Szakképesítés neve: Szoftverfejlesztő és –tesztelő

Szakma azonosító száma: 506131203

Szakdolgozat

Land of Fitness

Készítette
Kovácsfi Alex
Kuczi Dániel
Kovács Richárd (1999)

Pécs, 2022

Tartalomjegyzék

1. A téma ismertetése, témaválasztás indoklása, szakmai célkitűzés

A projekt alapötlete egy regisztrációs, illetve beléptető szoftver, amiket egy fő oldal nav bar-ról érhetünk el. A téma egy edzőterem weboldala, aminek alapvetően az a célja, hogy adott edzőteremről nem csak teljes körű információt tudjunk nyújtani, de egyben a forgalmazott étrendkiegészítőket online is meg lehessen vásárolni oldalunkról. A weboldal lényegében minden korosztályt egyformán megcélozza.

A lényege, hogy regisztrációnál bekérje a felhasználónak a különböző adatait és ezeket az adatokat elmentse egy adatbázisba. Amennyiben a regisztráció sikeresen megtörtént, a bejelentkezéshez irányit az oldal, és ha egyezés található a felhasználónévben és jelszóban sikeresen belépteti a felhasználót egy felületre, ahol elérheti az adott szolgáltatásokat. A weboldal bejelentkezés nélkül is látogatható, a rajta lévő információ megtekinthető, azonban a termékrendelés csak aktívan bejelentkezett felhasználók számára elérhető. Az oldalon elérhetők a személyi edzők, ahol minden fontos információ meg lesz található róluk, például amelyekkel kapcsolatba lehet velük lépni, illetve az edzőterem címe és egy email cím amennyiben az edzőterem csapatába jelentkezne egy felhasználó. Az oldalon továbbá étrendkiegészítő termékek (valós időben nyomon követhető raktári mennyiséggel) és számtalan bérlettípus vásárlása lehetséges.

A téma választásának oka a személyesen tapasztalt hiányos információ, amelyet a legtöbb edző termi weboldalnál érzékeltünk. Másik probléma volt, hogy míg egyes edzőtermekben lehet különböző táplálékkiegészítőket vásárolni, azt a weboldalukról nem lehet megrendelni, illetve nem lehet nyomon követni az aktuális készletet.

A téma választásához hozzátartozott a megfelelő design kialakítása. Fontos, hogy a színek és a forma passzoljon egy valós edzőterem kialakításához. Az edzőtermeknél és edző termi eszközöknél gyakori színkombináció a piros és fekete, illetve szürke, valamint a fehér színek árnyalatainak (ebben az esetben: piros: #ff0000; fekete: #000000; szürke: #777777, #adadad, #424242, #8d8d8d; fehér: #ffffff) használata, így mi is e kombináció mellett döntöttünk. Az oldalon található design elemek is úgy vannak kialakítva, hogy autentikus érzést nyújtson a felhasználóknak, mintha egy virtuális edzőteremben járnánk.

2. A fejlesztői dokumentáció

2.1 Fejlesztőkörnyezet (hardver és szoftverek) ismertetése

A fejlesztés során számos olyan szoftvereket alkalmaztunk, amelyekkel már az évek során egészében megismerkedtünk és továbbá felhasználóbarát. S akár kevésbé „erős" hardvereken is akadálymentesen vagy legalábbis különösebb nehézségek nélkül tudják futtatni a hamarosan felsorolt szoftvereket. Ennélfogva a projekt fejlesztése során alkalmazott programok közé tartozik a, MySql Workbench, Xampp, Intellij Idea, Postman, Visual Studio Code és végezettül a GitHub Desktop.

MySql Workbench-et abból az okból kifolyólag használtuk, hogy a legegyszerűbben tudjuk felvázolni mit és hogyan szeretnék eltárolni az adatokat és ekképpen megvalósítani az adatbázisunk modelljét. Így már egy átlátható modellel és tervel a kezünkben sokkal gördülékenyebben lehetett elkészíteni az adatbázis magát. Amelyet a phpMyAdmin adatbázis kezelő rendszer segítségével alkottuk meg a Xampp szoftver hozzájárulásának köszönhetően.

A back-end és jUnit fejlesztésnél és tesztelésnél egyértelmű választás volt az Intellij Idea fejlesztőikörnyezet, s nem csak azért, mert már otthonosan tudtak a csapat tagjai benne mozogni, hanem mellette, a választott Java keretrendszer - a Spring Boot –támogatta azt. Azért a megnevezett keretrendszerrel írtuk, és nem sima Java nyelvben, mert könnyűszerrel lehet létrehozni és felépíteni egy back-end-et. Ez legfőképpen abban mutatkozik meg, hogy például JPArepository implementálása után könnyűszerrel tudjuk a tárolteljárásokat és különböző tervezett featureket létrehozni a back-end-ben. Azonban sajnos késői felfogás miatt ez nem valósult meg, így hagyományosan, az adatbázisban valósultak meg és futnak le a tárolteljárások. S meglátásunk szerint, sokkal kevesebb kódot igényel, hogy rendesen működjön és fusson le, nem úgy, mint a klasszikus Java nyelvnél.

