**Baranya Megyei Szakképzési Centrum**

**Baranya Megyei SzC Radnóti Miklós Közgazdasági Technikum**

**Vizsgaremek**

Készítették: Pópé Dániel

és Kapus Benjámin

**Pécs**

**2023**

**Baranya Megyei Szakképzési Centrum**

**Baranya Megyei SzC Radnóti Miklós Közgazdasági Technikum**

Szakma megnevezése: Szoftverfejlesztő és –tesztelő

A szakma azonosító száma: 5 0613 12 03

**Vizsgaremek**

Szakikereső

Készítették: Pópé Dániel

és Kapus Benjámin

**Pécs**

**2023**

Tartalomjegyzék

[I. Bevezető, a feladat rövid ismertetése 4](#_Toc132298484)

[II. A felhasználói dokumentáció 4](#_Toc132298485)

[1. A program általános specifikációja 4](#_Toc132298486)

[2. Rendszerkövetelmények 4](#_Toc132298487)

[• Hardver követelmények 4](#_Toc132298488)

[• Szoftver követelmények 4](#_Toc132298489)

[3. A program telepítése 5](#_Toc132298490)

[4. A program használatának a részletes leírása 5](#_Toc132298491)

[III. A fejlesztői dokumentáció 5](#_Toc132298492)

[1. Témaválasztás indoklása 5](#_Toc132298493)

[2. Az alkalmazott fejlesztői eszközök 6](#_Toc132298494)

[3. Tervezési módszer 6](#_Toc132298495)

[4. Adatmodell leírása 6](#_Toc132298496)

[5. Részletes feladatspecifikáció, algoritmusok 6](#_Toc132298497)

[6. Forráskód 7](#_Toc132298498)

[7. Tesztelési dokumentáció 7](#_Toc132298499)

[8. Továbbfejlesztési lehetőségek 7](#_Toc132298500)

[IV. Összegzés 7](#_Toc132298501)

# I. Bevezető, a feladat rövid ismertetése

A vizsgaremekhez több projektötletünk volt, viszont a túlzott bonyolultság és a követelmények alapján szépen kiszűrtük az ötleteket és végül erre a témára esett a választásunk.

A mai világban az emberek törekednek a technológiák fejlesztésére, a dolgok praktikusabbá tételére. Ennek egyik hátránya, hogy minden egyre bonyolultabb és összetettebb lesz, ezért mindenhez szükség van szaktudásra vagy speciális eszközökre. A dolgok elhasználódnak, elavulnak, vagy tönkre mennek.

Bizonyára mindannyian, vagy legalábbis a családban, ismerettségi körben előfordult, hogy felfogadtunk valamilyen szakembert. Ezt sok keresés, kevés választási lehetőség és akár több hét, hónap várakozás előzte meg. Emellé természetesen hozzátartozik az is, hogy a szakember milyen munkát végzett. Sajnos gyakran hallunk, megbízhatatlan, munkáját rosszul elvégző emberekről, akik nagyobb kárt okoztak, mint amennyi hasznot. Ezekre a problémákra szeretnénk megoldásként ezt a projektet.

A fejlesztés során rengeteg új ismeretet szereztünk, mint például a kód lefutásakor keletkezett hibák megoldásáról. Emellett több olyan része is volt a projektnek, amivel az iskolában nem találkoztunk és ezért saját magunknak kellett rájönni, megkeresni a megoldást. Összességében egy jó „tanulópénz” volt ez a projekt.

A projekttel kapcsolatban felmerült az, hogy publikus és elérhető legyen mindenki számára az interneten, viszont a tapasztalatlanság és a tőke hiánya miatt arra a döntésre jutottunk, hogy erre nem kerül sor. A projekt viszont teljes mértékben tovább fejleszthető, ötletekre a későbbiekben kitérek.

A feladatok elosztását alapvetően két részre osztottuk: frontend és backend. Az ötletelési, tervezési szakasznál jobbnak találtuk, hogy teljesen együtt dolgozzuk ki az adott feladatot. Az adatbázis-táblák is így jöttek létre. Ezt követően a kinézettel és az azokhoz tartozó, böngészőben megvalósítható funkciókkal Pópé Dániel foglalkozott. Az adatbázis eljárások készítésével és az adatbázis kapcsolat létrehozásával Kapus Benjámin dolgozott. Természetesen végig kommunikáltunk és ahol tudtunk besegítettünk a másiknak.

