum 255 karakter. A GameId a játék azonosítója a ’games’ táblából, egész szám, előjel nélküli, mert nem lehet negatív.

ALTER TABLE `pcspecs`

  ADD PRIMARY KEY (`id`),

  ADD KEY `GameId` (`GameId`);

Az id-nek megadjuk, hogy elsődleges kulcs. A GameId oszlopot kulcsra alakítjuk.

ALTER TABLE `pcspecs`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=17;

ALTER TABLE `pcspecs`

  ADD CONSTRAINT `pcspecs\_ibfk\_1` FOREIGN KEY (`GameId`) REFERENCES `games` (`id`) ON DELETE SET NULL ON UPDATE CASCADE;

Az id-nek megadjuk, hogy az értékét automatikusan növelje. Megkötésekben megadjuk, hogy a GameId idegen kulcs legyen a games tábla id oszlopának és hogy ha a szülő rekord törlésre kerül akkor az idegen kulcs értéke legyen NULL, ha megváltozik az elsődleges kulcs, akkor kövesse a változtatást.

## ’platforms’ táblázat:

Ez a táblázat tartalmazza játék platformokat/konzolokat tartalmaz.

CREATE TABLE `platforms` (

  `id` int(10) UNSIGNED NOT NULL,

  `platform` varchar(40) NOT NULL COMMENT 'A platform neve'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='Különböző platformokat tárol, amelyek hivatalosan támogatják a játékok futtatását';

Az id az azonosító, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A GameId a játék azonosító a ’games’ táblából, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A platform mező a platform, avagy konzol neve, ami karakterlánc, maximum 40 karakter, nem lehet üres, mert kötelező adat.

ALTER TABLE `platforms`

  ADD PRIMARY KEY (`id`),

  ADD UNIQUE KEY `platform` (`platform`);

ALTER TABLE `platforms`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=0;

Az id-nek megadjuk, hogy elsődleges kulcs és hogy a platform egyedi adat legyen, szóval két megegyező nevű rekord ne legyen lehetséges. Végül megadjuk, hogy az id az értékét automatikusan növelje.

## ’rating’ táblázat:

A játékok értékeléseit tartalmazza, amit a regisztrált felhasználók adtak.

CREATE TABLE `ratings` (

  `id` int(10) UNSIGNED NOT NULL,

  `positive` tinyint(1) NOT NULL DEFAULT 0 COMMENT 'Az értékelés, ha 0 az negatív értékelést jelent míg az 1 pozitívat',

  `GameId` int(10) UNSIGNED DEFAULT NULL,

  `UserId` int(10) UNSIGNED DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='A játékok felhasználói értékeléseit tárolja';

Az id az azonosító, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A positive a felhasználó értékelése, ami logikai, ahol a hamis azt jelzi, hogy a felhasználó negatívan értékelte, az igaz meg hogy pozitívan értékelte, nem lehet üres, mert kötelező adat. A GameId a játék azonosítója a ’games’ táblázatból, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A UserId a felhasználó azonosítója a ’users’ táblázatból, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat.

ALTER TABLE `ratings`

  ADD PRIMARY KEY (`id`),

  ADD KEY `GameId` (`GameId`),

  ADD KEY `UserId` (`UserId`);

Az id-nek megadjuk, hogy elsődleges kulcs. A GameId és UserId oszlopokat kulcsokká alakítjuk.

ALTER TABLE `ratings`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=0;

ALTER TABLE `ratings`

  ADD CONSTRAINT `ratings\_ibfk\_1` FOREIGN KEY (`GameId`) REFERENCES `games` (`id`) ON DELETE SET NULL ON UPDATE CASCADE,

  ADD CONSTRAINT `ratings\_ibfk\_2` FOREIGN KEY (`UserId`) REFERENCES `users` (`id`) ON DELETE SET NULL ON UPDATE CASCADE;

Az id-nek megadjuk, hogy az értékét automatikusan növelje. Megkötésekben megadjuk, hogy a GameId idegen kulcs legyen a games tábla id oszlopának, illetve az UserId legyen az users tábla id oszlopának az idegen kulcsa. Mindkét esetben megadjuk hogy ha a szülő rekord törlésre kerül, akkor az összekötő táblában az idegen kulcs értéke legyen NULL, ha a szülő rekord elsődleges kulccsa megváltozik, akkor kövesse le a változtatást.

## ’reviews’ táblázat:

A felhasználók által írt kritikákat tárolja egy specifikus játék esetén.

CREATE TABLE `reviews` (

  `id` int(10) UNSIGNED NOT NULL,

  `title` varchar(100) NOT NULL COMMENT 'A kritika címe',

  `content` text NOT NULL COMMENT 'A kritika törzs szövege',

  `date` date NOT NULL COMMENT 'A kritika létrehozásának a dátuma',

  `UserId` int(10) UNSIGNED DEFAULT NULL,

  `GameId` int(10) UNSIGNED DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='A felhasználók által írt játék kritikákat tárolja';

Az id a kritika azonosítója, egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, kötelező adat. A title a kritika címe, karakterlánc, maximum 100 karakter, nem lehet üres, kötelező adat. A content a kritika törzs szövege, szöveg típusú, nem lehet üres, mert kötelező adat. A date a kritika létrehozásának dátuma, nem lehet üres, kötelező adat. A UserId a felhasználó azonosítója a users táblázatból, egész szám, előjel nélküli, mert nem lehet negatív. A GameId a játék azonosítója a games táblázatból, egész szám, előjel nélküli, mert nem lehet negatív szám.

ALTER TABLE `reviews`

  ADD PRIMARY KEY (`id`),

  ADD KEY `UserId` (`UserId`),

  ADD KEY `GameId` (`GameId`);

Az azonosítót elsődleges kulccsá alakítjuk. A UserId és GameId oszlopokat kulcsokká alakítjuk.

ALTER TABLE `reviews`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=0;

ALTER TABLE `reviews`

  ADD CONSTRAINT `reviews\_ibfk\_1` FOREIGN KEY (`UserId`) REFERENCES `users` (`id`) ON DELETE SET NULL ON UPDATE CASCADE,

  ADD CONSTRAINT `reviews\_ibfk\_2` FOREIGN KEY (`GameId`) REFERENCES `games` (`id`) ON DELETE SET NULL ON UPDATE CASCADE;

Az azonosítónak megadjuk, hogy az értékét automatikusan növelje. Megadjuk, hogy a UserId idegen kulcsként funkcionáljon a users tábla id oszlopának, a GameId pedig a games tábla id oszlopának. Mindkét idegen kulcs esetén, ha a szülő rekord törlésre kerül, akkor az összekötő tábla rekordjában az érték NULL-ra változzon, és ha a szülő rekordok elsődleges kulcsát megváltoztatják, az összekötő tábla kövesse le változást.

## ’studios’ táblázat:

A játék fejlesztő és kiadó stúdiókat tárolja és ezeknek a logóját.

CREATE TABLE `studios` (

  `id` int(10) UNSIGNED NOT NULL,

  `name` varchar(50) NOT NULL COMMENT 'A stúdió neve'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='Különböző játék stúdiókat tárol';

Az id az azonosító, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A name a cég neve, ami karakterlánc, maximum 50 karakter, nem lehet üres, mert kötelező adat.

ALTER TABLE `studios`

  ADD PRIMARY KEY (`id`),

  ADD UNIQUE KEY `name` (`name`);

ALTER TABLE `studios`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=30;

Az id-nek megadjuk, hogy elsődleges kulcs és hogy a name egyedi adat legyen, szóval két megegyező nevű rekord ne legyen lehetséges. Végül megadjuk, hogy az id az értékét automatikusan növelje.

## ’studiosgames’ táblázat:

Összeköti a ’games’ és a ’studios’ táblázatot azonosító alapján, kiegészítve ezt azzal, hogy az adott játék esetén fejlesztő és/vagy kiadói szerepet töltöttek be.

