

Miskolci Egyetem  
Gépészmérnöki És Informatikai Kar  
Általános Informatikai Intézeti Tanszék  
3515 Miskolc-Egyetemváros

**Komplex Tervezési Feladat**

*Feladat címe:*

**Munkaerőközvetítési adatok   
elemzése és hatékony kezelési   
módjainak vizsgálata**

*Készítette:*

**Hornyák Balázs Boldizsár**

*Bsc szintű, Mérnökinformatikus szakos  
Korszerű Web technológiák szakirányos hallgató*

*Konzulens:*

**Piller Imre**  
*egyetemi tanársegéd  
Matematikai Intézet*

**2020/2021 tanév, 2. félév**

Tartalomjegyzék

[1. Bevezetés 4](#_Toc87911359)

[2. Koncepció 5](#_Toc87911360)

[2.1. Megoldandó feladatok 5](#_Toc87911361)

[2.2. A probléma megoldására szolgáló módszerek 6](#_Toc87911362)

[2.3. Webalkalmazás használatának előnyei 6](#_Toc87911363)

[3. Követelmény analízis 8](#_Toc87911364)

[3.1. Befizetések kezelése 8](#_Toc87911365)

[3.2. Alap statisztikák megjelenítése 8](#_Toc87911366)

[3.3. Adatok megtekintése és szerkesztése 9](#_Toc87911367)

[3.4. Szakember ellenőrzése 9](#_Toc87911368)

[4. Specifikáció 10](#_Toc87911369)

[4.1.1. Megrendelések 10](#_Toc87911370)

[4.1.2. Alap adatok 11](#_Toc87911371)

[4.1.3. Adatok szerkesztése 12](#_Toc87911372)

[5. Technológiák kiválasztása 14](#_Toc87911373)

[5.1. Frontend 14](#_Toc87911374)

[5.1.1. Keretrendszerek áttekintése 14](#_Toc87911375)

[5.1.2. React, Angular és a Vue.js - alap gondolatok 15](#_Toc87911376)

[5.1.3. React, Angular és a Vue.js – funkcióinak összehasonlítása 16](#_Toc87911377)

[5.1.4. React, Angular és a Vue.js – szintaxisának összehasonlítása 17](#_Toc87911378)

[5.1.5. React, Angular és a Vue.js – komplexitásának összehasonlítása 18](#_Toc87911379)

[5.1.6. React, Angular és a Vue.js – teljesítményének összehasonlítása 18](#_Toc87911380)

[5.1.7. Frontend kiválasztása 20](#_Toc87911381)

[5.2. Backend 21](#_Toc87911382)

[5.2.1. React és a Node.js (Express) 21](#_Toc87911383)

[5.2.2. Laravel és a Ruby on Rails összehasonlítása 22](#_Toc87911384)

[5.2.3. A backend kiválasztása 23](#_Toc87911385)

[5.3. Adatbázis 24](#_Toc87911386)

[5.3.1. PostgreSQL 24](#_Toc87911387)

[5.4. A technológiák kiválasztásának összegzése 24](#_Toc87911388)

[6. A feladat megvalósítása 25](#_Toc87911389)

[6.1. Adatbázis bemutatása 25](#_Toc87911390)

[6.2. Tervezés 26](#_Toc87911391)

[6.3. Implementáció 26](#_Toc87911392)

[7. Összefoglalás 27](#_Toc87911393)

# Bevezetés

Egy munkaerőközvetítői weboldal üzemeltetése nagyon sok adminisztratív munkával jár. Ezeket az adatokat megfelelően kell tudni kezelni. Ennek a problémának a megoldására egy különálló webes applikáció nyújthat megoldást, melyet akár az informatikában nem annyira jártas ember is könnyen tud kezelni.

Egy már létező honlap a mestertkeresek.hu, amely szakemberek hirdetéseivel foglalkozik. Ennek a portálnak az ügyintézői felületének létrehozása a célom. Szakdolgozatom írásával törekszem ennek az alkalmazásnak a tervezését, létrehozását és működését teljeskörűen bemutatni. A rendszernek alkalmasnak kell lennie az adatok megtekintésére és módosítására, illetve általános információkat kell tudnia adni a vállalkozásról, melyet bárki tud kezelni és lerövidíti az adminisztratív munkák idejét. A kezelendő adatok között egyaránt szerepelnek a különféle szakterülethez tartozó mesterek, a hozzájuk tartozó rendelések és az ezekkel kapcsolatos számlázási információk. Ezen adatok ellenőrzését és javítását kell tudnia ellátnia, illetve kisebb szintű pénzügyi statisztikát megjelenítenie. Mindez naprakészen megtekinthetőnek kell lennie web-böngészőben.

//todo: bővítés?(mivel?)