A projektmunka során több projektszervezési programot is használtunk. Például Trello-t az ötletek rendszerezéséhez, GitHub-ot a projekt tárolására és verziókezelésére.

# II. A felhasználói dokumentáció

A felhasználói dokumentáció célja, hogy segítséget adjon a leendő felhasználónak a program telepítésével és használatával kapcsolatos minden probléma megoldásában, segítse a program kezelésének az elsajátítását.

A felhasználói dokumentációnak a következő részeket kell tartalmaznia:

## 1. A program általános specifikációja

Ez a rész a program fontosabb jellemzőit és funkcióit tartalmazza. A cél, hogy a leendő felhasználó ezt a fejezetet elolvasva el tudja dönteni, hogy a program megfelelő-e a számára.

Ajánlott terjedelem: 1-2 oldal.

## 2. Rendszerkövetelmények

### • Hardver követelmények

Ebben a részben kell leírni a minimális és ajánlott hardver konfigurációt, amely a program futtatásához szükséges. Pontos paramétereket kell megadni, még akkor is, ha a program amúgy minden gépen lefut.

### • Szoftver követelmények

Le kell írni, hogy mely operációs rendszere(ke)n fut a program, és milyen egyéb szoftver komponensek szükségesek a működéshez (pl. .NET, DirectX, esetleg adatbázis-szerver és adatbázis-állományok, stb.) A beadott CD-n ezeknek is ott kell lenniük.

Ajánlott terjedelem: ½ -1 oldal, felsorolásszerűen leírva

## 3. A program telepítése

• Képekkel illusztrált, részletes leírás a program telepítésének a menetéről.

• A leírás alapján a felhasználónak hiba nélkül telepíteni kell tudni a programot. A leírásnak ki kell térnie a telepítés során kiválasztható opciókra is.

• Ha esetleg nincs telepítőprogram, akkor kellő részletességgel le kell írni, hogy mely fájlokat, pontosan hova kell felmásolni, és hogy lehet a programot futtatni.

Ajánlott terjedelem: 2 -4 oldal, ábrákkal együtt.

## 4. A program használatának a részletes leírása

Mindenre kiterjedő, részletes leírás a program használatáról.

Alapszabályok:

• Amit leprogramozott, azt a dokumentációban is írja le, ne legyenek eltitkolt funkciók.

• Minden pontosan, „szájbarágósan” legyen leírva. A dokumentáció alapján a teljesen kezdő, vagy laikus felhasználóknak is használniuk kell tudni a programot.

• A stílus legyen pontos és közérthető, vegye figyelembe, hogy a felhasználói dokumentáció nem szakembereknek készül.

• Ugyanakkor kerülje a laza stílust: rövidítések, smilie-k, szleng kizárva.

• Alkalmazzon ábrákat, screenshot-okat , de a ne legyen túlzott a képek aránya a szöveghez képest. Kb. oldalanként egy ábra megfelelő (nem nagy).

Ajánlott terjedelem: 12-17 oldal, ábrákkal együtt.

# III. Fejlesztői dokumentáció

A kód írásakor törekedtünk a „clean code” elveinek megfelelni. Ennek fényében a lehető legegyszerűbb, legrövidebb és legjobb megoldásokat alkalmaztuk. A sorok, függvények köré megjegyzéseket is tettünk a még gyorsabb megértés érdekében. Az eredményen látszik, hogy könnyen átlátható és megérthető kódot sikerült készítenünk.

## 1. Témaválasztás indoklása

A téma kiválasztásának fő szempontjai természetesen a vizsgaremek elvárásai voltak, viszont fontosnak tartottuk, hogy hozzánk közel álló, ismert problémára, feladatra találjunk megoldást. Ezért is sikerült egy életszerű problémára megoldást nyújtó oldalt létrehozni.

Adatbázist is használ, mivel a felhasználók, a hirdetések és sok más adatot el kell tárolnunk. Ezekhez az adatokhoz rengeteg rész tartozik: A regisztrációtól kezdve a hirdetések lekérésén át az értékelések írásáig bármi. Vannak alap és különleges funkciók. Tehát sok lehetőség és elképzelés volt. Ezekre a függvényekre könnyen rálehetett építeni a RESTful komponenseket is.