CREATE TABLE `studiosgames` (

  `id` int(10) UNSIGNED NOT NULL,

  `isDeveloper` tinyint(1) NOT NULL DEFAULT 0 COMMENT 'A stúdió részt vett-e a játék fejlesztésében, a 0 nemet, az 1 igent jelent',

  `isPublisher` tinyint(1) NOT NULL DEFAULT 0 COMMENT 'A stúdió részt vett-e a játék kiadásában, a 0 nemet, az 1 igent jelent',

  `GameId` int(10) UNSIGNED DEFAULT NULL,

  `StudioId` int(10) UNSIGNED DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='Összeköti a játékokat a hozzájuk tartozó stúdiókkal kiegészítve a stúdió szerepével';

Az id az azonosító, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A GameId a játék azonosítója, a ’games’ táblázatból kerül ide az ’id’ oszlop, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A StudioId a stúdió azonosítója a ’studios’ táblázatból, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. Az isDeveloper azt mutatja, hogy a stúdió fejlesztette-e a játékot, ami logikai, nem lehet üres, mert kötelező adat, alapvetően hamis. Az isPublisher azt mutatja, hogy a stúdió kiadta-e a játékot, ami logikai, nem lehet üres, mert kötelező adat.

ALTER TABLE `studiosgames`

  ADD PRIMARY KEY (`id`),

  ADD UNIQUE KEY `Studiosgames\_StudioId\_GameId\_unique` (`GameId`,`StudioId`),

  ADD KEY `StudioId` (`StudioId`);

Az id-nek megadjuk, hogy elsődleges kulcs. A StudioId oszlopot kulccsá alakítjuk.

ALTER TABLE `studiosgames`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=0;

ALTER TABLE `studiosgames`

  ADD CONSTRAINT `studiosgames\_ibfk\_1` FOREIGN KEY (`GameId`) REFERENCES `games` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

  ADD CONSTRAINT `studiosgames\_ibfk\_2` FOREIGN KEY (`StudioId`) REFERENCES `studios` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

Az id-nek megadjuk, hogy az értékét automatikusan növelje. Megkötésekben megadjuk, hogy a GameId idegen kulcs legyen a games tábla id oszlopának, illetve az StudioId legyen az users tábla id oszlopának az idegen kulcsa. Megadjuk, hogy ha bármely két szülő rekord elsődleges kulcsát megváltoztatják, a változtatást ez a tábla kövesse le, és változtassa meg az idegen kulcs értékét az új értékre. Ha egy szülő rekord törlésre kerül, akkor az összekötő rekord is kerüljön törlésre.

## ’tags’ táblázat:

A játékokhoz tartozó címkék, avagy ’tag’-ek, ami a játékhoz illenek.

CREATE TABLE `tags` (

  `id` int(10) UNSIGNED NOT NULL,

  `tag` varchar(40) NOT NULL COMMENT 'A címke neve'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='Különböző címkéket tárol amivel a játékokat lehet besorolni';

Az id az azonosító, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A tag a játék címkéje, ami stílust, illetve műfajt jelöl, karakterlánc típusú, maximum 40 karakter, nem lehet üres, mert kötelező adat.

ALTER TABLE `tags`

  ADD PRIMARY KEY (`id`),

  ADD UNIQUE KEY `tag` (`tag`);

ALTER TABLE `tags`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=0;

Az id-nek megadjuk, hogy elsődleges kulcs és hogy a tag egyedi adat legyen, szóval két megegyező nevű rekord ne legyen lehetséges. Végül megadjuk, hogy az id az értékét automatikusan növelje.

## ’users’ táblázat:

Információk a felhasználókról.

CREATE TABLE `users` (

  `id` int(10) UNSIGNED NOT NULL,

  `username` varchar(30) NOT NULL COMMENT 'A felhasználó által megadott becenév',

  `password` varchar(255) NOT NULL COMMENT 'A felhasználó jelszava titkosítva',

  `email` varchar(255) NOT NULL COMMENT 'A felhasználó email címe',

  `admin` tinyint(1) NOT NULL DEFAULT 0 COMMENT 'A felhasználó admin státusza, ahol a 0 azt jelenti hogy csak általános felhasználó, míg az 1 azt jelenti hogy admin',

  `creation` date NOT NULL COMMENT 'A felhasználó fiókjának létrehozásának a dátuma'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='A felhasználók adatait tárolja';

Az id az azonosító, ami egész szám, előjel nélküli, mert nem lehet negatív, nem lehet üres, mert kötelező adat. A username a felhasználó neve, ami karakterlánc, maximum 30 karakter, nem lehet üres, mert kötelező adat. A password a felhasználó jelszava titkosítva, ami karakterlánc, maximum 255 karakter, nem lehet üres, mert kötelező adat. Az email mező a felhasználó email címe, ami karakterlánc, maximum 255 karakter, nem lehet üres, mert kötelező adat. Az admin azt mutatja, hogy a felhasználó rendelkezik-e admin jogokkal, ami logikai, nem lehet üres, mert kötelező adat. A creation a profil létrehozásának időpontja, ami dátum típusú, nem lehet üres, mert kötelező adat.

ALTER TABLE `users`

  ADD PRIMARY KEY (`id`),

  ADD UNIQUE KEY `username` (`username`),

  ADD UNIQUE KEY `email` (`email`);

Az id-nek megadjuk, hogy elsődleges kulcs. Az email és username oszlopok egyedinek állítjuk be, hogy ne engedjen két ugyanolyan rekordot a táblába.

ALTER TABLE `users`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=;

Megadjuk, hogy az id az értékét automatikusan növelje.

# Backend dokumentáció

## HTTP kérések

|  |  |  |  |
| --- | --- | --- | --- |
| URL | Metódus | Magyarázat | Státusz kódok |
| / | GET | A főoldalon megjelennek a hamarosan megjelenő játékok, a nemrég megjelent játékok, továbbá 15 db játékot a következő kategóriákban: lövöldöző, kaland, szerepjáték, versenyzős és stratégia. | 200 – Adatok sikeresen lekérdezve  500 – Ismeretlen hiba |
| PUT | A főoldalon egy működő kereső mező, ami a játékokat tud keresni az eredeti, illetve alternatív nevével is, a keresendő szöveget a frontend elküldi, amit a backend feldolgozz. | 200 – Keresés eredményei lekérve  500 – Ismeretlen hiba |
| /game/:gameId | GET | Az URL-ben megadott paraméterben szereplő szám alapján az azonos számú azonosítójú játék minden hozzá tartozó adatait kéri le. | 200 – Adatok sikeresen lekérdezve  404 – Az URL-ben megadott azonosítóhoz nincs játék  500 – Ismeretlen hiba |
| POST | Egy bejelentkezett felhasználó játék értékelését menti el. A felhasználót a frontend által elküldött token alapján határozzuk meg. A játék azonosítóját az URL paraméteréből nyerjük ki. Az értékelést a frontend küldi el egy body-ban, amiben egy „isPositive” logikai változó van, ahol a false negatív, a true pedig pozitív értékelést jelent, végül elmentjük az adatbázisban. | 201 – Az értékelés sikeresen lementve 400 - Hiányzik a játék azonosítója  401 – Nem engedélyezett a játék értékelés, mert a felhasználó nincs bejelentkezve  404 – Az URL-ben megadott azonosítóhoz nincs játék  409 – A felhasználó már értékelte a játékot  500 – Ismeretlen hiba |
| PUT | Egy bejelentkezett felhasználó kedvencekbe helyezheti az adott játékot. A felhasználót a frontend által elküldött token alapján határozzuk meg. A játék azonosítóját az URL paraméteréből nyerjük ki. Végül elmentjük a kapcsolatot az adatbázisban. | 201 – Az kedvencekbe tétel sikeresen lementve  401 – Nem engedélyezett a játék kedvencekbe tétele, mert a felhasználó nincs bejelentkezve  404 – Az URL-ben megadott azonosítóhoz nincs játék  409 – A felhasználó már a kedvencekbe helyezte a játékot  500 – Ismeretlen hiba |
| /registration | POST | A felhasználó regisztrációját kezeli, az adatokat a felhasználók adják meg amit a frontend továbbít a backend-nek, a backend ellenőrzi az adatok formátumát, hiba esetén hibát dob, ha átment az ellenőrzéseken akkor az adatbázisban elmenti a felhasználó adatait. | 201 – Az adatbázisba sikeresen feljuttattuk az új felhasználó adatokat  400 – Hiba a regisztrációs adatokkal  409 – Már van az adatbázisban olyan felhasználó, akinek ilyen felhasználóneve vagy email címe van.  500 – Ismeretlen hiba |
| /login | POST | A felhasználó bejelentkezését kezeli, az adatokat a felhasználó adja meg, amelyet a frontend közvetít a backend-nek. Ellenőrizzük, hogy van-e ilyen felhasználó az adatbázisban, ha van akkor létrehozzunk egy tokent, és a válaszban visszaküldjük. | 200 – Sikeres bejelentkezés, ehhez a felhasználó kap egy token-t  400 – Hiányzik valamilyen adat vagy üres  404 – Nem létezik ilyen adatokkal felhasználó  500 – Ismeretlen hiba |
| /myprofile | GET | Egy bejelentkezett felhasználó adatait kéri le. A felhasználót a frontend által elküldött token alapján határozzuk meg. | 200 – Adatok sikeresen lekérdezve  401 – A felhasználó nincs bejelentkezve  500 – Ismeretlen hiba |
| POST | Egy bejelentkezett felhasználó megtudja változtatni a felhasználónevét, az email címét és jelszavát. A felhasználót a frontend által elküldött token alapján határozzuk meg. Az adatokat a backend ellenőrzi, hogy a megfelelő formátumban van-e, ha igen akkor az adatbázisban megváltoztatjuk az adatokat. | 201 – Az adat(ok) sikeresen változtatása  400 – Probléma az adatok formátumával  409 – Már ilyen adat vagy adatokkal már van felhasználó  500 – Ismeretlen hiba |
| /favourites | GET | Egy bejelentkezett felhasználó a kedvencekben elmentett játékait kéri le az adatbázisból. A felhasználót a frontend által elküldött token alapján határozzuk meg. | 200 – Adatok sikeresen lekérve  401 – A felhasználó nincs bejelentkezve  500 – Internal Server Error |
| POST | Egy bejelentkezett felhasználó eltávolíthatja a kedvencekben mentett játékát. A felhasználót a frontend által elküldött token alapján határozzuk meg. A játék azonosítóját a frontend küldi el. | 200 – A kapcsolat sikeresen törölve  400 – A játék azonosítója nincs megadva  401 – A felhasználó nincs bejelentkezve  404 – A megadott játék azonosítóhoz nincs játék hozzáadva  500 – Internal Server Error |
| /game/:gameId/review | POST | Egy bejelentkezett felhasználó rövid kritikáját kezeli. Kritikát csak az a felhasználó írhat, aki értékelte a játékot. A felhasználót a frontend által elküldött token alapján határozzuk meg. A játék azonosítóját az URL paraméteréből nyerjük ki. A kritika egy címből (title) és egy törzs szövegből (content) áll, amiket a frontend küldd el és a backend ellenőrzi. Ha jó formátumban van, akkor elmentjük a kritikát az adatbázisban. | 201 – Sikeres kritika létrehozás  400 – Hiba a kapott adatokkal  401 – A felhasználó nincs bejelentkezve  404 – Nincs ilyen azonosítójú játék  409 – A felhasználó már írt kritikát  500 – Ismeretlen hiba |
| DELETE | Egy felhasználó kritikáját törli. Kritikát a saját bejelentkezett felhasználó írója tudja törölni, illetve admin jogokkal rendelkező felhasználó. A felhasználót a frontend által elküldött token alapján határozzuk meg. A játék azonosítóját az URL paraméteréből nyerjük ki. A kritika azonosítóját a frontend küldi el reviewId néven. | 200 – A kritika sikeresen törölve.  401 –A felhasználó nincs bejelentkezve, vagy nincs jogosultsága a törléshez  400 – Hiányzik a játék azonosító vagy a kritika azonosítója  404 – Nincs ilyen azonosítójú játék vagy kritika  500 – Ismeretlen hiba |