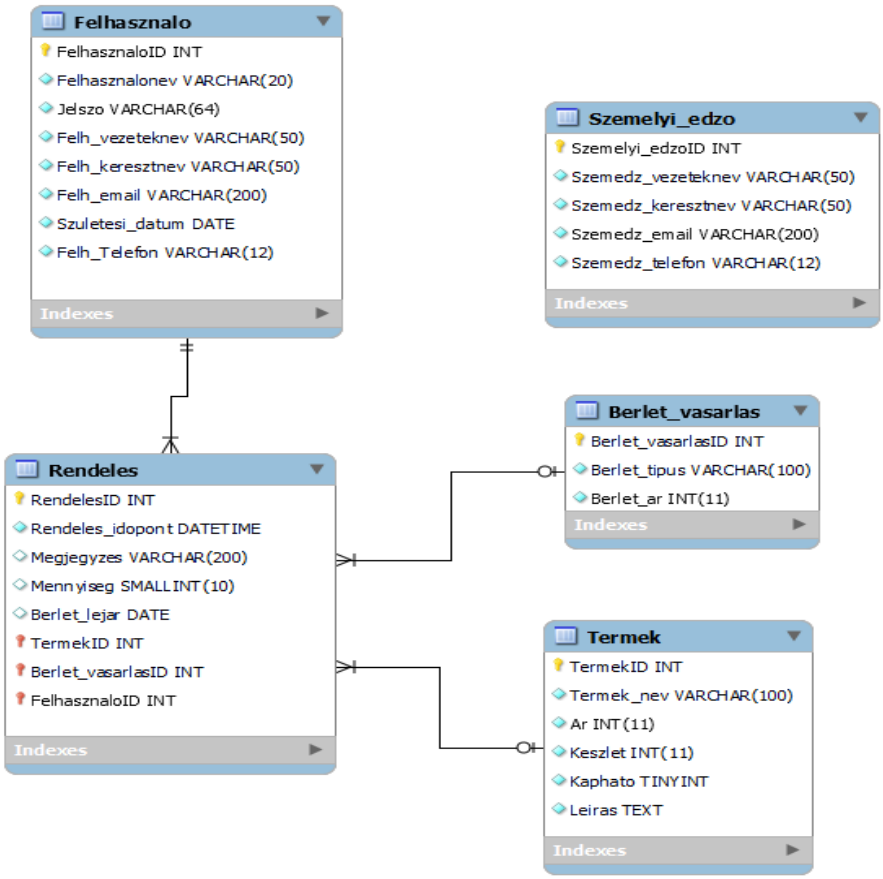
Mikor a back-end fejlesztésnél létrejöttek Rest API-k akkor azokat leszerettük tesztelni, hogy megbizonyosodjunk a hibátlan lefutásukról és az esetleg hibákat kiszűrjük mielőtt a front-end használhatná. Ebből kifolyólag a Postman nevezetű program jött kizárólag szóba. Ahol kimondhatatlanul könnyedén és különösebb nehézségek nélkül van lehetőségünk letesztelni mindenféle típusú API-okat. Ezen felül, ha security-t és authentikációt szeretnék készíteni a tervezett projektünkben, akkor is kiváló választás ez az adott program.

A front-end fejlesztéshez a Visual Studio Code nevezetű nyílt forráskódú kódszerkesztő lett igénybe véve, ahol Angular keretrendszerrel alkottuk meg magát az egész front-end-et és lett összekötve a back-end-el is. A választott back-end keretrendszert sajnálatos módon nem lehet összekötni a sima HTML, CSS, JS rendszerrel, ebből adódóan kénytelen voltunk az Angular keretrendszert magunkhoz fogni és abban megalkotni a projekt további részét, vagyis a front-end-et.

Annak érdekében, hogy mindenki kényelmesen hozzáférhessen a projekthez és meg tudjuk osztani a legfrissebb verziójú munkánkat egymással a GitHub-ot választottuk (a GitLab helyett) és ezen belül is a Desktop verzióját vettük számításba. A repository létrehozatala után minden készen állt, hogy igénybe vegyük azt és elkezdjük pusholni és pullolni a programot.

2.2 Adatmodell leírása

Ahogy korábban lett említve az adatbázis a phpMyAdmin-ban került megvalósításra, a képen látható modell pedig a MySql Workbench-ben.



1. kép: Land of Fitness Gym adatbázismodellje

Forrás: MySql Workbench, saját készítés

Összesen öt tábla került elkészítésre, ahol a szó szoros értelmében, vagyis leginkább a rendelés nevezetű tábla játszik kulcs fontosságú szerepet a táblák összekapcsolásánál. Az összes delete és update metódus „cascade”-ra lett állítva összekapcsolásnál, hogy minden zökkenő mentesen menjen. Az ábra pontos szemléltetése érdekében szeretném leírni a következőket: a sárga „izzó körte” a primary key-re utal, viszont a piros „izzó körte” pedig a foreign key-t jelzi, a kékre beszínezett négyzetek Not Null értékek, s az ennek ellentétje színtelen négyzet pedig Null értékéket is fogad hiba nélkül. Az adatbázisban kizárólag csak egy a többhöz kapcsolat jött létre. A személyi edző tábla abból az okból kifolyólag nem kapcsolódik semelyik táblához, mert a webapplikáció még nem tartalmazza ezt a funkcionalitást. Ebből adódóan a felhasználónak kapcsolatba kell lépnie az adott személyi edzővel, hogy időpontot tudjon foglalni a felhasználó. Az adatbázis többi része pedig egy kifejezett táblába kapcsolódik, ami nem más, mint a rendelés című tábla. Azonban egyszerre nem lehet terméket és bérletet vásárolni, csupán külön-külön van lehetőség megvásárolni a kiszemelt bérletet vagy terméket az oldalon. Tárolt eljárások segítségével például a felhasználó képes lesz megtudni, hogy a leadott rendelése mely termékhez tartozott és annak az adatait is olvasni lesz képes.