# Koncepció

Ahhoz, hogy megoldást találjunk, először mélyebben bele kell ásni magunkat a problémába. A már működő, a bevezetésben említett weboldal adatbázisát felhasználva kell az információkat megjeleníteni és szerkeszthetővé tenni az adminisztrációs munkát végző felhasználó számára. De miért is van erre szükség? Nagyon sok cég mai napig Excelben vagy egyszerű CSV fileokban tárolja az adatait. Véleményem szerint ez azért rossz, mert azok sokszor átláthatatlanok és sérülékenyek. E szakemberkereső oldal viszont komplex adatbázisban tárolja adatait és ezt felhasználva kell a rendszert felépíteni úgy, hogy azok integritása megmaradjon a relációs adatbázisban a szerkesztés után is. Az adatbázis műveletek használata programozói munkát és sok időt igényel ehhez, az egyébként egyszerűnek tűnő feladathoz. Erre fókuszálva kell alkotni egy admin felületet, amelyet használva egy, az informatikában nem jártas ember is tud kezelni. Így tehát egy könnyedén áttekinthető felületre van szükség, amelynek használatát pár perc alatt el tudja sajátítani a felhasználó. Ennek következtében a pénzügyi, marketing és ügyintézői munkavégzés kiadható nem fejlesztők kezébe is.

## Megoldandó feladatok

A szoftver feladata, hogy a mestertkeresek.hu adatbázisából kinyerve megkapjuk az ügyintézéshez szükséges információkat és mindezeket módosítani is tudjuk. Az egyik ilyen megoldandó probléma a pénzbefizetések adminisztrálása. A munkaerőközvetítői weboldalon egy hirdetés feladásáért a felhasználónak fizetnie kell. Az ilyen tranzakciók ellenőrzést igényelnek, különösképp egy banki átutalás, hiszen ez akár több napot is igénybe vehet, hogy beérkezzen az összeg a felhasználótól a vállalkozás számlájára jusson. Ezeknél nagy segítség egy olyan alkalmazás, amely nyilvántartja ezeket és segítséget nyújt az adminisztrálásban azzal, hogy át tud írni adatokat, egyszerre több táblában akár egyetlen gombnyomásra.

Adatbázis táblákból kiolvasni elég nehéz megtudni, hogy éppen hogyan is áll a vállalkozás. A program feladata többek között, hogy kisebb statisztikát készítsen a bevételről és vásárlásokról. Mindezeket, akár diagramban, akár egyszerű kimutatásként ciklusokra felosztottan meg kell tudnia jeleníteni. Ezekkel az információkkal naprakészen lehet látni a cég állását, amellyel egyszerűbben lehet számolni akár azt, hogy van e szükség nagyobb marketingre, vagy másféle változtatásokra, hogy a kívánt bevételt termelje az oldal.

Regisztráció során sok különféle adat kerül eltárolásra az adatbázisban egy felhasználóról, az ezekhez való hozzáférést könnyedén el kell látni a rendszernek. Gyakorta van szükség megtekinteni vagy adott esetben változtatni is egy kliens adatain. Illetve előfordulhat a hirdetéseket feladó emberekkel, hogy hibásan írják le a megjelenítésre szánt szöveget anélkül, hogy ezt észrevennék. Az így ejtett magyar nyelvtani hibák átírását, javítását is egyszerűen el kell tudni végezni, ezzel is növelve a szakemberkereső oldal színvonalát.

## A probléma megoldására szolgáló módszerek

Egy adatbázis kezelésére léteznek adatbáziskezelő szoftverek, melyben át lehet tekinteni az egyes táblák adatait, vagy akár szerkeszteni is lehet ezeken. Ilyek például a RazorSQL, TablePlus, pgAdmin és még sok más. Azonban ezekben elég nehéz lehet egy hatalmas komplex adatbázis kezelni, még akkor is, ha saját SQL lekérdezéseket írunk. Esetenként nem csak a problémákra való megoldás tűnhet időigényesebbnek és bonyolultabbnak, de maga az adminisztrátori munkavégzés is. A fent említett problémákra egy saját webalkalmazás fejlesztése a legmegfelelőbb megoldás, amellyel egy webes böngészőben szinte bárhonnan el lehet végezni a szükséges munkát. Így nincs szükség semmiféle adatbázis kezelő szoftver ismeretére, illetve az ügyintézési folyamat is gördülékenyebbé válik.

## Webalkalmazás használatának előnyei

Ha egy adatbázis kezelő szoftvert úgymond bekonfiguráltunk az adott munkához szükséges SQL lekérdezések megírásával, ha ezeken változtatni szeretnénk vagy megsérülnek ezek a lekérdezések, akkor azt hatalmas munkával jár helyreállítani. Nem is beszélve arról, hogy ezek a szoftverek egy adott számítógépre vannak telepítve, amellyel hatalmas veszélynek tesszük ki az adatbázist és ezzel együtt a vállalkozást is a vírus és egyéb támadás szempontjából. Vagy például ha az alkalmazott számítógépe meghibásodik, és nem tud dolgozni az adatbázissal, akkor ez idő alatt nem fogja tudni ellátni az adminisztrátori feladatát.