# Eszközök

A backend Node.js segítségével lett létrehozva az Express keretrendszerrel, továbbá a Sequelize ORM-et használjuk az adatbázis eléréséhez, ehhez telepítve van mellé a Mariadb csomag is hogy megadjuk az adatbázis dialektusát. A Cors csomag lehetővé teszi, hogy kapcsolatot alakítsunk ki a frontend-el.

Az oldal token-eket alkalmaz a felhasználók kezelésére, ehhez a Jose könyvtár JWE token-eket alkalmazunk, ami az alapvető JWT-hez képest további titkosítással van ellátva. A felhasználók jelszavának titkosításához a Bcrypt csomagot használjuk. A leo-profanity egy olyan csomag, amely eszközöket ad ahhoz, hogy angol nyelvű káromkodásokat szűrjünk ki egy megadott szövegből.

A Nodemon csomagot használjuk, hogy megkönnyítse a fejlesztési oldalt azzal, hogy a szerver minden változtatás mentése után újra induljon, illetve biztosít tiszta kilépést hiba esetén, ez a csomag fejlesztési függőséként van telepítve.

## Mappa struktúra

A backend mappában három fájl található, a package.json és a package-lock.json olyan fájlok amik egy Node.js projekt megkezdésekor létre jönnek és meta adatokat tartalmaznak, a .gitignore-t arra használjuk hogy a git commit-ok és push-ok során a fájlban megadott mappák és fájlokat ignorálja, jelen esetben itt csak a node modules mappa van megadva, mert az itt lévő viszonylag sok helyet foglalnak és a tartalmuk az npm install paranccsal gyorsan létrehozhatóak. Itt található még kettő mappa, a node modules a telepített csomagokat tartalmazza, az src mappa a forrás fájlokat tartalmazza.

Az src mappa további öt mappát tartalmaz, a config mappa egy fájlt tartalmaz, amiben az adatbázist Sequelize konstansát tartalmazza. A controllers mappában lévő fájlok kezelik a felhasználó által kezdeményezett kéréseket. A models mappa az adatbázis tábláinak a Sequelize modelljeit tartalmazza, minden tábla egy külön fájl. A routes mappa a controllers mappában lévő metódusokat társítja egy URL-hez és http metódussal, majd ezt exportálja. Az utilities olyan metódusokat tartalmaz, amelyek a projektben több helyen felvannak használva, így csökkentve a redundanciát és átláthatóbb lesz a kód, a különböző fájlok valamilyen csomagra specifikusak, például a bcrypt.methods.js a bcrypt csomaghoz tartozó műveleteket és változókat tartalmazza. A mappa tartalmaz még két fájlt, az app.js segít létrehozni a http szervert és összesíti, hogy milyen kérésekre milyen választ adjon, de ezek meghatározása nem itt történik. Az index.js a szoftver fő fájlja, avagy belépési pontja, itt határozzuk meg a modellek kapcsolatait és ha minden sikeres lefut és hozzá fér az adatbázishoz és szinkronizálta a modelleket, akkor elindítja a szervert.