2.2.1 A táblák és az egyes mezői bemutatása

2.2.1.1 Felhasználó tábla

Ebben a táblában tároljuk mind azon felhasználónak az adatait, akik regisztráltak az oldalunkra. S az adott tábla tartalmaz egy FelhasznaloID amely primary key, Int, auto increment és not null tulajdonságokkal van ellátva. Minden felhasználónak automatikusan lesz generáltatva egy ID amely alapján belehet majd azonosítani az egyes felhasználókat.
Felhasználónév Varchar 20 karakteres (vagyis kizárólag 20 karakterből állhat a felhasználónév), unique, s not null. A felhasználónevet, azt maga a felhasználó állítja be magának tetszése szerint, megadott kritériumún belül a regisztrálás során. A felhasználónév az egyik olyan adat, amivel a felhasználó képes bejelentkezni a regisztrált fiókjába. Azonban abban az esetben, ha a megadott felhasználónév már létezik az adatbázisban akkor a felhasználó kénytelen lesz más, vagy módosítani a felhasználónevét.
Jelszó Varchar 64, not null, felhasználónév után a jelszó az utolsó szakasz, hogy be tudjon sikeren lépni a felhasználó a fiókjába. Sajnos jelenleg a fejlesztés nem jutott el odáig, hogy legyen titkosítva a felhasználó jelszava.
Felh_vezeteknev vagyis felhasználó vezetékneve egy Varchar(50), és not null, amely ugyanúgy a regisztráláskor kell megadni.

Felh_keresztnev azaz felhasználó keresztneve Varchar(50), végül not null és ezt is regisztrálás során kell be gépelni.
A felhasználó teljes neve eddig a rendeléskor fog megjeleni és magán a felhasználó saját oldalán, ahol nyilvánvalóan képes lesz kezelni az adatait.
Felh_email felhasználó email címe, Varchar(200), unique not null regisztráláskor elengedhetetlen feltétele, hogy meg legyen adva. Viszont, mint a felhasználónévnél, ha már megtalálható az adatbázisban, akkor a felhasználó kénytelen lesz más email címmel regisztrálni az oldalra.
Szuletesi_datum felhasználó pontos születési dátuma, amely Date és not null, az oka, hogy elvan tárolva ez a bizonyos adat, hogy felhasználó csakis azt a bérletet legyen képes megvásárolni, amely épp rá vonatkozik. Példaképp 22 évesen csak a „diák" és a „felnőtt" jelenjen meg a táblázatban vagy legalább rendelés leadásánál.
Felh_telefon felhasználó telefonszáma Varchar(12), not null magyar telefonszámhoz mérten lett megszabva, hogy hány karakterből állhat csak a telefonszám. Varchar-t használtunk mert legyen lehetőség „+" karaktert is belevinni a telefonszám megadáskor, ha esetleg a felhasználó regisztrálás során ekképpen szeretné megadni telefonszámát.

2.2.1.2 Személyi edző tábla

A személyi edzők manuálisan vannak felvéve az adatbázisba, maguk a személyi edzők is regisztrálhatnak az oldalra, azonban például jelen állás szerint, egy felhasználó csak az alapján tudd személyi edzőt maga mellé rendelni, ha kapcsolatba lép a megadott adatok egyikével, amely lehet telefonszám vagy email cím. Az összes adat egy személyi edzőről pontosan egy sorban lesz látható és olvasható.
Szemelyi_edzoID primary key Int, auto increment és not null minden személyi edzőhöz tartozik értelem szerűen egy ID, ami alapján lehet kezelni őket.
Szemedz_vezeteknev Varchar(50), not null, személyi edző vezetékneve.
Szemedz_keresztneve Varchar(50), not null személyi edző keresztneve.
Szemedz_email Varchar(200),unique not null személyi edző email címe, ahol kizárólag egyszer szerepelhet egy darab email cím az adatbázisban.
Szemedz_telefon Varchar(12), not null személyi edző telefonszáma. Itt is ugyanúgy, mint a felhasználónál a magyar viszonyokhoz mérten lehet begépelni a telefonszámot, akár tetszés szerint +36-os kezdéssel a 06 helyett.

2.2.1.3 Bérletvásárlás tábla

Ebben a táblában található az összes jelenleg elérhető és vásárolható bérlet az oldalunkon, mondhatni minden korosztály számára.
Berlet_vasarlasID primary key Int, auto increment és not null, tábla elsődleges kulcsa és ahogy korábban minden bérlethez tartozik egy ID.
Berlet_tipus Varchar(100), not null, és ebben az egységben találhatóak meg, hogy kiknek szól a bérlet, szóval példa kedvéért a felnőtt bérlet leginkább annak való aki nem bír diákigazolvánnyal. Terveztük, hogy az alapján jelenjenek meg a bérlet típus ahány éves maga a felhasználó. Azonban ezt még nem sikerült jelen állapotban kivitelezni.
Berlet_ar Int (11), unsigned, not null, bérletnek a bruttó szerinti ára található meg. Unsigned abból az okból került bele, hogy ne lehessen mínusz értéket megadni egy új bérletnél vagy módosításnál. Int típus pedig, hogy szabadon tudjuk változtatni az árakat komolyabb korlátozások nélkül.