Mind ez elkerülhető egy olyan webalkalmazás használatával, amely egy felhőben futó szerveren tárolja ezeket a funkciókat és biztonságosan használható bármely internetre csatlakozó számítógép. Továbbá, alkalmas akár egy régebbi vagy alacsony specifikációjú eszköz is a munka elvégzésére, hiszen nincsen akkora memóriaigénye, mint egy adatbázis kezelő szoftvernek.

Könnyebben testre szabható azáltal, hogy csak azokat az információk jelennek meg, amelyek relevánsak az épp elvégzendő feladathoz. A vizuális megjelenítés is egyszerűbbé teszi azáltal, hogy szinte bármilyen diagramot vagy egyéb kimutatást létre lehet hozni. Mindemellett előnye még, hogy könnyebb a karbantartása és a szükség szerinti fejlesztése a webalkalmazásnak.

# Követelmény analízis

## Befizetések kezelése

A mesterkeresek.hu-n havi előfizetéses rendszer működik, amelyek három, hat és tizenkét havi megjelenést biztosít egy mesterember számára, aki ezáltal hirdetheti magát az oldalon. Ezeket az előfizetéseket banki átutaláson keresztül vagy online bankkártyás fizetéssel tehetik meg a felhasználók.

A bankkártyás fizetés Barion rendszeren keresztül történik, amely egy elektronikus fizetéseket lebonyolító szolgáltatás. Ha ezt a fizetési módot választotta a felhasználó és sikeres volt a tranzakció, akkor azzal automatikusan megtörténik az előfizetés aktiválása. Azonban ha banki átutalást választja, akkor azt egy adminisztrátornak ellenőriznie kell. Amíg nem érkezik be az előfizetés helyes összege, addig az oldalon nem jelenik meg a felhasználó hirdetése. Többek között ezt az ellenőrző és aktiváló folyamatot hivatott elősegíteni e webes alkalmazásnak a létrehozása.

## Alap statisztikák megjelenítése

Az alkalmazásban szerepelnie kell a bevétel kisebb statisztikájának megjelenítése. Az eladások darabszámának és a bevétel forint alapú kiírásának, illetve ezeknek ciklusokra bontott reprezentálását meg kell tudni oldani. Mindezeknek az előző periódushoz képesit százalékos megjelenítését is képes legyen kiszámolni. Például, ha előző héten 4 darab eladás volt ezen a héten pedig 6, akkor az +50%-os többleteladást ismertet. Továbbá az összes regisztrált mester és a függőben lévő rendelések aktuális darabszámának megjelenítése is fontos, hogy naprakész információkat kapjunk a vállalkozásról. Így sokkal egyszerűbb meghatározni a cég aktuális pénzügyi egyenlegét. Ez akkor lehet hasznos, ha látjuk az előző ciklushoz hasonlítottan, hogy több vagy kevesebb bevételt termelt az oldal. Például kevesebb keresetnél nagyobb marketingre van szükség.

Az alkalmazásban szerepelnie kell az eladások szemléltetésére egy havi lebontású diagram megjelenítésére. Ez az alapvető statisztikai illusztráció a csomagok eladásának darabszámú megoszlását kell tudnia reprezentálni.

## Adatok megtekintése és szerkesztése

Az adatbázis egyes tábláit - melyet részletesen egy későbbi fejezetben mutatok be - megjeleníthetővé és szerkeszthetővé kell tenni. Ilyenek például a szakemberek, felhasználók és munkák tábla, melyek sorait elsősorban pár lényegi információval szükséges megmutatni, melyből beazonosítható az adott rekord.

Ezek szerkesztésére és további információi megjelenítésére egy adott gombra kattintva egy új felületnek kell megnyílnia. Ennek a felületnek űrlapszerű módon kell megjelennie, ahol a különböző adattípusoknak megfelelő kitöltési módnak szükséges szerepelnie. Például dátum esetében dátumválasztó (date picker), logikai változónál pedig kapcsoló (switch button).

Továbbá indokolt a CSV exportálás is a kijelölt rekordokra, amely az adatok egyszerű továbbításában és csoportosításában tud segíteni.

## Szakember ellenőrzése

A mesterkeresek.hu egyik nagy előnye, hogy a szakember ellenőrzésre kerül, amikor a hirdetése megjelenik az oldalon. Ennek kivitelezése a regisztrációnál megadott adószám felhasználásával történik. A hiteles nyilvántartó rendszerekből ellenőrizhető, hogy a hirdetést feladó szakember adatai a valóságnak megfelelnek-e, valamint a tényleges tevékenységi köre megegyezik az általa megadott információkkal. Ezen eljárás a fogyasztók védelmére irányul, növelve ezáltal a szolgáltatás minőségét.