Ezen kívül volt rendelkezésre álló megjeleníthető tartalom is, mint például a hirdetések, kedvencek, profil adatok. Ezekhez az elemekhez könnyű volt a megjelenítést kialakítani, ami különböző méretű eszközökön is jól látható és áttekinthető.

## 2. Az alkalmazott fejlesztői eszközök

* Programnyelvek
  + Adatbázis: MySql
  + Frontend: Angular
  + Backend: Java
* Fejlesztői környezet
  + Adatbázis: Xampp (PHPMyAdmin), dbdiagram.io
  + Frontend: Visual Studio Code, PostMan, Paint
  + Backend: Netbeans, PostMan, WildFly szerver
  + Mindegyik: Git, GitHub Desktop

## 3. Tervezési módszer

• Alkalmazás tervezése,

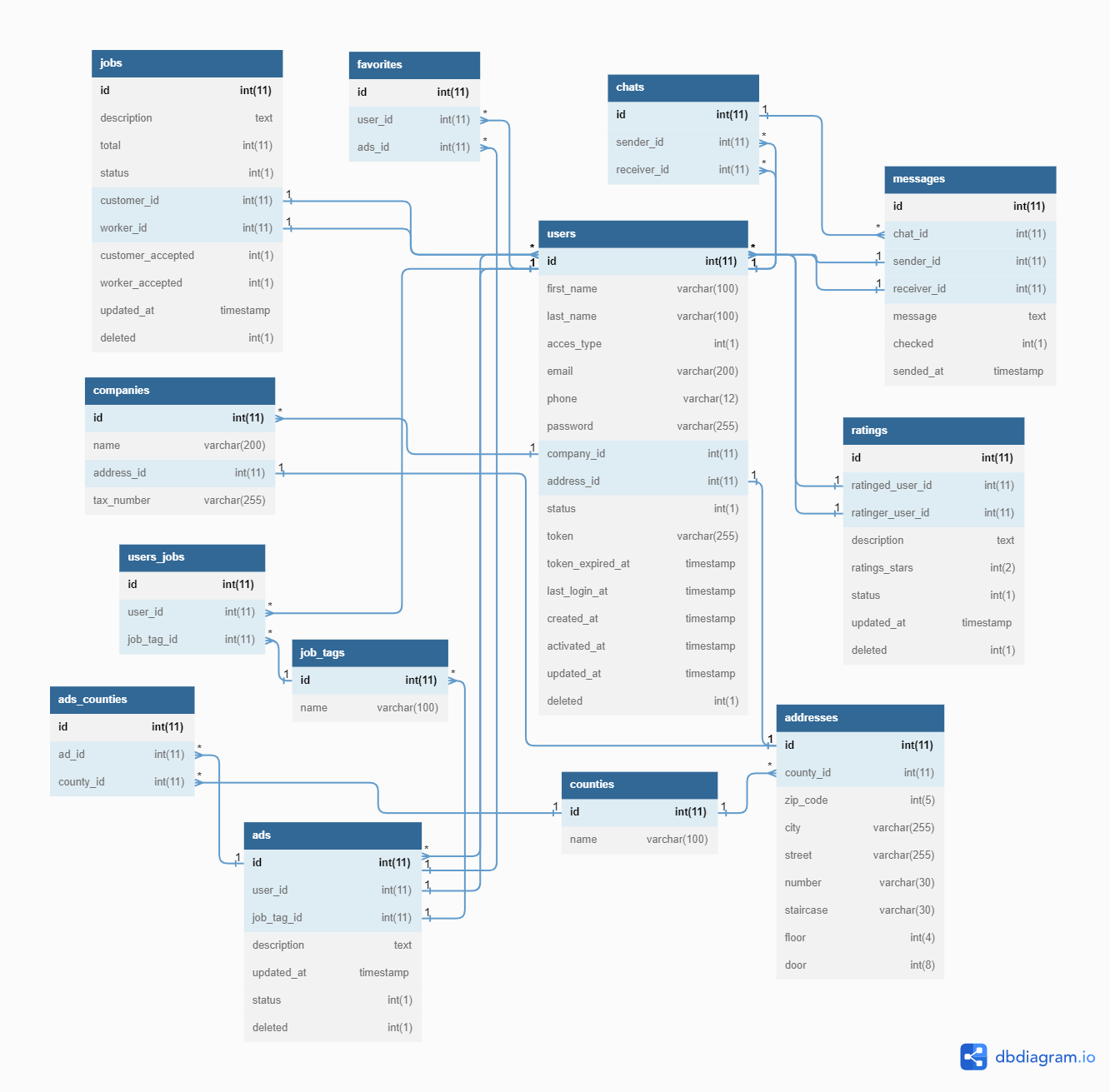
• tervezési módszertan ismertetése,

• vizuális modellezés

• OOP megvalósulásának leírása, részletezése

Ajánlott terjedelem: a feladat jellegétől függően 2-5 oldal.

## 4. Adatmodell leírása

Az adatbázis tervezéséhez a dbdiagram nevű, adatbázismodell-diagram készítő oldalt választottuk. Ezzel a diagram szerkesztővel könnyen átalakíthattuk és rendezhettük a táblákat, oszlopokat és kapcsolatokat. Az 1. ábrán látható a teljes adatbázis az adattáblák összekapcsolásával.

*1. ábra Adatbázis adatbázismodell-diagramja*

Az alábbiakban részletezem az 1. ábra diagramjának a részeit és az adatbázis szerkezetét.

Mindenhol próbáltunk egyértelmű, egyszerű és rövid elnevezéseket használni. Az adatbázist „szakemberkereso”-nek neveztük el, így egyértelmű, hogy ehhez a projekthez tartozik. A táblák nevei az angol megfelelője annak, amit benne tárolunk (többeszszámban), mint például a hirdetéseknek „ads” és a felhasználóknak „users”. Amennyiben több a többhöz kapcsolat van az adatok között ott kapcsolótáblát alkalmaztunk. Ezeknek a nevét a két összekapcsolt tábla alkotja egy alsó vonallal elválasztva. Több helyen is előfordulnak azonos oszlopnevek, de csak ha ugyanazt a szerepet töltik be. Úgy gondoltuk, hogy ezeket nem írjuk le külön-külön minden táblához, hanem itt az elején részletezzük őket.