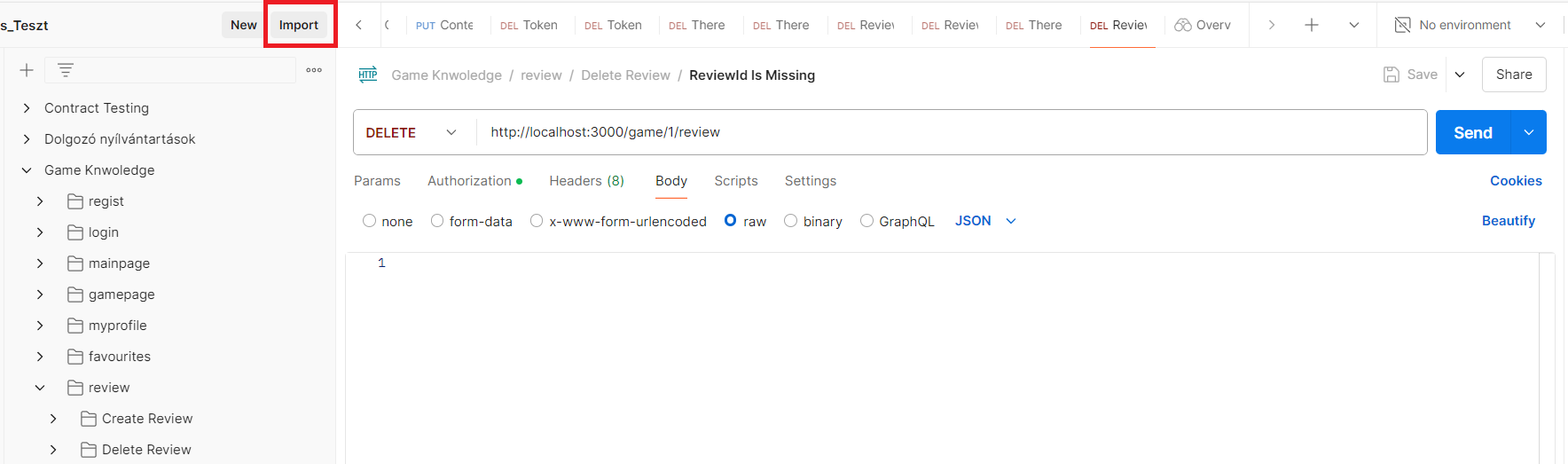
## Backend komponensek működése

msvdfi

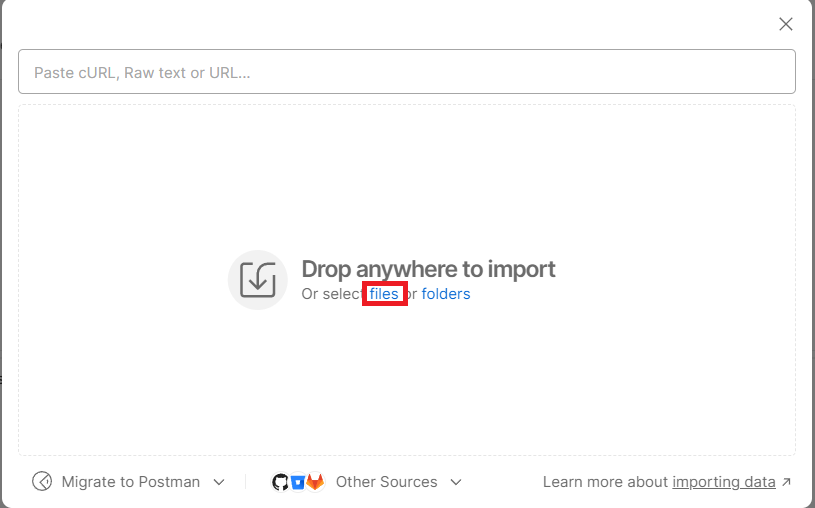
## Backend teszt Postman-nel

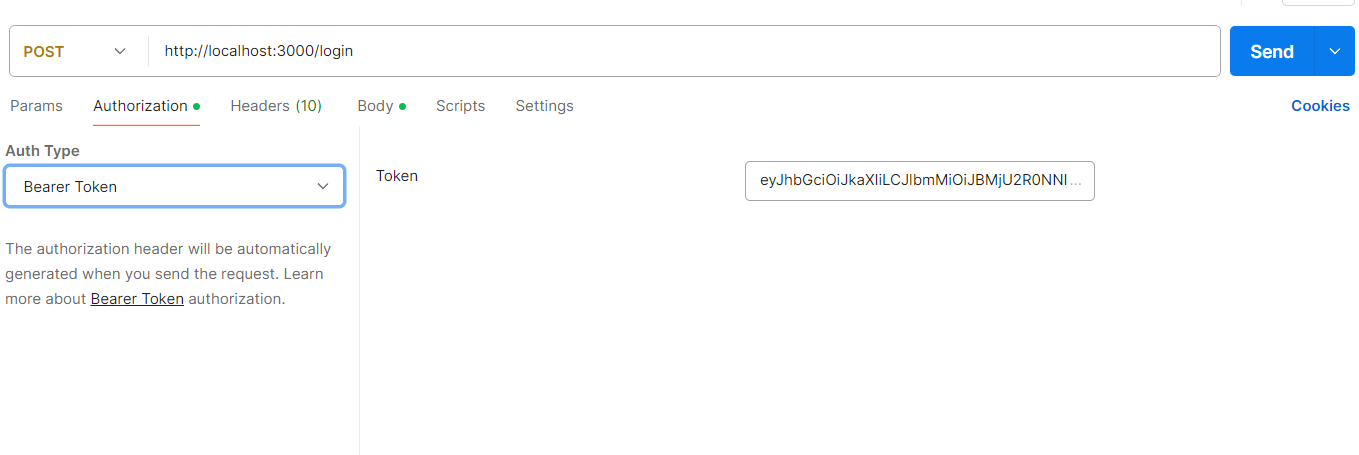
Az API teszteket a Postman alkalmazással tesztelhetjük, így importálhatjuk a ’Game Knwoledge.postman\_collection.json’ fájlt a Postman-be:

Nyissuk meg a Postman alkalmazást és a bal felső sarokban kattintsunk az Import gombra.



Ez bedob egy új ablakot, ahol kattintsunk kékkel írt files-ra majd válasszuk ki a ’Game Knwoledge.postman\_collection.json’.



Ezzel be importáltuk a teszteket, itt hét mappát találhatunk, a mappák neve utal a hozzá tartozó backend fájlra. Ahhoz, hogy biztosítsuk, hogy az elvárt eredményeket kapjuk érdemes egy frissen importált adatbázist használni, és egy tesztnél nincs meghatározva, hogy hányszor futtassuk le, akkor csak egyszer tegyük meg, mert megtörténhet, hogy adatváltoztatások vagy törlések miatt bizonyos funkciók már nem úgy működnek ahogy elvártuk. Bizonyos funkciókhoz egy bejelentkezett felhasználóra van szükség, ehhez a ’login’ mappán belül a ’Login’ mappában található három sikeres bejelentkezés eset: ’Successfull Login With Admin User’,’ Successfull Login With General User’ és ’Successfull Login For Test User’ (de ez esetén később megváltoztatjuk az adatait, azután nem fog működni ez a bejelentkezés). Ezek a sikeres bejelentkezések biztosítanak egy token-t a válasz body-ban, illetve a fejléceknél az ’Authorization’ fejlécében. A token-t tartalmazza a felhasználó adatait és ha egy Postman kérés Authorization fülén, itt az Auth Type-on kiválasztjuk a Bearer Token opciót és a token-t beillesztjük a megadott mezőbe akkor hozzáférhetünk azokhoz a funkciókhoz, amelyekhez be kell jelentkezni előtte.

Fontos megjegyezni, hogy egy token élettartalma egy óra, ezután a token már nem érvényes, tartsuk észbe, ha nem várt hibát kapunk egy kérés során. A backend rendelkezik egy token újító funkcióval, ha a token élettartalma kevesebb mint harminc perc, akkor a ’login’ és ’regist’ mappán belüli kérések kivételével, mindegyik kérésre adhat egy új token-t, ami a válasz fejléc Authorization-nél található. Ha valamelyik kérés során olyan hiba történik amire nem számítottunk akkor http 500-as kódú hibát kapunk.

### ’regist’ teszt mappa

A ’regist’ mappában kettő további mappa található, mindkettő a kéréseket a <http://localhost:3000/registration> küldjük.

A Registration mappában a regisztrációs tesztek érhetőek el POST metódussal.