A program célja többek között a vizsgálatot végző személy munkájának könnyebbé tétele, azáltal, hogy - a valós eredmény esetében - egy kattintásra megváltozik a felhasználói webes felületen a szakember hirdetésének státusza („Nem ellenőrzött mester”-ről „Ellenőrzött mester”-re).

# Specifikáció

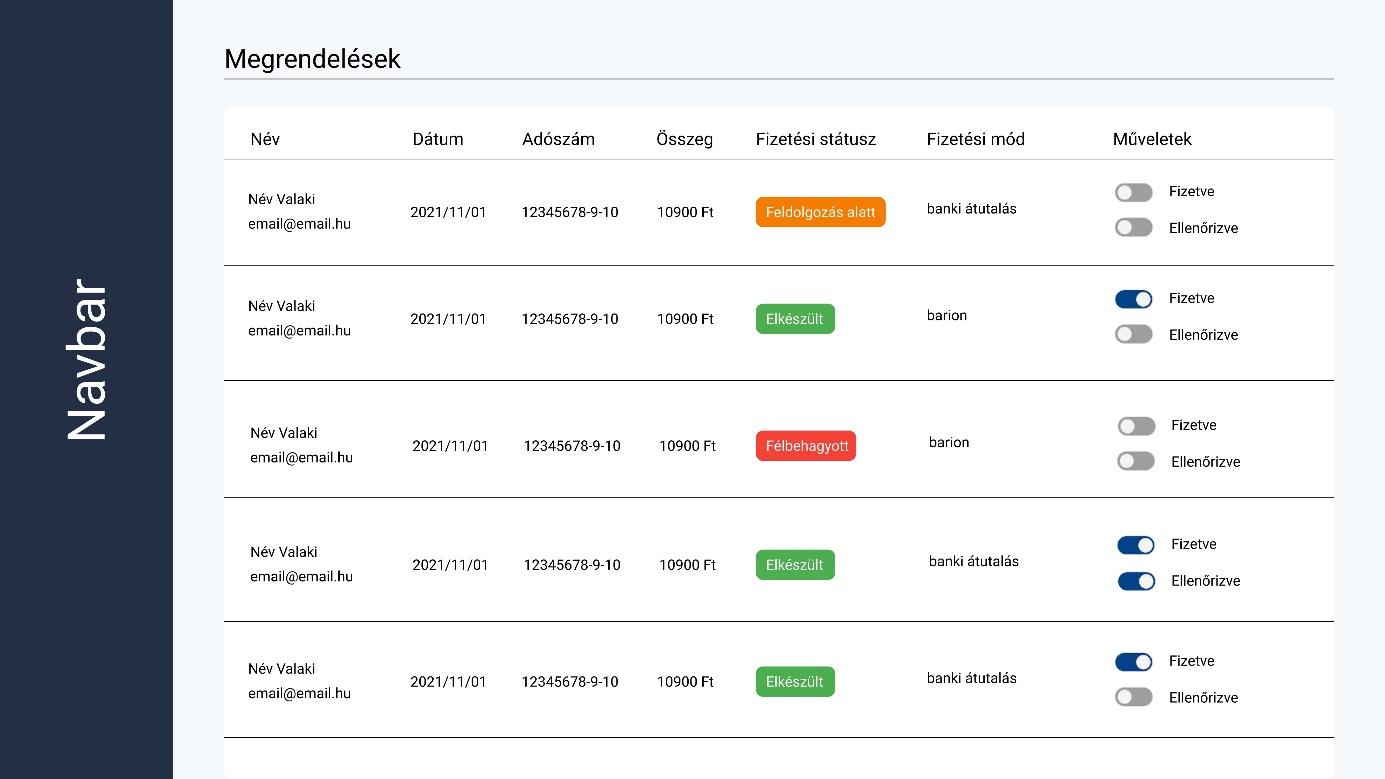
A következőkben sorra veszem, hogy mik azok, amikre képesnek kell lennie az alkalmazásnak, illetve mire hogyan kell reagálnia az egyes esetekben. Azonban ezek megvalósítását egy későbbi fejezetben fogom részletesen bemutatni.

### Megrendelések

Milyen használati esetek fordulhatnak elő, illetve hogyan kell működnie mindennek? Táblázatszerűen meg kell jeleníteni a felhasználó egyes adatait, melyek; név, email cím, adószám. Továbbá a megrendelés adatait is ezen a felületen kell látni egyben a vásárló adataival, amelyek nem mások, mint a; rendelésszám, keltezés, a befizetés összege, státusza és módja (online bankkártyás vagy banki átutalásos). Ezek a felsorolt adatok a táblázat egy sorát szükséges képeznie, melyek mellett helyet kell kapnia a fizetési állapotot és ellenőrzöttségi státuszt átállító gombnak. Az utóbbi egy manuális folyamat melyet az alkalmazást használó ügyintézőnek kell elvégezni erre alkalmas nyilvántartó rendszerek segítségével.