2.2.1.4 Termék tábla

A táblában tárolódnak a különféle étrendkiegészítők számtalan adatai, amelyeket lehetőség van megvásárolni az oldalon, egy gomb megnyomásának a segítségével.
TermekID, primary key Int, auto increment, not null, a tábla elsődleges kulcsa. A rendelésnél kulcs fontosságú szerepet
Termek_nev Varchar(100), not null a termék pontos megnevezése.
Ar Int (11), unsigned, not null a termék áfával növelt ára.
Keszlet Int (11) unsigned, not null jelenleg mennyi van készleten egy adott termékből, ez redukálódik, amint a felhasználó vásárolt egy termékből.
Kaphato Tinyint/Boolean not null, ha esetleg egy termék jelenleg nincs készleten akkor azt egy bizonyos módon kilesz írva a felhasználónak.
Leiras Text, not null a termékről egy iromány, hogy mi is pontosan, esetleg, hogyan érdemes szedni, egyéb tulajdonságai egy adott terméknek stb.

2.2.1.5 Rendelés tábla

Egy bizonyos rendelés mely termékhez vagy bérlethez és felhasználóhoz tartozik, megjegyzéssel együtt némely esetekben, mivel az opcionális és időponttal együtt.

A rendelés az egyetlen olyan tábla, melyet nem lehet módosítani rendelés leadása után. Csak törölni, olvasni és létrehozni lehetséges. Az összes többi táblában mindazonáltal a módosítás funkció megtalálható.
RendelesID primary key Int, auto increment, not null, abban az esetben, ha egy felhasználó, bérlet típus vagy termék törlődik akkor ezzel párhuzamosan a hozzájuk tartozó rendelés ID is törlődni fog. Azaz az adatbázis nem tárolja el azokat a rendeléseket, amelyekhez már nem tartozik felhasználó, vagy bérlet, vagy termék.
Rendeles_idopont Datetime, not null, ez automatikusan generálodig másodperc pontosan, abban a pillanatban, amint a rendelés létre lett hozva a felhasználó által.
Megjegyzes Text, null, a termékrendelésnél opcionálisan lehet írni, abban a kapcsolatban, ha a felhasználó esetleg szeretne üzenni a rendelés ügyben.
Mennyiseg Smallint(10), null, a felhasználó megadhatja, hogy pontosan hány darabot szeretne rendelni az adott termékből. MySql és frontend álltal is be van korlátozva, hogy egyszerre maximum mennyit tud rendelni.
Berlet_lejar Date, null, a rendelt bérlet mikor jár le napra pontosan. Szám szerint 30 nap adódig a rendelés napjához.
TermekID Foreign key Int, null, termek nevezetű táblának az elsődleges kulcsa, viszont mint idegen kulcs a rendelés táblában.
Berlet_vasarlasID Foreign key Int, null, berlet_vasarlas táblából származó elsődleges kulcs.
FelhasznaloID Foreign key Int, not null, felhasznalo tábla elsődleges kulcsa.
Az idegen kulcsokhoz tartozó adatai láthatóak lesznek a rendelés leadását követően getById segítségével. TermekID és berlet_vasarlasID azért lehetnek null mert ne ütközőn hibába a rendszer mikor rendelés létrehozás történik a weboldalon.

2.3 Forráskód

A back-end felépítésének és fontosabb lépéseinek érdekében egy tábla vagy modell lesz górcső alá véve, ami a berlet_vasarlas táblára került a választás.
A model rétegjében vagy osztályában - mint az esetek többségében, mikor egy új osztályt deklarálunk – meg találhatóak az adatbázis alapján az attribútumok, ha például az adatbázisban varchar típus szerint lett elmentve egy adatmező, akkor egyértelműen a back-end-ben string-ént kell eltárolni ugyanazt. Konstruktor és getterek és setterek is megtalálhatóak. Azonban amely kifejezetten szemet tud azok a különböző annotációk és az osztály felett lévő hatalmas kód halmaz. Az annotációk:

- @ Entity – jelentése, hogy az adott osztály egy bizonyos táblához kötődhet.
- @ Table – itt van lehetőség megadni, hogy mely adatbázis táblához szeretnék kapcsolni az osztályt.
- @ NamedStoredProcedureQueries- a tárolt eljárások meghívása, abban az esetben kell, ha egy osztályhoz több getByID tárolt tartozik. Máskülönben pedig elég a @NamedStoredProcedureQuery annotációt alkalmazni.
- @ StoredProcedureParameter- azt a pontos megnevezést, típust és módszert várja be amit a fejlesztő megadott a tárolt eljárás készítésénél. Ha véletlenül rosszul írtuk volna be valamelyiket, akkor az alkalmazás exception-be fog futni.
- @ Id- az elsődleges kulcs megjelölésé, kötelező jelleggel.
- @ GeneratedValue-miként jöjjenek létre az ID-k, például ha „*Identity*” a típus akkor auto increment lesz az elsődleges kulcsra alkalmazva.
- @ Column- az oszlop megnevezése, lehet-e null és módosítható.
- @ Autowired- lehetővé teszi az objektumfüggőség hallgatólagos beillesztését

A „serializable” implementálása röviden arra való, hogy lehetővé teszi egy osztály állapotának mentését.