Természetesen mindenhol található elsődleges kulcs, amit „id”-nak neveztünk el. Ez a mező szám típusú és maximum 11 karakter hosszú lehet. A táblák közötti kapcsolatokat, ha egy másik táblában szerepelt idegen kulcsként, általában a tábla neve egyesszámban, egy alsó vonal és az „id” részek alapján neveztük el. Például „ad\_id”.

A következő mező a „status”, ami szám típusú és csak 1 karakter lehet. Az adatbázisunknak két formája van:

* A „users” táblában 3 értéke lehet a következő párosításban:
  + „-1”, ha a felhasználó regisztrált az oldalon, viszont a megerősítő E-mail-ben a linket még nem nyitotta meg.
  + „0”, amennyiben az előző pontban említett linket megnyitva aktiválta az E-mail címét, viszont az oldalra ebben a pillanatban nincs bejelentkezve a felhasználó, tehát „offline”.
  + „1” akkor lehet, ha megerősítette a regisztrációját és éppen bejelentkezve van a felhasználó, „online” állapotban van.
  + Ha egyszer megerősítette a regisztrációt, az utolsó két érték között váltakozhat a „status” tulajdonság. Ezt a „login” és „logout” (bejelentkezés - kijelentkezés) tárolt eljárások biztosítják.
* A „ratings”, „ads” táblákban két érték lehet: „0” és „1”. Egy értékelés vagy hirdetés létrehozásakor, vagy módosításakor alapértelmezettként a „0” értékre állítjuk, ami azt jelenti, hogy nincs ellenőrizve, elfogadva. Ezt egy admin jogosultságú felhasználó tudja elfogadni, miután megnézte és nem talált kifogásolható tartalmat benne. Miután elfogadta a „acceptRating” vagy „acceptAd” eljárást meghívva „1”-esre változik az érték.
* A „jobs” táblában bár értékek terén nem változik az előző ponthoz képest, jelentésben egy kicsit eltér. Itt a „0”-hoz a nincs kész az „1”-hez pedig a kész jelentés van rendelve. A szakember tudja kész állapotba helyezni a „changeJobStatus” procedúrával.