A fizetési státusz állításának lehetőségét csak banki átutalásnál kell lehetségessé tenni, ugyanis a bankkártyás tranzakciónál egyértelmű információk vannak a fizetés sikerességéről. A fizetés állapot jelzői a következők lehetnek: elkészült, félbehagyott, feldolgozás alatt. Az elkészült státusz vagy egy sikeres Barionos (bankkártyás) fizetés után lehetséges vagy egy olyan banki átutalás, amelynek összege sikeresen beérkezett és azt az ügyintéző már jóváhagyta. A feldolgozás alatti állapotjelző banki átutalásnál lehetséges, ezek olyan megrendelések, melyek pénzösszege nem érkezett még be vagy az adminisztrátor még nem hagyta jóvá. Ezt az ellenőrző folyamatot bezáróan a fizetési státusz módosítására szolgáló gomb megnyomásával kell tudnia a programnak lezárni, ezzel elkészültbe léptetni a megrendelést. Félbehagyott státusz Barionos fizetésnél lehetséges, amikor a felhasználó vagy sikertelen tranzakcióval zárta le a folyamatot, vagy pedig elnavigált az oldalról. Ennek megjelenítése marketing szempontból fontos, ugyanis ha az ügyintéző látja, hogy adott megrendelésre sikertelen volt a fizetés, akkor felkeresheti emailben a felhasználót segítségnyújtás szempontjából.

A fent leírtakra a 2. ábra ad segítséget a vizuális elképzelésben, amelyen az alkalmazás egy képernyőterve (wireframe) látható.