* Correct/Already Exist – Ez a teszt sikeresen létrehoz egy új felhasználót az adatbázisban, hisz minden adat a megfelelő formátumban van, erre így egy 201-es http kódot kapunk, a válaszban láthatjuk, hogy a jelszót sikeresen titkosította. Ha még egyszer lefuttatjuk akkor hibát kapunk hisz ezekkel az adatokkal már van felhasználó az adatbázisban, a http kód ez esetben 409.
* New User for Test – Ez a teszt sikeresen létrehoz egy új felhasználót az adatbázisban, hisz minden adat a megfelelő formátumban van, erre így egy 201-es http kódot kapunk. Ha még egyszer lefuttatjuk akkor hibát kapunk hisz ezekkel az adatokkal már van felhasználó az adatbázisban, a http kód ez esetben 409. A célunk ezzel a felhasználóval, hogy később felhasználjuk az adat változtatás funkcióhoz.
* Empty Body – Ez a teszt egy üres body-t küldd a backend-nek, mivel az adatok hiányoznak, ezért HTTP 400-as kódú hibát kapunk vissza.
* Missing Username – Ez a teszt egy alapvetően helyes formátumú regisztrációt küldene el, de hiányzik a felhasználónév, ezért HTTP 400-as kódú hibát kapunk vissza.
* Missing Email - Ez a teszt egy alapvetően helyes formátumú regisztrációt küldene el, de hiányzik az email cím, ezért HTTP 400-as kódú hibát kapunk vissza.
* Missing Password - Ez a teszt egy alapvetően helyes formátumú regisztrációt küldene el, de hiányzik a jelszó, ezért HTTP 400-as kódú hibát kapunk vissza.
* Missing Password Confirmation - Ez a teszt egy alapvetően helyes formátumú regisztrációt küldene el, de hiányzik a jelszó megerősítő, ezért HTTP 400-as kódú hibát kapunk vissza.
* Username Too Long – Ebben a tesztben a felhasználónév túllépi a maximum 30 karakter megengedett hosszt, ezért HTTP 400-as kódú hibát kapunk vissza.
* Username Too Short – Ebben a tesztben a felhasználónév nem éri el a minimum 5 karater hosszt, ezért HTTP 400-as kódú hibát kapunk vissza.
* Username Contains Space – Ebben a tesztben a felhasználónév szóközt tartalmaz, amit nem engedünk meg a felhasználónévben, ezért HTTP 400-as kódú hibát kapunk vissza.
* Username Is a Profanity – Ebben a tesztben a felhasználónév egy angol káromkodást, ezt nem engedjük a felhasználónévnek, ezért HTTP 400-as kódú hibát kapunk vissza.
* Email is Missing: @ - Ebben a tesztben az email címből hiányzik a @ jel, ami azt jelenti, hogy ez nem lehet egy hiteles email cím, ezért HTTP 400-as kódú hibát kapunk vissza.
* Email Missing: dot . – Ebben a tesztben az email címből hiányzik a @ jel utáni pont, ez azt jelenti, hogy ez nem lehet egy hiteles email cím, ezért HTTP 400-as kódú hibát kapunk vissza.
* Passwords Don't Match – Ebben a tesztben a jelszó és a jelszó megerősítő nem egyezik, ezért HTTP 400-as kódú hibát kapunk vissza.
* Password Too Short – Ebben a tesztben a jelszó nem éri el a minimum 10 karaktert, ezért HTTP 400-as kódú hibát kapunk vissza.
* Password Too Long – Ebben a tesztben a jelszó meghaladja a 30 karaktert, ezért HTTP 400-as kódú hibát kapunk vissza.
* No Uppercase In The Password – Ebben a tesztben a jelszó nem tartalmaz nagy betűs karaktert, ezért HTTP 400-as kódú hibát kapunk vissza.
* No Number In The Password – Ebben a tesztben a jelszó nem tartalmaz számot, ezért HTTP 400-as kódú hibát kapunk vissza.
* Empty Datas – Ebben a tesztben az adatok értéke üres, ezért HTTP 400-as kódú hibát kapunk vissza.
* Username Is Empty – Ebben a tesztben a felhasználónév értéke üres, ezért HTTP 400-as kódú hibát kapunk vissza.
* Email Is Empty - Ebben a tesztben az email cím értéke üres, ezért HTTP 400-as kódú hibát kapunk vissza.
* Password is Empty - Ebben a tesztben a jelszó értéke üres, ezért HTTP 400-as kódú hibát kapunk vissza.
* Password Confirmation Is Empty – Ebben a tesztben a jelszó megerősítő értéke üres, ezért HTTP 400-as kódú hibát kapunk vissza.
* Username Is Null - Ebben a tesztben a felhasználónév értéke null, ezért HTTP 400-as kódú hibát kapunk vissza.
* Email Is Null - Ebben a tesztben az email cím értéke null, ezért HTTP 400-as kódú hibát kapunk vissza.
* Password Is Null – Ebben a tesztben a jelszó értéke null, ezért HTTP 400-as kódú hibát kapunk vissza.
* Password Confirmation Is Null - Ebben a tesztben a jelszó megerősítő értéke null, ezért HTTP 400-as kódú hibát kapunk vissza.

A ’regist’ mappában a Method Not Allowed mappán belül teszteljük, hogy ha olyan metódusokkal próbáljuk elérni az adott címet, amit nem támogat.

* PUT – Ebben a tesztben PUT metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.
* PATCH – Ebben a tesztben PATCH metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.
* DELETE – Ebben a tesztben DELETE metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.
* GET – Ebben a tesztben GET metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.

### ’login’ teszt mappa

Ebben a mappában további két mappa található, mindkettőben a kéréseket a <http://localhost:3000/login> címre küldi.

A ’Login’ mappában a bejelentkezéshez tartozó tesztek tartózkodnak, amiket POST metódussal érünk el:

* Successfull Login With Admin User – Ebben a tesztben egy sikeres bejelentkezés található egy admin jogosultsággal rendelkező felhasználóval, a válaszban láthatjuk, hogy az ’isAdmin’ értéke igaz. A válaszban továbbá láthatjuk, hogy megkaptuk a felhasználó token-t, amit a felhasználó azonosításához használunk. Ez a token megtalálható a válasz fejléc ’Authorization’ értékénél. A sikeres választ HTTP 200 kóddal kapjuk meg.
* Successfull Login With General User - Ebben a tesztben egy sikeres bejelentkezés található egy átlagos jogosultsággal rendelkező felhasználóval, a válaszban láthatjuk, hogy az ’isAdmin’ értéke hamis. A válaszban továbbá láthatjuk, hogy megkaptuk a felhasználó token-t, amit a felhasználó azonosításához használunk. Ez a token megtalálható a válasz fejléc ’Authorization’ értékénél. A sikeres választ HTTP 200 kóddal kapjuk meg.
* Successfull Login For Test User - Ebben a tesztben egy sikeres bejelentkezés található egy admin jogosultsággal rendelkező felhasználóval, a válaszban láthatjuk, hogy az ’isAdmin’ értéke igaz. A válaszban továbbá láthatjuk, hogy megkaptuk a felhasználó token-t, amit a felhasználó azonosításához használunk. Ez a token megtalálható a válasz fejléc ’Authorization’ értékénél. A sikeres választ HTTP 200 kóddal kapjuk meg, de csak akkor lesz sikeres, ha a regisztrációs tesztnél futtattuk a New User for Test nevű tesztet.
* Username Is Missing – Ebben a tesztben a felhasználónév hiányzik, ezért HTTP 400-as kódú hibát kapunk.
* Password Is Missing - Ebben a tesztben a jelszó hiányzik, ezért HTTP 400-as kódú hibát kapunk.
* Username Is Empty - Ebben a tesztben a felhasználónév üres, ezért HTTP 400-as kódú hibát kapunk.
* Password Is Empty - Ebben a tesztben a jelszó üres, ezért HTTP 400-as kódú hibát kapunk.
* Username Is Null - Ebben a tesztben a felhasználónév NULL, ezért HTTP 400-as kódú hibát kapunk.
* Password Is Null - Ebben a tesztben a felhasználónév NULL, ezért HTTP 400-as kódú hibát kapunk.
* Username Is Incorrect - Ebben a tesztben ilyen felhasználónévvel nincs az adatbázisban felhasználó, ezért HTTP 400-as kódú hibát kapunk.
* Password Is Incorrect - Ebben a tesztben van ilyen felhasználónévvel felhasználó az adatbázisban, de a megadott jelszó nem egyezik az adatbázisban lévővel, ezért HTTP 400-as kódú hibát kapunk.

A login mappán belül a Method Not Allowed mappában találhatunk olyan teszteket, amikkel a nem támogatott metódusokat teszteljük a megadott címen.

* GET – Ebben a tesztben GET metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.
* PUT – Ebben a tesztben PUT metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.
* PATCH – Ebben a tesztben PATCH metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.
* DELETE – Ebben a tesztben DELETE metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.

### ’mainpage’ teszt mappa

A ’mainpage’ mappán belül további három mappa található, amelyekben a tesztek a <http://localhost:3000/> címre kerülnek küldésre.

A ’Mainpage Fetch’ mappában egy darab teszt található, amit GET metódussal tudunk elérni.

* Fetching Games On Mainpage – Ez a teszt azokat az adatokat kéri le, amelyeket a weboldal fő oldalára szeretnénk megjeleníteni, a sikeres lekérdezést HTTP 200-as kóddal kapjuk meg.

A ’Search’ mappában a weboldal keresőmezőjére tartozó tesztek tartoznak, amiket PUT metódussal érünk el. A szöveget, amelyet a kereséshez használunk a kérés body-ban kapjuk meg. A keresést a játék neve és alternatív neve alapján is keres.

* Valid Search – Ebben a tesztben egy sikeres keresés eredményét látjuk, ahol azokat a játékokat adja a válaszban eredményül, ami tartalmazza a body-ban lévő search eredményének a szövegét, kis- és nagybetűtől függetlenül. Az eredményben a játékoknál azokat az adatokat kérjük le, amelyeket a keresés eredményénél szeretnénk megjeleníteni. A válasz HTTP kódja 200.
* Text Is Missing – Ebben a tesztben hiányzik a keresés szövege, erre nem adunk hiba kódot, de a válaszban tudatjuk, hogy ez egy üres keresés. A válasz HTTP kódja 200.
* No Text – Ebben a tesztben a keresés szövege üres, erre nem adunk hiba kódot, de a válaszban tudatjuk, hogy ez egy üres keresés. A válasz HTTP kódja 200.