Egy másik gyakran használt tulajdonság a „deleted”. Ennek a szerepe, hogy tudjuk törölve lett-e az adott sor. A „0” a nem törölt, az „1” a törölt.

A „jobs” táblában mindkét fél el tudja fogadni a „szerződést”. „worker\_accepted” és „customer\_accepted” mezőnevekkel vannak ellátva. Két értéke lehet: „0”, ami azt jelenti, hogy nem fogadta el a feltételeket és az „1” amikor el van fogadva.

Az első és egyben a legfontosabb tábla a „users”. Ebben a felhasználók adatait tároljuk. A nevét két részben, vezetéknévre és keresztnévre lebontva. Mindkettőt szöveg formában tároljuk és legfeljebb 100-100 karakter lehet. Szöveges formában tároljuk még az E-mail címet 200 és a telefonszámot 12 karakterig. A felhasználó jogosultságát az „access\_type” tulajdonság alapján kezeli az adatbázis, a backend és a frontend is. 1 karakteres szám formájú lehet: „0” – Általános felhasználó, „1” – Szakember és „2” – Admin. A felhasználó jelszava SHA256-al lett titkosítva, ami ma az egyik legbiztonságosabb titkosítási algoritmus. Ezt szövegként 255 karakterig tároljuk. A regisztrációhoz és elfelejtett jelszóhoz az adatbázisba eltárolunk egy generált tokent. Ez 255 karakterből állhat. Tartozik ehhez egy lejárati időbélyeget tartalmazó mező is a „token\_expired\_at”. Ezeknek alapértelmezettként „NULL” az értékük, viszont, ha a hozzá tartozó funkciót használják, akkor egyedi értékeket kapnak. Az értékek csak megerősítésig, vagy adott ideig elérhetőek. Több időbélyeget is tárolunk, mint például a „last\_login\_at”, a „created\_at” és az „updated\_at”.

A „users” tábla „company\_id” egy egyed id-ja a „companies” táblából. Az „address\_id” pedig az „addresses” tábla egy sorára hivatkozik. Ennek a táblának az id-ja más táblákkal is kapcsolatban van: „jobs - customer\_id, - worker\_id”, „ads – user\_id”, „users\_jobs – user\_id”, „messages – sender\_id, - receiver\_id”, „chats – sender\_id, - receiver\_id”, „ratings – ratinged\_user\_id, - ratinger\_user\_id” és a „favorites – user\_id”.

• Normalizálás folyamatának leírása

• OOP jellegű megvalósítás: UML osztálydiagram

• Biztonsági kérdések, titkosítás, GDPR

• Ajánlott terjedelem: a feladat jellegétől függően 2-5 oldal.

## 5. Részletes feladatspecifikáció, algoritmusok

• A program lényeges függvényeinek, az osztályok metódusainak a specifikációja (mit valósít meg az adott függvény, illetve metódus, milyen paraméterei vannak, mi a visszatérési érték)

• Az algoritmizálható részek leírása valamilyen algoritmus-leíró eszközzel (struktogram, pszeudo-kód, esetleg UML aktivitás-diagram)

Ajánlott terjedelem: a feladat jellegétől függően 2-5 oldal.

## 6. Forráskód

• A teljes forráskódot a nyomtatott dokumentációba nem kell beletenni!

• Lehet viszont a nyomtatott dokumentációban a fontosabb kódrészeket magyarázattal szerepeltetni

## 7. Tesztelési dokumentáció

• Részletes tesztelési dokumentáció készítése

• A tesztelés során kiderült hibák felsorolása

A tesztelési dokumentációból derüljön ki, hogy ismeri a különböző tesztelési módszereket (pl. fekete doboz, fehér doboz módszer), tesztelési szinteket, stb.

Ajánlott terjedelem: a feladat jellegétől függően 5-7 oldal.

## 8. Továbbfejlesztési lehetőségek

A projekt tervezésénél és a megvalósítás közben is számos ötlet született, amik a jövőben a program fejlesztéséhez hozzájárulhatnak. Volt, amelyiket meg szerettük volna valósítani, de közös megegyezés alapján végül nem került bele ez a funkció.