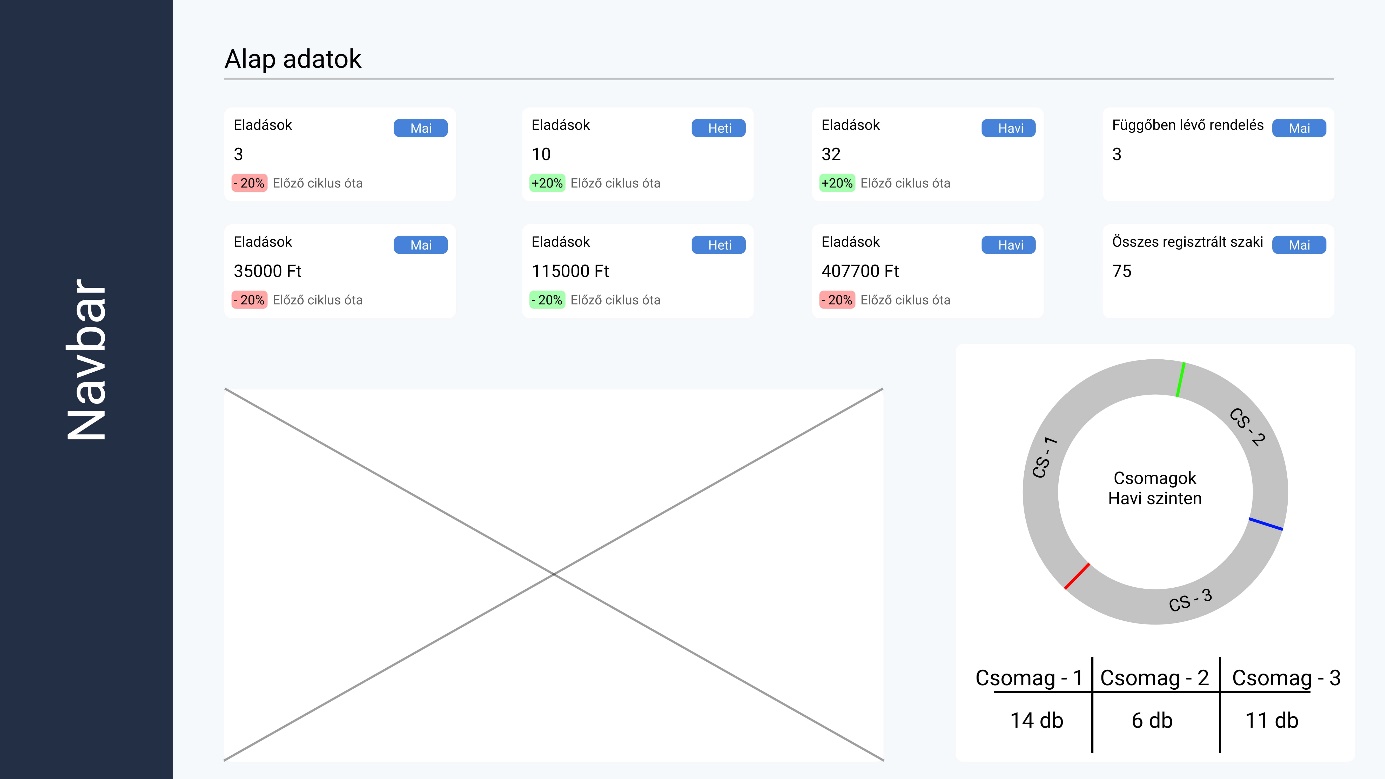


1. ábra Megrendelések képernyőterv

Ennek az egész táblázatos kialakításnak oldalanként lapozhatónak kell lennie és ki kell tudni jelölni azt, hogy hány elemet szeretnénk megjeleníteni az adott oldalon. Mindez különösen nagy várakozási idő nélkül kell tudnia teljesíteni az alkalmazásnak a gördülékeny munkavégzés érdekében.

### Alap adatok

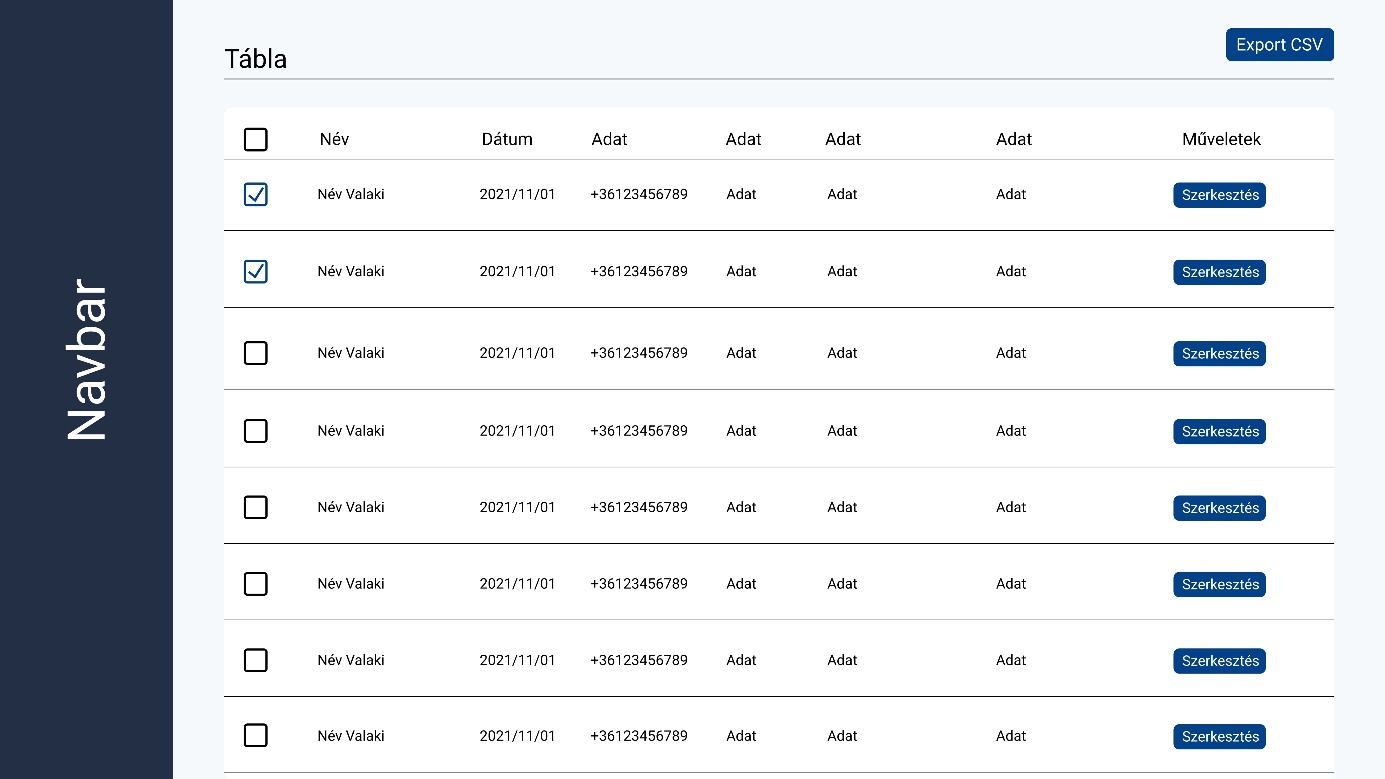
A 3.2. fejezetben leírtak alapján ezen az oldalon a vállalkozás naprakész statisztikáját kell tudni átláthatóan megjeleníteni. A bevétel illetve az eladások számát napi, heti és havi lebontásban szükséges kimutatni. Ezeket az előző ciklushoz mérten a rendszernek ki kell tudni a számolni a többletet vagy épp a csökkenést. Ezeket mindig az aktuális naptól számítva kell nézni, hiszen például ha a havi lebontást mindig a hónap első napjától számolnánk, akkor reális értéket csak a hónap utolsó napján kapnánk. Mindez igaz a csomag-eladások megoszlásának kimutatására szolgáló diagram esetében is. Ezek szemléltetésére a 3. ábrán található képernyőtervet készítettem el.