A ’Method Not Allowed’ mappa olyan teszteket tartalmaz, amelyek a megadott címre nem támogatott metódussal küld kéréseket.

* POST - Ebben a tesztben POST metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.
* PATCH - Ebben a tesztben PATCH metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.
* DELETE - Ebben a tesztben DELETE metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.

### ’gamepage’ teszt mappa

A ’gamepage’ mappán belül négy további mappa található, amelyekben a minden teszt a <http://localhost:3000/game/:gameId> címre kerül elküldésre. Ezek a tesztek a paraméterben megadott számhoz tartozó játékra specifikusak.

A ’Gamepage Fetching’ mappában egy konkrét játék adatainak a lekérdezésére való teszteket tartalmazza a GET metódussal.

Successfull Fetch – Ez a teszt lekérdezi a paraméterben megadott számhoz megegyező azonosítóval rendelkező játét adatait, amelyet a weblapon szeretnénk megjeleníteni. A sikeres lekérdezést HTTP 200-as kóddal jelezzük és válasz tartalmazza a játék adatait.

Game With This Id Does Not Exist – Ebben a tesztben a paraméterben megadott számhoz nem tartozik játék, ezért egy HTTP 404-es kódú hibát kapunk.

A ’Rating’ mappán belüli tesztek a paraméterben megadott számhoz tartozó játék értékelés létrehozását teszteli egy bejelentkezett felhasználó által PUT metódussal. A felhasználó értékelést a válasz body-ban kapjuk az ’isPositive’-ból, ami egy logikai értéknek kell lennie.

* Successfull Positive Rating / Conflict – Ennél a tesztnél használjunk egy általános felhasználó token-jét, és ha egyszer lefuttatjuk akkor egy HTTP 201 kódú választ kapunk, hogy a pozitív értékelést sikeresen lementettük. Ha még egyszer lefuttatjuk akkor egy HTTP 409-es kódú hibát kapunk, hogy a felhasználó már értékelte a játékot.
* Successfull Negative Rating / Conflict - Ennél a tesztnél használjunk egy általános felhasználó token-jét, és ha egyszer lefuttatjuk akkor egy HTTP 201 kódú választ kapunk, hogy a negatív értékelést sikeresen lementettük. Ha még egyszer lefuttatjuk akkor egy HTTP 409-es kódú hibát kapunk, hogy a felhasználó már értékelte a játékot.
* The Game's Not Released Yet - Ennél a tesztnél használjunk egy általános felhasználó token-jét, és ha lefuttatjuk akkor egy HTTP 401-es kódú hibát kapunk, mert olyan játékot próbálunk értékelni, ami még nem jelent meg.
* No Token Provided – Ennél a tesztnél ne használjunk token-t, ha lefuttatjuk a tesztet akkor egy HTTP 401-es kódú hibát kapunk, mert hiányzik a token.
* Faulty Token – Ennél a tesztnél, mér meg van adva egy hibás token, ha lefuttatjuk a tesztet akkor egy HTTP 401-es kódú hibát kapunk, mert a token hibás.
* No Rating Value – Ennél használjunk egy általános felhasználó token-jét, ha lefuttatjuk a tesztet akkor egy HTTP 400-es kódú hibát kapunk, mert a kérés body-ban az értékelés értéke üres.
* Missing Value - Ennél használjunk egy általános felhasználó token-jét, ha lefuttatjuk a tesztet akkor egy HTTP 400-es kódú hibát kapunk, mert a kérés body-ból hiányzik az értékelés.
* Rating's Value Is Not a Boolean - Ennél használjunk egy általános felhasználó token-jét, ha lefuttatjuk a tesztet akkor egy HTTP 400-es kódú hibát kapunk, mert a kérés body-ban nem egy logikai értéket kaptunk.
* Game With This Id Does Not Exist - Ennél a használjunk egy általános felhasználó token-jét, ha lefuttatjuk a tesztet akkor egy HTTP 404-es kódú hibát kapunk, mert a paraméterben lévő számhoz nem tartozik játék.

Az ’Add To Favourites’ mappában lévő tesztek egy bejelentkezett felhasználó a paraméterben megadott számhoz tartozó játék kedvencekbe tételét ellenőrzi.

* Successfully Added To Favourites / Conflict - Ennél használjunk egy általános felhasználó token-jét, ha egyszer lefuttatjuk a tesztet akkor egy HTTP 201-es kódú választ kapunk, hogy a játékot sikeresen lementettük a kedvencekbe. Ha még egyszer lefuttatjuk a tesztet, akkor egy HTTP 409-es kódú hibát kapunk, miszerint a felhasználó már a kedvencekbe tette a játékot.
* Game Does Not Exist - Ennél használjunk egy általános felhasználó token-jét, ha lefuttatjuk akkor egy 404-es kódú hibát kapunk, mert a paraméterben olyan szám van megadva, amihez nem tartozik játék.
* No Token Provided – Ennél a tesztnél ne használjunk tokent, ha lefuttatjuk a tesztet, akkor egy HTTP 401-es kódú hibát kapunk, mert a hiányzik a token.
* Faulty Token – Ennél a tesztnél már megvan adva egy hibás token, ha ha lefuttatjuk a tesztet, akkor egy HTTP 401-es kódú hibát kapunk, mert a token hibás.

A ’Method Not Allowed’ mappában lévő tesztesetekkel teszteljük, hogy ha nem támogatott metódusokkal próbáljuk elérni a megadott címet.

PATCH - Ebben a tesztben PATCH metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.

DELETE - Ebben a tesztben DELETE metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.

### ’myproflie’ teszt mappa

A ’myprofile’ mappában négy további mappa található. Ezek a tesztek egy bejelentkezett felhasználó adatainak lekérését, illetve módosítását kezeli. A kéréseket a http://localhost:3000/myprofile címre küldjük. A felhasználót a token alapján azonosítjuk.

A ’Fetch User Data’ mappában lévő tesztesetek a bejelentkezett felhasználó adatainak lekérését teszteli GET metódussal.

* Successfull Fetch – Ehhez használjuk ’Successfull Login For Test User’-ban lévő felhasználó token-jét, ha lefuttatjuk akkor sikeres választ kapunk HTTP 200-as kóddal, ami tartalmazza a felhasználó adatait.
* No Token Provided – Ehhez ne használjunk token-t, ha így lefuttatjuk a tesztet, akkor HTTP 401-es kódú hibát kapunk, mert nincs token megadva.
* Faulty Token – Ennél a teszt esetnél már meg van adva hibás, ha így lefuttatjuk a tesztet, akkor HTTP 401-es kódú hibát kapunk, mert a megadott token hibás.

A ’Change User Data’ mappában a bejelentkezett felhasználó adatainak a változásaival kapcsolatos tesztek találhatóak. Minden ebben a mappában lévő teszthez használjuk a ’Successfull Login For Test User’ tesztből származó token-t. A címet POST metódussal érjük el. Jegyezzük meg hogy ha sikeresen megváltoztatjuk a felhasználó nevet vagy a jelszót, akkor a bejelentkezéshez használt teszt már nem fog működni a jelenlegi formájában.