Abban az esetben, ha a tárolt eljárásokat az adatbázisban tároltuk és készítettük el, akkor a képen látható az egyik megoldás a getById típusú tárolt meghívására.

```
@NamedStoredProcedureQuery(name = "Berlet_vasarlasOlvasByID", procedureName = "Berlet_vasarlasOlvasByID",
        parameters = {
                @StoredProcedureParameter(mode = ParameterMode.IN, name = "berlet_vasarlasID", type = Long.class),
                @StoredProcedureParameter(mode = ParameterMode.OUT, name = "berlet_tipus", type = String.class),
                @StoredProcedureParameter(mode = ParameterMode.OUT, name = "berlet_ar", type = Integer.class),
        })
```

2. kép: Berlet_vasarlas model osztályának az egyik olvasás tárolt meghívása

Forrás: Spring boot Java, Intellij IDEA, Berlet_vasarlas.class

A megfelelő annotáció meghívása után azt az adott tárolt eljárást meg kell nevezni, és a paramétereit a minta alapján a szükséges annotáció segítségével létrehozni.

A repository package-ben jönnek létre és hívódnak meg gyakorlatilag a tárolteljárások az adatbázisból. Jparepository kiterjesztés arra add megoldást, hogy egyszerűen lehessen a tárolt eljárásokat létrehozni a back-end-ben, ahelyett, hogy az adatbázisban tennénk ezt. Számtalan beépített metódust foglal magába. Azonban ebben a projektben fordítva történt mind ez, késői ráeszmélés miatt. Ebben a package-en belül is számos annotáció található:

- @ Repository- Egy adatbázis repository-ként viselkedik az adott osztály, és különböző viselkedéseket utánozza.

- @ Query - Repository interfész metódusainak annotálására használható. Meghívhatjuk vele a tárolt eljárásainkat az adatbázisból.
- @ Procedure- get tárolt eljárások megnevezésénél használandó.
- @ Modifying- a törlés, módosítás és létrehozás tárolt eljárások meghívásoknál elengedhetetlen feltétele az annotáció jelenléte.
- @ Param- annoticáió által lehet megadni, hogy egy módosításnál vagy létrehozásnál milyen adatokat szeretnénk menedzselni.

A repository osztályok után a GymService a következő, ahol a maga a logika történik és ezen felül az összes tárolt eljárás megtalálható. Itt implementáljuk bele a tárolt eljárásokba vagy fejezzük ki, hogy milyen adatokkal térjenek vissza, mikor meg lesznek hívva. Annak érdekében, hogy -a példa modelnél maradva- bérletlétrehozásnál és szerkesztésénél ne lehessen null értéket megadni, vagyis üresen hagyni a mezőt az alábbi függvény lett létrehozva. Ez a bizonyos boolean függvény tulajdonképpel bevár egy objektumot paraméterként s azt ellenőrzi, hogy üres-e vagy sem. Ha esetleg valamely bemenő érték üresen maradt, akkor a saját készítésű exception hívodjon meg egy hibakóddal és üzenettel.

```
public static boolean isNull(Object obj) {
    return obj != null && !"".equals(obj);
}
```

4. kép: Függvény, mely megvizsgálja, hogy adott objektum null-e

Forrás: Spring boot java, Intellij Idea, GymService.class

A backend utolsó bástyája a controller rész, itt a projektben GymController néven található meg. Ebből az osztályból van lehetősége a front-end-nek meghívnia a tárolt eljárásokat és azokat használni, az API url-k által. Ezen a controlleren belül is fellelhetők újabb annotációk és egy függvény is. A függvény hasonlóan, mint a Service rétegben, itt is nagyjából azt vizsgálja meg, hogy üres-e érték található a megadott Id-ú iterable-nél (iterálható objektumok gyűjteménye – jelentése ismételhető). S ha itt is hibába ütközik, akkor meg hívódig az exception. A választás az iterable mellett (és nem a lista lett használva), hogy getById eljárásoknál egy Id-hoz tartozhat egynél több adat, vagyis például 1 bizonyos bérlet Id-hoz tartozhat számtalan rendelés és ezt a lista nem képes kiíratni csak az iterable. Mivel egynél több adat után hibába fut.

GymControllerben fellelhető új annotációk:

- @ RestController-A RESTful webalkalmazások létrehozásához használt annotáció.
- @ RequestMapping- A webes kérések meghatározott kezelőosztályokra és/vagy kezelői metódusokra való leképezésére szolgál.
- @ GetMapping, @PostMapping, @DeleteMapping, @PutMapping-Az API-k megjelölése, hogy mely http feladatot fogják elvégezni, és az elérési útvonalukat.
- @ PathVariable- Az API url végén mely változott várja be bemenő paraméterként.
- @ RequestBody- A megjelölt API payload-ot vár be, mikor meghívódig.

```
public static void isEmpty(Iterable<?> iterable) {
    int counter = 0;
    for (Object obj : iterable) {
        counter++;
    }
    if (counter == 0) throw new ApiNotFoundException("Nem található érték az adott ID-nál!");
}
```

5. kép: Függvény, mely megvizsgálja, hogy adott iterable üres-e

Forrás: Spring boot java, Intellij Idea, GymController.class

2.4 Tesztelési dokumentáció

A back-end teszteléséhez a jUnit modulteszelő rendszert használtuk, ahol pár API került letesztelésre plusz egy exception féle teszt, amelyek mindegyike hibátlanul lefut. Az említett exception-ös teszt, nevén szólítva a „findTermekByIDNotFound()” metódus nem vár be semmilye paramétert az API meghívásakor. Kizárólag az a feladata vagy képes ellenőrizni, hogy ami manuálisan „termekID” be lett égetve előzőleg a „findTermekByID()” metódusnál az létezik-e egyáltalán az adatbázisban. Ha nem található az adott ID adat, akkor 404-es nem található hiba kód fog kiírodni a konzolban.

