Dokumentáció

HelixLab JAVA vizsgamunkához

Rövid ismertetés:

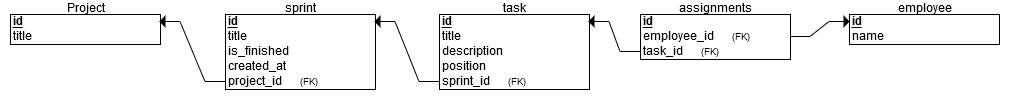
Az alkalmazás egy egyszerű projektmanagement program backend része. A felhasználóknak lehetőségük van:

* projekteket létrehozni, átnevezni, törölni, nevüket megváltoztatni
* projekteken belül sprinteket létrehozni, lezárni, törölni
* sprinteken belül taskokat létrehozni, törölni, task állapotát változtatni
* dolgozókat felvenni, törölni, nevüket változtatni
* taskokat dolgozóknak kiosztani, ilyen kiosztásokat törölni
* projekt és dolgozó statisztikát elkészíteni, projektet validálni

A felhasználók lehetőségein kívül léteznek egyéb dolgok, amiket vagy segédfüggvényként használok, vagy olyan módon lehet velük az adatbázist módosítani, ahogy a végső programban a felhasználóknak majd nem lesz lehetősége. Ez utóbbiakat is benne hagytam a programban. Mivel van validálás is, így könnyebb „elrontani” az adatbázist érdekesebb eredményekért.

Adatbázis:

Relációs séma:



Adattáblák:

* project: projekteket tartalmazza
* sprint: sprinteket tartalmazza. is\_finished alapértéke 0, azt jelzi, hogy az adott sprintet lezárták-e. created\_at alapértelmezetten a létrehozás ideje CURRENT TIMESTAMP formátumban
* task: position értéke jelenti a sprinten belüli helyét. 0- backlogban, 1- dolgoznak rajta, 2- tesztelés alatt, 3 – kész.
* assignment: dolgozókhoz rendelt taskok
* employee: dolgozók

Minden foreign key ONDELETE CASCADE-del lett beállítva. Ha egy olyan rekordot törölnek, amihez tartozik foreign key, akkor az összes olyan rekord is törlődik, ahol a foreign key a primary key. Magyarul ha egy projektet törölnek, akkor az összes benne lévő sprint, az azokban lévő összes task és a hozzájuk rendelt összes assignment is kitörlődik. Ha egy taskot törlünk, akkor „csak” az összes hozzá rendelt assignemnt törlődik ki pluszban.

JAVA alkalmazás:

A követelménynek megfelelően MVC patternnel készült. 3 packagye található benne:

* models: adatbázis manipulációért felel
* services: üzleti logikát tartalmazza
* controllers: az endpointokat tartalmazza és az input validálásért felel

Minden package minden osztályában a függvények nevei úgy lettek elnevezve amit csinálnak. Teljesen egyértelműen utalnak a feladatukra, pl getAllAssignemntByEmployeeId lekérdezi az összes Assignmentet Employee id alapján stb. Azok a változók, amelyek egy példány tulajdonságait jelentik az adatbázis oszlopneveinek camelCase-es verziói, kivéve amikor több id-val kellett dolgoznon, ilyenkor ez egyik „csak sima” id, a másik pedig valahogy utal a saját tartalmára – pl id és sprintId. Azok a változók, amik segédváltozóként vannak használva a függvénynevekhez hasonlóan egyértelműen utalnak a tartalmukra pl taskInLastSprint egy taskot jelent, ami a legutolsó sprintben van.

Models:

Entitás osztályokat tartalmaz az adatbázis alapján automatikusan generálva. A saját kód A //START of own code ÉS //END of own code között található.

Minden entitás toJson() függvénye: a JPA valamiért a foreign key-eket a saját maguknak megfelelő osztály típusban tartja nyilván (pl a task-ban lévő sprintId foreign key Sprint típusú). Értékük viszont a classuk toString() függvényének értéke. Mivel számként szeretném megadni őket, ezért először a classukról stringre, majd onnan intre kell castolni őket.