* Username Is a Profanity – Ennél a tesztnél a felhasználónevet egy angol káromkodásra próbáljuk megváltoztatni, ezt nem engedélyezzük, ezért egy HTTP 400-as kódú hiba üzenetet küldünk.
* Username Is Too Short – Ennél a tesztnél a megadott felhasználónév nem éri el az 5 karakter hosszúságot, ezért egy HTTP 400-as kódú hiba üzenetet küldünk.
* Username Is Too Long - Ennél a tesztnél a megadott felhasználónév túllépi a 30 karakter hosszúságot, ezért egy HTTP 400-as kódú hiba üzenetet küldünk.
* Username Contains a Space - Ennél a tesztnél a felhasználónévben szóköz található, ezt nem engedélyezzük, ezért egy HTTP 400-as kódú hiba üzenetet küldünk.
* Username Is Same as Before – Ennél a tesztnél a megadott felhasználónév megegyezik az eredeti névvel, mivel ez egy felesleges változtatás lenne, ezért egy HTTP 409-as kódú hiba üzenetet küldünk.
* Username Is Already Taken – Ennél a tesztnél a megadott felhasználónév megegyezik az adatbázisban egy meglévő felhasználó nevével, mivel nem engedjük, hogy két ugyanolyan nevű felhasználó legyen, ezért egy HTTP 409-as kódú hiba üzenetet küldünk.
* Email Is Missing: @ - Ennél a tesztnél a megadott email címből hiányzik a @ jel, ami azt jelenti, hogy ez egy nem hiteles email cím, ezért egy HTTP 400-as kódú hiba üzenetet küldünk.
* Email Is Missing: Dot . - Ennél a tesztnél a megadott email címből hiányzik a @ jel utáni pont, ami azt jelenti, hogy ez egy nem hiteles email cím, ezért egy HTTP 400-as kódú hiba üzenetet küldünk.
* Email Is The Same As Before - Ennél a tesztnél a megadott email cím megegyezik az eredeti email címmel, mivel ez egy felesleges változtatás lenne, ezért egy HTTP 409-as kódú hiba üzenetet küldünk.
* Email Is Already Taken - Ennél a tesztnél a megadott email cím megegyezik az adatbázisban egy meglévő felhasználó email címével, mivel nem engedjük, hogy két ugyanolyan email című felhasználó legyen, ezért egy HTTP 409-as kódú hiba üzenetet küldünk.
* Passwords Don't Match – Ennél a tesztnél a megadott jelszó és jelszó megerősítés nem egyezik, ezért egy HTTP 400-as kódú hiba üzenetet küldünk.
* Password Same As Before – Ennél a tesztnél a megadott jelszó megegyezik az eredeti jelszóval, mivel ez egy felesleges változtatás lenne, ezért egy HTTP 409-as kódú hiba üzenetet küldünk.
* Password Does Not Contain a Number – Ennél a tesztnél a megadott jelszóban nincs szám, ezért egy HTTP 400-as kódú hiba üzenetet küldünk.
* Password Does Not Contain Uppercase Letter - Ennél a tesztnél a megadott jelszóban nincs nagybetűs karakter, ezért egy HTTP 400-as kódú hiba üzenetet küldünk.
* Password Is Too Short - Ennél a tesztnél a megadott jelszó nem éri el a minimum 10 karakter hosszt, ezért egy HTTP 400-as kódú hiba üzenetet küldünk.
* Password Is Too Long - Ennél a tesztnél a megadott jelszó túllépi a maximum 30 karakter hosszt, ezért egy HTTP 400-as kódú hiba üzenetet küldünk.
* Password's Missing Confirmation – Ennél a tesztnél meg van adva a jelszó, de hiányzik a megerősítés, ezért egy HTTP 400-as kódú hiba üzenetet küldünk.
* Only Password Confirmation Given - Ennél a tesztnél nincs megadva jelszó, de a megerősítés meg van adva, ezért egy HTTP 400-as kódú hiba üzenetet küldünk.
* Successfull Username Change – Ennél a tesztnél sikeresen megváltoztatjuk a felhasználó nevét, erre egy HTTP 201-es kódú választ kapunk.
* Successfull Email Change Copy – Ennél a tesztnél sikeresen megváltoztatjuk a felhasználó email címét, erre egy HTTP 201-es kódú választ kapunk.
* Successfull Password Change - Ennél a tesztnél sikeresen megváltoztatjuk a felhasználó jelszavát, erre egy HTTP 201-es kódú választ kapunk.
* Successfull Change Every Data - Ennél a tesztnél sikeresen megváltoztatjuk a felhasználó összes adatát egyszerre, erre egy HTTP 201-es kódú választ kapunk.

A ’Delete User’ mappában a felhasználó törlésével kapcsolatos tesztek találhatóak, illetve található egy bejelentkezést a ’Successfull Login For Test User’-ból lévő felhasználóhoz, aminél lefuttattuk a ’Successfull Change Every Data’ tesztet, ezt továbbra is a POST metódussal érjük el a <http://localhost:3000/login> címen, ezt leszámítva a további teszteket a <http://localhost:3000/myprofile> címen érjük el DELETE metódussal.

* Login For Changed Test User – Ennél a tesztnél bejelentkezünk a teszt felhasználóval, hogy a további tesztet az innen szerzett token-nel hajtsuk végre.
* Successfull Deleting User / User Doesn't Exist – Ha megadtuk a teszt felhasználó token-jét, és egyszer lefuttattjuk, akkor a felhasználó törlésre kerül az adatbázisból, erre HTTP 200-as kódú választ kapunk. Ha még egyszer lefuttatjuk akkor HTTP 404-es kódú hibát kapunk, mert a felhasználó már nem létezik.
* Faulty Token – Ennél a tesztnél már megvan adva egy hibás token, ha lefuttatjuk akkor HTTP 401-es kódú hibát kapunk, mert a token hibás.
* No Token Provided - Ennél a tesztnél nincs megadva token, ha lefuttatjuk akkor HTTP 401-es kódú hibát kapunk, mert a token hiányzik.

A ’Method Not Allowed’ mappában lévő tesztekkel a nem támogatott metódusokat ellenőrizzük.

* PUT - Ebben a tesztben PUT metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.
* PATCH - Ebben a tesztben PATCH metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.

### ’favourites’ teszt mappa

Ebben a mappában három további mappa található, ezekben a mappákban minden kérés a <http://localhost:3000/favourites> címre kerül elküldésre. A kérések egy bejelentkezett felhasználó kedvencekbe mentett játékait kezeli, a tesztekhez használjuk a ’Successfull Login With Admin User’ tesztből származó admin felhasználó token-jét, kivéve ahol a teszt esetnél más van írva.

A ’Fetch Favourites’ mappában a bejelentkezett felhasználó kedvencekbe elmentett játékait kéri le GET metódussal.

* Fetch User's Favourites – Ebben a tesztben a bejelentkezett felhasználó kedvenceikbe elmentett játékait kéri, ha lefuttatjuk akkor egy HTTP 200-as kódú válasz tartalmazza játékok megjelenítésre kívánt adatait.
* Faulty Token – Ebben a tesztben már meg van adva egy hibás token, ha lefuttatjuk akkor HTTP 401-es kódú hibát kapunk, mert a token hibás.
* No Token Provided - Ennél a tesztnél nincs megadva token, ha lefuttatjuk akkor HTTP 401-es kódú hibát kapunk, mert a token hiányzik.

A ’Delete Favourite’ mappában egy bejelentkezett felhasználó a kedvencekből való törlés tesztjeit tartalmazza. Azt, hogy melyik játékot kell törölni azt a kérés body-ban a ’gameId’ értékében kell megadni a játék azonosítóját.

* Game Is Not a Favourite – Ennél a tesztnél egy olyan játék azonosítója van megadva, aminek a megadott felhasználónak nincs a kedvencekbe mentve, ha lefuttattjuk akkor http 404-es kódú hibát kapunk.
* GameId Is Not a Number - Ennél a tesztnél a játék azonosítójának nem egy szám van megadva, ezért, ha lefuttattjuk akkor http 400-as kódú hibát kapunk.
* Empty GameId - Ennél a tesztnél a játék azonosítója üres, ezért, ha lefuttattjuk akkor http 400-as kódú hibát kapunk.
* GameId Is Null - Ennél a tesztnél a játék azonosítójának NULL érték van megadva, ezért, ha lefuttattjuk akkor http 400-as kódú hibát kapunk.
* Missing GameId - Ennél a tesztnél a játék azonosítója hiányzik, ezért, ha lefuttattjuk akkor http 400-as kódú hibát kapunk.
* Successfull Delete – Ennél a tesztnél van ilyen játék elmentve az adott felhasználó kedvenceinél, ezért, ha lefuttattjuk akkor http 200-as kódú választ kapunk a sikeres törlésről.
* Faulty Token – Ebben a tesztben már meg van adva egy hibás token, ha lefuttatjuk akkor HTTP 401-es kódú hibát kapunk, mert a token hibás.
* No Token Provided - Ennél a tesztnél nincs megadva token, ha lefuttatjuk akkor HTTP 401-es kódú hibát kapunk, mert a token hiányzik.

A ’Method Not Allowed’ mappában lévő tesztekkel a nem támogatott metódusokat ellenőrizzük.

* POST - Ebben a tesztben POST metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.
* PUT - Ebben a tesztben PUT metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.
* PATCH - Ebben a tesztben PATCH metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.