2. ábra Alap adatok képernyőterv

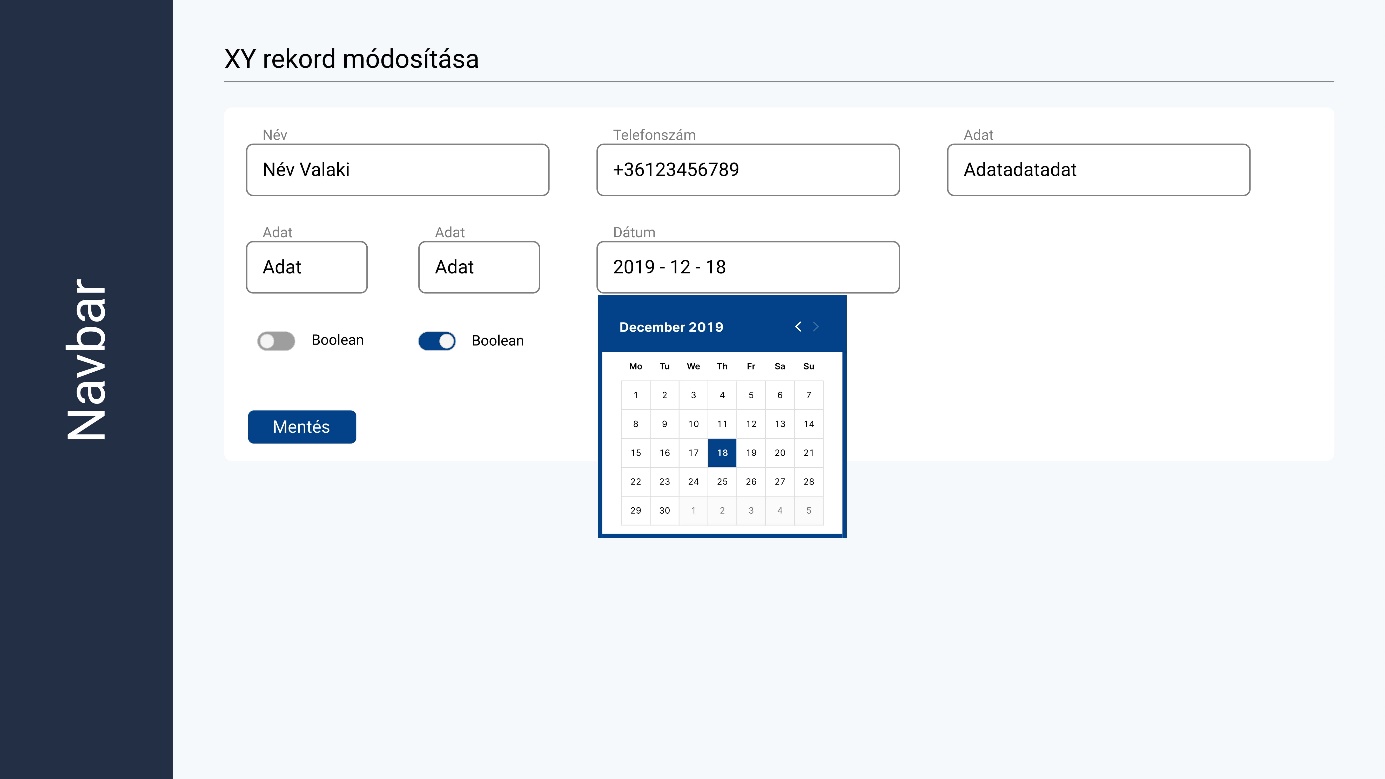
### Adatok szerkesztése

Ebben a funkcióban az adatbázis legfontosabb tábláit kell megjeleníteni és módosíthatóvá tenni. Ezeket egy rendezhető táblázatban és csakúgy, mint a megrendelések funkciónál, itt is testreszabhatónak kell lennie az adott oldalon megjelenő sorok számának. Egy adott sorban csak pár lényegi információnak kell szerepelnie az átláthatóság érdekében. A 3.3. fejezetben leírtak alapján az alkalmazásban szerepelnie kell CSV exportálás funkciónak is. Ehhez szükség van minden sorban egy jelölőnégyzet elhelyezésére, melynek segítségével összegyűjthetjük azokat a rekordokat, melyeket az exportálás gombra azonnal letölthetünk. Ezek megjenési tervét a 4. ábrán található wireframe mutatja be.