*Assignment.java*

employee-k és task-ok összekötésére szolgál

függvények:

* getAssignmentById – nem használom, csak validálásra, ezért nincs sem a service sem a controller rétegben
* getAllAssignmentByEmployeeId – egy employee összes task-ja
* getAllAssignmentByTaskId – egy task összes employee-ja
* createNewAssignment
* deleteAssignmentById
* deleteAllAssignmentByEmployeeId – egy employee-t minden task-ból törölni
* deleteAllAssignmentByTaskId – egy task összes employee-ját törli

mindegyik teljesen egyértelmű.

*Employee.java*

employee-k managelése

függvények:

* getAllEmployees
* getEmployeeById
* createNewEmployee
* deleteEmployeeById
* getEmployeeStatisticsById – öszetett függvény, ezekről később

itt is minden egyértelmű

*Project.java*

projectek kezelése

függvények:

* getAllProjects
* getProjectById
* createNewProject
* deleteProjectById
* getProjectStatisticsById - öszetett függvény, ezekről később
* validateProjectById - öszetett függvény, ezekről később

meg itt is minden egyértelmű

*Sprint.java*

sprintek kezelése

függvények:

* getAllSprintByProjectId
* getSprintById
* createNewSprintByProjectId – öszetett függvény, ezekről később
* deleteSprintById
* deleteAllSprintByProjectId – egy projektben az összes sprintent kitörli
* finishSprint – a befejezni kívánt sprinten megvizsgál minden taskot, és amelyik task nem kész (isFinished !=3), azt beállítja 0-nak.
* validateSprintById – segédfüggvény validateProjectById függvényhez

*Task.java*

taskok kezelése

függvények:

* getAllTasksBySprintId – egy sprint összes taskját lekéri
* getTaskById
* createNewTask
* deleteTaskById
* deleteAllTaskBySprintId egy sprint összes taskját kitörli
* validateTaskById – segédfüggvény validateProjectById-hez

Services:

Az összes service függvény feladata az, hogy a controller felől érkező kéréseket fogadja, leellenőrizze, hogy logikailag lehetséges-e a kérés és ha igen, akkor meghívja a modelsben neki megfelelő másik függvényt.

Ahol a service függvény true/false értékkel tér vissza, ott az összes logikai esetet leellenőriztem, de csak true/fals értéket küldtem vissza, így ezeknél nem lehet tudni, hogy miért nem valósult meg a kérés. Ahol JSONArray v JSONObject a visszatérési érték, ott ezekbe beleírtam a probléma okát.

Lehetséges ellenőrzések:

* létezik-e a rekord, amin akarunk valamit csinálni: meghívom az entitás osztályból a getXXXById függvényét (pl getTaskById), és leellenőrzöm, hogy van-e ilyen id-jú példány. Mivel azoknál az entitásoknál, ahol van valamilyen default érték a létrehozáskor (task, sprint), ott ha nem talál id alapján példányt, akkor nem egy üres példány objektet kapok vissza, hanem egy olyent, ahol a default értékek ki vannak töltve, ezért nem tudtam úgy ellenőrizni h isEmpty v length()==0 v size()==0, ezért lekérem a példány id-ját a getterével, és arra ellenőrzök.
* egy lista üres-e : listától függően .isEmpty() vagy .size()==0 vagy length()==0
* task position ellenőrzés: csak 0-3 közötti értéket fogadunk el, sima position <3 && position >0
* 3-as position-ú task positionját nem lehet tovább változtatni: task.getPosition == 3

Contorllers:

Kezeli a http kéréseket és input validálást végez.

A taskok neve általában megegyezik a meghívandó service függvény nevével (ami meg az entitás osztály és az adatbázis tárolt eljárás függvény nevével egyezik meg). Minden egyéb bemenő paraméter valamelyik entitás valamelyik tulajdonsága, ezért elnevezésük ezeknek megfelelő.