### ’review’ teszt mappa

Ebben a mappában további három mappa található, ezekben a bejelentkezett felhasználó egy játékról írt kritikájához tartozó tesztjeit tartalmazzák. Ezekhez a tesztekhez használjuk a ’Successfull Login With Admin User’-ból származó admin felhasználó és ’Successfull Login With General User’-ból származó általános felhasználó token-jét. Az adott teszteknél meg van adva, hogy milyen típusú felhasználó token-t alkalmazzunk. Ebben a mappában minden teszt során a <http://localhost:3000/game/:gameId/review> címet érjük el. Az URL-ben lévő paraméterbe az adott játék azonosítója kerül.

A ’Create Review’ mappában egy bejelentkezett felhasználó kritika létrehozását teszteljük egy adott játéknál, ehhez használjuk az admin jogosultsággal rendelkező felhasználó tokone-jét, de egy kritikát bármilyen bejelentkezett felhasználó létre tud hozni. Ahhoz, hogy valaki kritikát írjon, először értékelnie kell az adott játékot. A kéréshez a body-ban küldeni kell egy ’title’-t, ami a kritika címét kell tartalmaznia és egy ’content’-et ami a kritika törzsszövegét kell tartalmaznia. A megadott címet PUT metódussal érjük el.

* Content Contains Profanity – Ebben a tesztben egy kritikát próbálunk létrehozni, de a törzsszöveg káromkodást tartalmaz, ezt nem engedélyezzük, ezért http 400-as kódú hibát kapunk.
* Title Contains Profanity – Ebben a tesztben egy kritikát próbálunk létrehozni, de a cím káromkodást tartalmaz, ezt nem engedélyezzük, ezért http 400-as kódú hibát kapunk.
* Content Is Null - Ebben a tesztben egy kritikát próbálunk létrehozni, de a törzsszöveg NULL értékű, ezért http 400-as kódú hibát kapunk.
* Content Is Empty - Ebben a tesztben egy kritikát próbálunk létrehozni, de a törzsszöveg üres, ezért http 400-as kódú hibát kapunk.
* Content Is Missing - Ebben a tesztben egy kritikát próbálunk létrehozni, de a törzsszöveg hiányzik, ezért http 400-as kódú hibát kapunk.
* Content Is Too Short - Ebben a tesztben egy kritikát próbálunk létrehozni, de a törzsszöveg nem éri el a minimum 50 karaktert, ezért http 400-as kódú hibát kapunk.
* Content Is Too Long - Ebben a tesztben egy kritikát próbálunk létrehozni, de a törzsszöveg túllépi a maximum megengedett 5000 karaktert, ezért http 400-as kódú hibát kapunk.
* Title Is Null - Ebben a tesztben egy kritikát próbálunk létrehozni, de a cím NULL értékű, ezért http 400-as kódú hibát kapunk.
* Title is Empty - Ebben a tesztben egy kritikát próbálunk létrehozni, de a cím üres, ezért http 400-as kódú hibát kapunk.
* Title Is Missing - Ebben a tesztben egy kritikát próbálunk létrehozni, de a cím hiányzik, ezért http 400-as kódú hibát kapunk.
* Title Is Too Long - Ebben a tesztben egy kritikát próbálunk létrehozni, de a cím túllépi a maximum megengedett 100 karaktert, ezért http 400-as kódú hibát kapunk.
* Title Is Too Short - Ebben a tesztben egy kritikát próbálunk létrehozni, de a cím nem éri el a minimum 5 karaktert, ezért http 400-as kódú hibát kapunk.
* Faulty Token – Ebben a tesztben már meg van adva egy hibás token, ha lefuttatjuk akkor HTTP 401-es kódú hibát kapunk, mert a token hibás.
* No Token Provided - Ennél a tesztnél nincs megadva token, ha lefuttatjuk akkor HTTP 401-es kódú hibát kapunk, mert a token hiányzik.
* There's No Game With This Id – Ebben a tesztben az URL paraméterében olyan szám van megadva, amihez nem tartozik vele megegyező azonosítójú játék, ha lefuttatjuk akkor HTTP 404-es kódú hibát kapunk.
* Successfull Creation / Already Reviewed – Ha egyszer lefuttattjuk akkor sikeresen létrehozzunk egy kritikát az adott játékhoz, erre http 201-es kódú választ kapunk. Ha még egyszer lefuttattjuk akkor http 409-es kódú hibát kapunk, mert a felhasználó már írt egy kritikát az adott játékról.
* The User Did Not Rate The Game – Ebben a tesztben egy olyan játékhoz próbálunk kritikát írni, amit a megadott felhasználó még nem értékelt, ezért http 400-as kódú hibát kapunk.

A ’Delete Review’ mappában a kritikák törléséhez tartozó tesztek tartózkodnak. Egy kritikát csak annak írója vagy admin jogosultságokkal rendelkező felhasználó tudja törölni. Azt, hogy melyik kritikát akarjuk törölni azt a kérés body-ban a ’reviewId’ tartalmazza, amiben a kritika azonosítójának kell lennie. A címet a DELETE metódussal érjük el.

* ReviewId Is Missing – Ehhez a teszthez használjuk az admin jogosultsággal rendelkező felhasználó token-jét, ha lefuttattjuk akkor http 400-as kódú hibát kapunk, mert hiányzik a kritika azonosítója.
* ReviewId Is Null - Ehhez a teszthez használjuk az admin jogosultsággal rendelkező felhasználó token-jét, ha lefuttattjuk akkor http 400-as kódú hibát kapunk, mert a kritika azonosítója NULL.
* ReviewId Is Empty - Ehhez a teszthez használjuk az admin jogosultsággal rendelkező felhasználó token-jét, ha lefuttattjuk akkor http 400-as kódú hibát kapunk, mert a kritika azonosítója üres.
* There's No Review With This Id - Ehhez a teszthez használjuk az admin jogosultsággal rendelkező felhasználó token-jét, ha lefuttattjuk akkor http 404-es kódú hibát kapunk, mert a megadott reviewId-hoz nem tartozik kritika.
* There's No Game Id - Ehhez a teszthez használjuk az admin jogosultsággal rendelkező felhasználó token-jét, ha lefuttattjuk akkor http 400-as kódú hibát kapunk, mert az URL paraméteréből hiányzik a játék azonosítója.
* There's No Game With This Id - Ehhez a teszthez használjuk az admin jogosultsággal rendelkező felhasználó token-jét, ha lefuttattjuk akkor http 404-es kódú hibát kapunk, mert az URL paraméterében megadott számhoz nem tartozik játék ilyen azonosítóval.
* Faulty Token – Ebben a tesztben már meg van adva egy hibás token, ha lefuttatjuk akkor HTTP 401-es kódú hibát kapunk, mert a token hibás.
* No Token Provided - Ennél a tesztnél nincs megadva token, ha lefuttatjuk akkor HTTP 401-es kódú hibát kapunk, mert a token hiányzik.
* Successful Delete With Admin / No Review With This Id - Ehhez a teszthez használjuk az admin jogosultsággal rendelkező felhasználó token-jét, ha egyszer lefuttattjuk akkor sikeresen töröljük a megadott kritikát és http 200-as kódú választ kapunk. A törlés sikeres, mert admin jogosultsággal bármilyen kritikát törölhetünk. Ha még egyszer lefuttattjuk akkor http 404-es hibát kapunk, mert a ’reviewId’-ban megadott kritika nem létezik.
* Successful Delete By The Author / No Review With This Id - Ehhez a teszthez használjuk az említett általános jogosultsággal rendelkező felhasználó token-jét, ha egyszer lefuttattjuk akkor sikeresen töröljük a megadott kritikát és http 200-as kódú választ kapunk. A törlés sikeres, mert az a felhasználó törölte a kritikát, akitől származik. Ha még egyszer lefuttattjuk akkor http 404-es hibát kapunk, mert a ’reviewId’-ban megadott kritika nem létezik.
* Delete Is Not Authorized - Ehhez a teszthez használjuk az említett általános jogosultsággal rendelkező felhasználó token-jét, lefuttattjuk akkor http 401-es hibát kapunk, mert a megadott felhasználónak nincs admin jogosultsága és nem a kritika írója.

A ’Method Not Allowed’ mappában lévő tesztekkel a nem támogatott metódusokat ellenőrizzük.

* GET - Ebben a tesztben GET metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.
* POST - Ebben a tesztben POST metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.
* PATCH - Ebben a tesztben PATCH metódussal próbáljuk elérni a címet, de ezt backend nem támogatja ezért HTTP 405-ös kódú hibát küldünk.