Create és update API-k abból az okból nem kerültek tesztelésre, mert bármely módszer megvalósításánál hibába ütközött. S gyakorlatilag már kifogytunk az ötletekből hogyan lehetne működőképessé varázsolni őket.

Mind a kettő eljárás „*entity must not be null*” hibába futott, ahol kis olvasás után, az lehetett a gond, hogy valamely adat hiányzik vagy rosszul lett megadva. De a mi esetünkben minden megfelelően lett meghívva és használna. S ez által időszűkében kénytelen lettünk feladni és más feladatokra összpontosítani.

2.5 Továbbfejlesztési lehetőségek

A webalkalmazás fejlesztése során nagy mennyiségű feature-t terveztünk és szerettünk volna végbe vinni. Sajnálatos módon az idő hiányában és kevés szakmai tapasztalatnak köszönhetően nem sikerült az elképzeléseink szerint minden egyes ötletet megalkotni.

Szerettünk volna egy QR kódolvasó rendszert a bérletekre, amelyet kilehetett volna nyomtatni (vagyis magát a bérletet, amelyen az egyéni QR szerepelt volna). Ily módon lehetősége lett volna a felhasználónak nemes egyszerűséggel be szkennelni bérleten található QR kódot és megtudnia, hogy az adott bérlet jegye még meddig érvényes még. Természetesen ezen kívül a QR tartalmazta volna a felhasználó összes adatát és a bérlethez kapcsolódó fontosabb információkat.

Telefonos alkalmazás, amelyben szintén lehetett volna termékeket és bérleteket vásárolni, utóbbit digitális formában akár a telefonon kijelzett QR kód segítségével, valamint üzenőfelületet az edzőpartnerekkel, egy edzőnaplót, ahova feljegyezhetjük edzéseinket, nyomon követhetjük teljesítményünket és különböző szimbolikus digitális díjakat, amelyeket teljesítményünkért személyi edzőnk adhat nekünk az alkalmazásban. Asztali alkalmazás a személyzet számára, hogy könnyű legyen a készletek információk kezelése.

Különböző role-ok bevezetése, hogy gördülékenyebben lehessen használni az oldalt, admin, sima felhasználó, személyi edző jogok. Admin joggal rendelkező felhasználóknak nyilvánvalóan lett volna lehetősége minden tárolt eljárás vagy feature használatára. Felhasználók és a személyi edzők kizárólag a nevükből fakadóan csak azokat a funkciókat lehettek volna képesek használni a weboldalon.
A különböző jogok után szerintem elmaradhatatlan a security, legalább a legalapabb biztonság, hogy valami szinten a felhasználók adatai lelegyenek védve. Belépéskor a token lelegyen kódolva és maga jelszó is. Véletlenül se legyen olvasható a cím sorban és a konzolban.

Blob fájlok kezelése, azon belül is legfőképpen képek menedzselésének a lehetősége. A felhasználónak, személyi edzőnek és a termékeknek legyen saját egyedi képe, avatárja, portréja, amivel könnyedén felismerhetőek tudtak volna válni. S ezt különösebb korlátok nélkül lettek volna képesek változtatni. Az egyetlen kritérium a képarány, ami pontosan meglett volna határozva.

A következő a listán, hogy a felhasználónak legyen arra lehetősége, hogy megadott szabad időpontok alapján tudjon személyi edzőt lefoglalni magának. Anélkül, hogy kapcsolatba keljen neki lépnie a személyi edzővel az ügy érdekében. Természetesen ettől függetlenül a személyi edző azon adatai továbbra is megtalálhatóak lehetnének, amellyel korábban kapcsolatba lehetett lépni. Ezen belül a személyi edzők fel legyenek osztva, akik csak személyes órát tartanak, és akik csak csoportos órát, akik csoportos edzők lennének. Értelemszerűen lenne olyan edző, aki mind két tábort is erősítené.

Rendelésnél lehessen ingyenes házhozszállítással is kérni a felhasználónak a megrendelt árukat, amivel bejönne egy újabb tábla például tartózkodási hely címmel. Ez a funkció csak egy bizonyos km hatótávolságban lenne engedélyezve, akik kívül estek azoknak csak a személyes átvétel lehetőség lenne az edzőteremben.

Termékekről legyen lehetőség értékelést írni a felhasználóknak és akár ötös skálán csillagozni, hogy mennyire volt megelégedve a termékkel. Ráadásul minden terméknek külön oldalt biztosítani, ahol kényelmesen és rendezetten lennének leírva alaposan a termékkel kapcsolatos információk és esetleg, hogy mely cég hozza forgalomba vagy gyártja Magyarországon.

Közös véleményünk alapján úgy gondoljuk, hogy a felsoroltak összese érdemes megvalósítani a jövőben, némelyek kötelező jelleggel is, de csomó színesebbé és felhasználó barátabbá tudná varázsolni a weboldalt.

