**Szakképesítés neve:** Szoftverfejlesztő és –tesztelő

**Szakma azonosító száma:** 506131203

**Szakdolgozat**

**Címe**

**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. Elérhető lesznek a személyi edzők, ahol minden fontos információ meg lesz található róluk, például amelyekkel kapcsolatba lehet velük lépni. Étrendkiegészítő termékek (raktári mennyiséggel) és számtalan bérletek vásárlásának a lehetősége.

**A fejlesztői dokumentáció**

Fejlesztőkörnyezet (hardver és szoftverek) ismertetése és választás indoklása

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 mint a klasszikus Java nyelvnél.

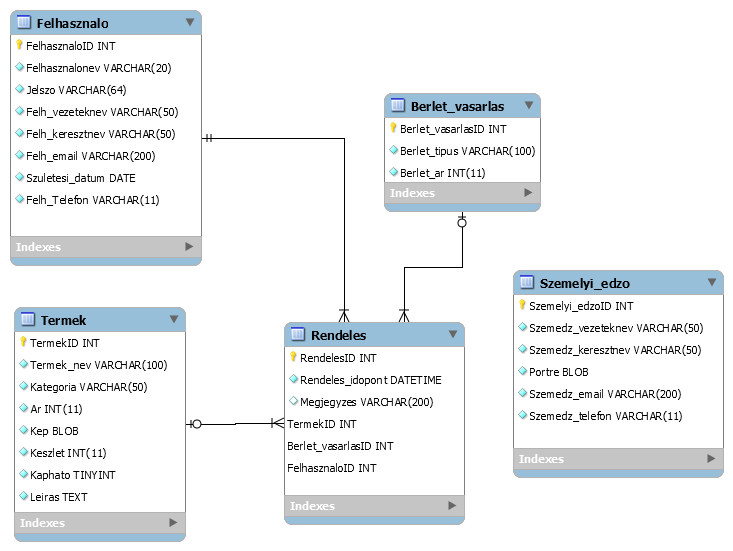
Mikor a back-end fejlesztésnél létrejöttek Rest API-k akkor azokat leszerettük tesztelni, hogy megbizonyosodjunk a hibátlan lefutásukról és az esetleg hibákat kiszűrjük mielőtt a front-end használhatná. Ebből kifolyólag a Postman nevezetű program jött kizárólag szóba. Ahol kimondhatatlanul könnyedén 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.

**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, 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.

**A táblák és az egyes mezői bemutatása**

**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.

**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.

**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.

**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.

**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 amelyt 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.

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.

**Részletes feladatspecifikáció, algoritmusok**

**Forráskód**

**Tesztelési dokumentáció**

**Továbbfejlesztési lehetőségek**

A webalkalmazás fejlesztése során nagy mennyiségű feature-t terveztünk és szerettünk volna végbe vinni. Sajnálatos módon az idő hiányában és kevés szakmai tapasztalóknak 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 bérlett jegye még meddig érvényes még. Természetesen ezen kívül a QR tartalmazta volna a felhasználó összes adatát és a bérlethez kapcsolódó fontosabb információkat.

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 használata. 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.

Ezt a gondolatot folytatva Blob fájl vagyis kép fájl kezelése. A felhasználónak, személyi edzőnek és a termékeknek legyen saját képe, avatárja, portréja, amivel könnyedén felismerhetőek tudtak volna válni. S ezt különösebb korlátok nélkül tudták volna változtatni. A képarány pontosan meglett volna határozva.

A következő a listán, hogy a felhasználónak legyen arra lehetősége, hogy megadott szabad időpontok alapján tudjon személyi edzőt le foglalni 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 mint két tábort erősíteni.

Rendelésnél lehessen ingyenesen 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.

**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

**Záródolgozat témák, szempontok, ötletek**

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

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

**Felhasználói dokumentáció**

A program egy webapplikáció vagy magyarán egy edzőteremnek az oldala. Ezen a weboldalon jelenleg lehetőség van vásárolni bérletet, éttrendkiegészítőket és személyi edzőket felbérelni. Csak azután nyílnak meg ezek a lehetőségek, ha regisztrált valaki. Mivel máskülönben nem lesznek elérhetőek. A weboldal ezen felül képes ellátni a legalapvetőbb eljárásokat, törlés, módosítás, létrehozás és végül adatok olvasása. Természetesen megfelelő jogosultságok mellett érhetőek el ezek a lehetőségek. A weboldal nem igényel többet, mint egy működő számítógépet és egy támogatott webböngészőt.