Ilyen volt például a szakember profiljában a referencia munkáknak a képként való feltöltése. Ezeket a képeket akár a hirdetésekhez is odalehetne tenni, így a felhasználók a keresés közben egyből láthatnák, milyen munkát végez az adott szakember. Természetesen nem az összes képet, hanem csak egy pár darabot a szemléltetés miatt. Ha a felhasználó szeretné megnézni a többi referenciát is, itt a hirdetésnél lehetne egy gomb, hivatkozás, ami a szakember profiljához visz és ott az összes képet végig lehetne nézni. Terveztünk ehhez a funkcióhoz egy biztonsági részt is, amely annyiból állt volna, hogy a feltöltött képek átmennének egy vizsgálaton, amit egy admin jogosultságú felhasználó végezne. A vizsgálat lényege, hogy valóban témához, munkához kapcsolódó képek kerüljenek feltöltésre. Ilyen szűrés egyébként a projektben más résznél is megtalálható, például az értékeléseknél. Ott a szöveget ellenőrizzük, hogy kulturált és értelmes megjegyzések kerüljenek fel az oldalra és az értékelt szakember arculatába.

Egy másik feladat lehetne az, hogy a cégeket még mélyebben, komolyabban vegyük. Gondolunk itt arra, hogy több adatot kéne megadni, több ember is tartozhatna egy céghez. Akár úgy is megvalósítható lehetne ez a rész, hogy egy új jogosultság kerülne be a rendszerbe, egy úgynevezett „cég” felhasználó. Ez a felhasználó lehetne a cég tulajdonosa, igazgatója, vezetője vagy akár egy megbízott alkalmazott is. A legtöbb dolgot használhatná az oldalon. Például adhatna fel komplett hirdetést, amiben akár egy egész munkafolyamatot elvállalhatnának (építkezés: betonozás, szigetelés, berendezés…) és ehhez több alkalmazott és munkakör is tartozhatna. Valamint az alkalmazott szakembereket akár a vállalkozáson keresztül is lehetne regisztrálni, így ha új alkalmazottat vesz fel a cég, egyből regisztrálhatnák az oldalon is. Ez a rész akár a támogatáson és a használaton keresztül a jövedelem és a fenntartás költségeit is állhatnák.

A harmadik fejlesztési lehetőség lenne a legösszetettebb, legbonyolultabb és leghosszabb. Ez az ötlet az oldalon való fizetést tenné lehetővé. Tehát a munka befejeztével a teljes összeget, előleget vagy akár közben, több részletben való kifizetést is lehetővé tenné. Természetesen ehhez több biztonsági lépést is megkéne lépni, de ez sok átverést is kiküszöbölne. Például előleg kifizetése után eltűnik a szakember, vagy éppen a munka megvalósítását követően a megrendelő nem fizet a nem megfelelő munkavégzésre hivatkozva. A fizetéseket nyomon lehetne követni, így a „nem jött meg a pénz”, vagy a „rossz számlára utalta” problémákat is meg lehetne oldani. Emellett a munkaszerződésben lekéne fektetni a fizetéstől való elállásnak a feltételeit is. Készpénzes fizetésnél pedig annyival egészülne ki, hogy miután megtörtént a fizetés, egy tableten, vagy más eszközön aláírná mindkét fél és ezzel bizonyítanák, hogy megtörtént a megbeszélt összeg kifizetése.

# IV. Összegzés

Szerintem elmondhatom, hogy sok új dologgal találkoztunk, megismertük a fejlesztés különböző részeit és ezek által fejlődtünk. Mivel csapatban kellett dolgozni, így a kommunikációs készségeink is fejlődtek és a feladatok felosztásában is szereztünk némi gyakorlatot, ami a jövőben a hasznunkra válik. A témát szerencsére ismertük és nem is nehéz, ezért inkább a „hogy csináljuk” kérdés okozott fennakadást. Persze ezen is gyorsan tovább leptünk és így egy nagyon jó weboldalt sikerült készítenünk.

## Irodalomjegyzék, forrásmegjelölés

• Minden olyan forrás pontos megadása, amelyet a vizsgaremekben felhasznált.

• A forrás lehet pl.

• Könyv. Meg kell adnia a következőket: szerző(k), cím, kiadó, kiadás éve

• Weboldal. Meg kell adni a linket, az oldal címét

• Elektronikus dokumentum. Meg kell adni a szerzőt, a letöltés helyét, idejét

Ajánlott terjedelem: ½ -1 oldal.