Minden típusú input validálás mindegyik constrollernél ugyanúgy néz ki:

int validálás:

* értékét beállítom olyanra, ami nem lehet (pl id-nál 0-ra, task position-nál -1re
* try-on belül megpróbálom a bemenő értéket átcastolni int-té (mivel string-ként jön be)
* ellenőrzöm, hogy nagyobb-e, mint az általam megadott alapérték

string validálás:

* itt csak annyit ellenőrzök le, .trim().length()-tel, hogy nem csupa whitespace-t adtak-e be bemenő értéknek

Öszetett függvények:

*Employee | getEmployeeStatisticsById*

Employee Id alapján lehet statisztikát készíteni.

* lekérjük az összes assignmentjét, ami van neki
* ha nincs, akkor az lesz az özenet, hogy nincs neki egy sem
* ha van, akkor kiszámoljuk mennyi van neki, ezekből mennyi task van kész, hány projektben találhatók az assignmentjei
* kiszámoljuk, hogy milyen arányban voltak az összes/kész taskjai az assignmentjei alapján
* feltöltjük az eredmény JSON-t

*Project | getProjectStatisticsById*

Project Id alapján lehet statisztikát készíteni

* lekérjük a project összes sprintjét
* megszámláljuk, hány darab task van ezekben a sprintekben, és hány darab finished task van közöttük
* kiszámoljuk a taskok, finished taskok sprint darabszámra viszonyított átlagát és az finihsed taks/összes task átlagot
* feltöltjük az eredmény JSON-t

*Project | validateProjectById*

Project id alapján le lehet ellenőrizni, hogy minden hozzá kapcsolódó adat a business logic-nak emgfelelő-e. Két segédfüggvényt tartalmaz:

* Task | validateTaskById: leelenőrzi, hogy egy taskban a position 0 és 3 közt van-e
* Sprint | validateSprintById: minden saját magában lévő task validálását elvégzi, valamint leellenőrzi, hogy ha a sprint lezárt 8finished), akkor csak finished taskok vannak-e benne.

Maga a Project | validateProjectById függvény lekéri az összes saját taskjának a validálását, illetve leellenőrzi, hogy a legfiatalabb sprint kivátelével minden más sprint le van-e zárva (mert új sprintet csak akkor lehet létrehozni, ha az előző le lett zárva).

*Sprint | createNewSprintByProjectId*

Új sprint létrehozása. Sprintet csak akkor lehet létrehozni, ha a korábbi sprint már le lett zárva. Új sprint létrehozásánál az előző sprintből minden olyan taskot áthozunk az új sprintbe, ami nem lett befejezve (position-je nem 3-as)

* lekérjük annak a projektnek az összes sprintjét, ahol új sprintet szeretnénk létrehozni
* leellenőrizzük, hogy minden sprint le lett-e már zárva, egyidejűleg leellenőrizzük, hogy az adott sprint régebbi-e, mint egy segédváltozó, így az ellenőrzés végére megtaláljuk a legutóbbit is.
* ha minden sprint le van zárva, akkor csinálunk egy új sprintet
* mivel az adatbázisban tárolt eljárásban nem tudtam visszaadni az újonnan létrehozott sprint id-ját (valamiért nem ment a last\_created\_id és hasonló módszerek sem), ezért az új sprint létrehozása után lekérem újra az összes sprintet, az új listából .remove()-olom a régit, így az új listában már csak az új sprint mard. Így kapom meg az új sprint id-ját
* a legutóbbi sprintből az összes nem kész (position =+ 3) taskot áthelyezem (sprintId-jukat átírom) az új sprintbe

**Ismert „bug”:** xampp, phpmyadmin, java program, postman fut rendesen. Mindent lehet csinálni, szápen küldeni a kéréseket postmanből, jön válasz, adatbázis változik, sehol semmi gond.

Aztán egyszer egy postman kérés folyamatosan a *sending request* résznél várakozik, és nem történik semmi. Az adatbázis működik tovább, a java program nem jelez hibát sehol, a kérést is lehet cancel-ezni postmanben és a postmant tovább használni, de onnantól kezdve minden kérés csak a *sending request*-ig jut, és nem megy tovább. Legtöbbször valamelyik összetett függvénynél jön elő. Semmilyen hibát nem észleltem, és semmilyen hasonló dolgot nem találtam az interneten.

ui.: a szerver nálam a 8282-es porton meg, uh a postman oda küldi az adatokat