2.6 Irodalomjegyzék, forrásmegjelölés

Spring Boot back-end fejlesztése és megértése

https://www.youtube.com/watch?v=oTosinxsR5g&t=1523s

https://www.youtube.com/watch?v=Gx4iBLKLVHk&t=785s

Spring Boot exception handling elkészítése

https://www.youtube.com/watch?v=PzK4ZXa2Tbc

jUnit tesztelés megvalósítása

https://stackabuse.com/guide-to-unit-testing-spring-boot-rest-apis/

https://mkyong.com/spring-boot/spring-rest-integration-test-example/

Angular front-end elkezdése és összekapcsolás a back-end-el

https://www.youtube.com/watch?v=Gx4iBLKLVHk&t=785s

Front-end design elemek

https://www.w3schools.com/howto/howto_css_modals.asp

Záródolgozat témák, szempontok, ötletek

https://docplayer.hu/8351479-Szakdolgozat-kovetelmenyek.html

http://infojegyzet.hu/webszerkesztes/zarodolgozat/

3. Felhasználói dokumentáció

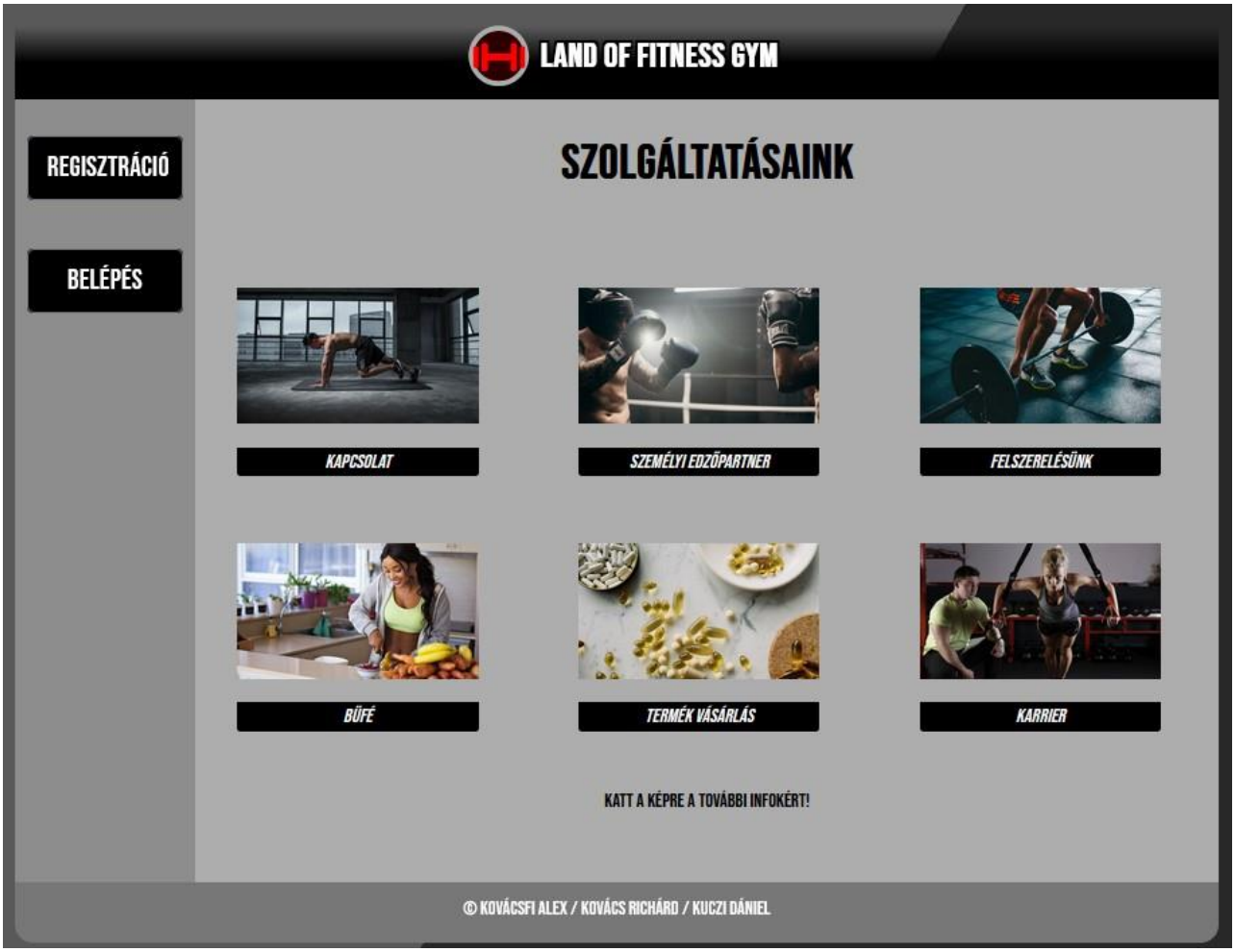
A program egy webapplikáció vagy magyarán egy edzőteremnek az oldala. Ezen a weboldalon jelenleg lehetőség van vásárolni bérletet, éttrendkiegészítőket és személyi edzőkkel kapcsolatba lépni. Csak azután nyílnak meg ezek a lehetőségek, ha regisztrált valaki. Mivel máskülönben nem lesznek elérhetőek, azonban az információk nagy részéhez regisztráció nélkül is hozzá lehet férni.

A weboldal képes ellátni a legalapvetőbb eljárások közül létrehozni és olvasni adatokat. Természetesen megfelelő jogosultságok mellett érhetőek el ezek a lehetőségek. A weboldal nem igényel többet, mint egy működő számítógépet és egy támogatott webböngészőt. A weboldal reszponzív így a legtöbb képernyőméretet támogatja a hagyományos monitoroktól a telefonokig.

Az oldal betöltésekor a főoldalra kerülünk, ahol elénk tárul az oldal kidolgozása. Látjuk a fejlécet azon belül a címet és az edzőterem logóját, egy oldalsó navigációs sávot benne gombokkal, melyek további oldalakra kalauzolnak bennünket és középen a tartalmi részt, ahol további hat képet és leírást, amik szintén további oldalakra irányítanak át. Az oldal alján pedig láblécben található az edzőterem címe. Itt a főoldalon számos lehetőség közül választhatunk. Az oldalsó navigációs sávban megtalálható regisztráció gomb. A regisztrációs gombot megnyomva egy regisztrációs felületre kerülünk, ahol az oldal számos adatot kér tőlünk.

Regisztráció során meg kell adni, egy felhasználónevet amely 20 karakterből állhat, vagyis 20 betűből. Ha már egy másik felhasználó regisztrált azzal felhasználónévvel amit magunknak kitaláltunk, akkor ebben az esetben sajnos újat kell választanunk. Jelszó megadásnál 64 tetszőleges karaktert lehet bevinni, vezetéknév, keresztnév megadásnál 50-50. Email címnél ugyanaz az a helyzet, mint a felhasználónévnél, hogy ha már létezik akkor nem enged regisztrálni, mellesleg itt pedig 200 hosszúságú email címet is belehet vinni. Telefonszám az a magyar viszonyokhoz mérten lett beállítva, vagyis 12 karakter hosszú (+ jel miatt), ahol minimum 11 karaktert bekell írni. Természetesen nem kötelező valóban 50 karakternyi hosszúságú vezetéknevet beírni. Egyedül a telefonszámnál van a korlátozás.

Az adatok megadása után a „regisztrálok” feliratú gombra kattintva adataink mentésre kerülnek, vagyis sikeresen regisztráltunk. A főoldalon hat további lehetőség található: havi bérlet, személyi edzőpartner, felszerelésünk, büfé, étrendkiegészítőink, karrier.



6. kép: Land of Fitness Index oldala

Forrás: Angular, Visual Studio Code, fooldal-component

Ezeken belül információ és további funkciók -például termékrendelés- találhatóak. Az információ mindenki számára elérhető, viszont a rendeléshez aktív felhasználói fiók szükséges. A havi kapcsolat képre kattintva eljutunk a kapcsolati oldalra, ahol a nyitvatartással találkozhatunk, illetve az edzőterem címével. A személyi edzőpartner képre kattintva az oldal átirányít bennünket a személyi edzőpartner oldalra, ahol kapcsolattartási információt kaphatunk az edzőteremben dolgozó edzőkről. Ezek az információk a következőek: vezetéknév, keresztnév, email cím, telefonszám.

A felszerelésünk képre kattintva meglátogathatjuk a felszerelésünk oldalt, ahol pontos lista található az edzőterem igénybevehető eszközeiről.

A büfé képre kattintva megtekinthetjük a büfé oldalát, ahol lehetőségünk van információt szerezni a büfében kínált termékekről, azok tartalmáról és a lehetséges allergénekről. Étrendkiegészítőink képére kattintva eljutunk a bejelentkező felületre. A bejelentkező felületen már csak a felhasználónevet és jelszót kell megadnunk, amellyel regisztráltunk.

Ez után a bejelentkezés gombra kattintva elkerülünk a vásárlás oldalra regisztrált felhasználóként, ahol étrendkiegészítő termékeket, illetve havi bérletet vásárolhatunk, utóbbit kategóriák szerint. Ezek a kategóriák: diák, felnőtt, nyugdíjas, csecsemő. A „bérletet vásárolok" feliratú gombra kattintva egy előugró ablakban adhatjuk meg a mennyiséget, majd a rendelés gombra kattintva meg is rendeltük a kiválasztott kategóriát, továbbá a megjegyzés mezőbe bevihetünk egy maximum 50 karakter hosszú megjegyzést vásárlásunk mellé. A termékvásárlás gombra kattintva egy előugró ablakban az adott termék nevét kiválasztva adhatjuk meg a mennyiséget, ismét egy maximum 50 karakter hosszú megjegyzést, illetve az ablak elkéri egyedi felhasználói azonosítónkat, majd a rendelés gombra kattintva véglegesíthetjük vásárlásunkat.

A karrier oldalra kattintva meglátogatjuk a karrier oldalt, ahol az edzőteremről találunk kapcsolattartási információt. Ezen oldalak elrendezése mind megegyező, illetve kisebb eltérések vannak csupán. Az oldalon a már megismert fejléc alatt oldalsó navigációs sáv mellette a fő tartalom rész, ebben egy kép található alatta rövid leírás, ez alatt pedig információ listába illetve táblázatba szedve.

Egyes táblázatokban különböző információkra kattintva rendelhetünk terméket. Ilyen táblázat például a bérlet és a termék oldalon található.

Az oldalsó navigácós sávból bármely oldalról ellátogathatunk bármely másik oldalra az annak megfelelő gomb lenyomásával.

Képjegyzék