3. ábra Egy tábla megjelenítésének képernyőterve

Az egyes rekordok szerkesztése az adott soron elhelyezett gomb segítségével kell tudni eljutni. Itt minden szerkeszthető információt fel kell tüntetni és az adott adattípushoz megfelelő megjelenítési és változtatási módot kell használni. Ezek illusztrálására az 5. ábra szolgál.



4. ábra Rekord szerkesztése képernyőterv

# Technológiák kiválasztása

A webalkalmazás elkészítése három fő részre bontható: frontend, backend és az adatbázis. Ezek létrehozására több technológia létezik. A következőkben összehasonlítom ezeket és megpróbálom a legmegfelelőbbet kiválasztani.

## Frontend

Először azt érdemes megvizsgálni, hogy van e szükség valamilyen keretrendszer használatára [1] az adott feladat megoldására. Ha egy kisebb alkalmazáson dolgozunk elég lehet natív JavaScripttet vagy jQueryt használni. Ezt csak akkor érdemes választani, ha biztosak vagyunk benne, hogy nem lesz komplexebb a projekt. Egy keretrendszer több száz MB-nyival is megnövelheti az applikáció méretét, ami negatívan hathat ki a betöltési időre. Azonban ha megoldandó feladat összetett és van rá esély, hogy további funkciókkal fog bővülni, akkor érdemes elgondolkodni egy keretrendszer vagy könyvtár használatán.

### Keretrendszerek áttekintése

A keretrendszerek sok mindenben eltérnek a hagyományos JavaScripttől viszont merőben megkönnyítheti a munkát. Vannak dolgok amiket könnyebb kivitelezni velük vagy csak egyszerűen rövidebb idő alatt megolthatók bizonyos problémák. Frontend keretrendszerek előnyei:

* Komponensekből állnak, amiket újból fel lehet használni
* Áttekinthetőbb és strukturáltabb kódot adnak
* Időt spórolnak meg számunkra
* Lehetőséget ad single-page applikáció írására

React, Angular és a Vue.js: Frontend framework esetén ők a piacvezetők, így ezeket hasonlítom össze.

### React, Angular és a Vue.js - alap gondolatok

Először egy fontos megjegyezni, hogy nincs legjobb keretrendszer vagy könyvtár [2]. Mindhárom könyvtár jó okokból nagyon népszerű. Mindegyiküknek megvannak az erősségei és gyengeségei, és általában bármelyik könyvtárat fel lehet használni bármely projekthez.

* Az Angular a Google által kifejlesztett keretrendszer ami 2010-ben jelent meg. A Google belsőleg is használja az Angulart, ezért elképzelhetetlennek tartom, hogy az Angular eltűnik egyik napról a másikra, hiszen fenntartják és folyamatosan fejlesztik.
* A React.js-t eredetileg a Facebook fejlesztette, ebből adódóan a Facebook is azt használja, valamint az Instagram. 2013-ban jelent meg, tehát később, mint az Angular.
* A Vue egy önálló projekt, amelyet egyetlen vállalat sem épít. Korábban egy személy fejlesztette (Evan You), azonban mára már elkötelezett munkatársai vannak, akik a Vue-n dolgoznak.

A három közül az Angular határozottan a „legnagyobb” keretrendszer. Az Angular sok mindenre kiterjed. Segít a fejlesztőnek a felhasználói felület(user interface) vezérlésében, a felhasználói bevitelre való reagálásban, a felhasználói input formában történő validálásában, útválasztásban(routing), az Ajax Http kérések küldésében, offline támogatás és PWA képességek biztosításában, tesztelésben, alkalmazás felépítésében, több alkalmazás kezelésében és még sok-sok minden másban. Ezen kívül létezik egy hivatalos CLI, amely segítséget nyújt az Angular projektek létrehozásában és kezelésében, naprakészen tartásával, függőségek hozzáadásával és még a telepítéssel is.

A React teljesen más! Amíg az Angular mindent megad, a React csak egy dolgot ad: egy könyvtárat, amely tartalmat jelenít meg a DOM (Document Object Model) számára, és azt hatékonyan ellenőrzi. A React megadja az összes szükséges eszközt annak meghatározásához, hogy milyen körülmények között, milyen módon kell megjeleníteni az információt a képernyőn. De nem tartalmaz beépített űrlap-ellenőrzési támogatást, útválasztót (router) és nem szállítja a saját Http-kliensét. Ezen kívül nem jár más különleges funkció hozzá, és ebben a tekintetben határozottan „kisebb”, mint az Angular. Ezeknél a funkcióknál a React közösségre kell támaszkodni.

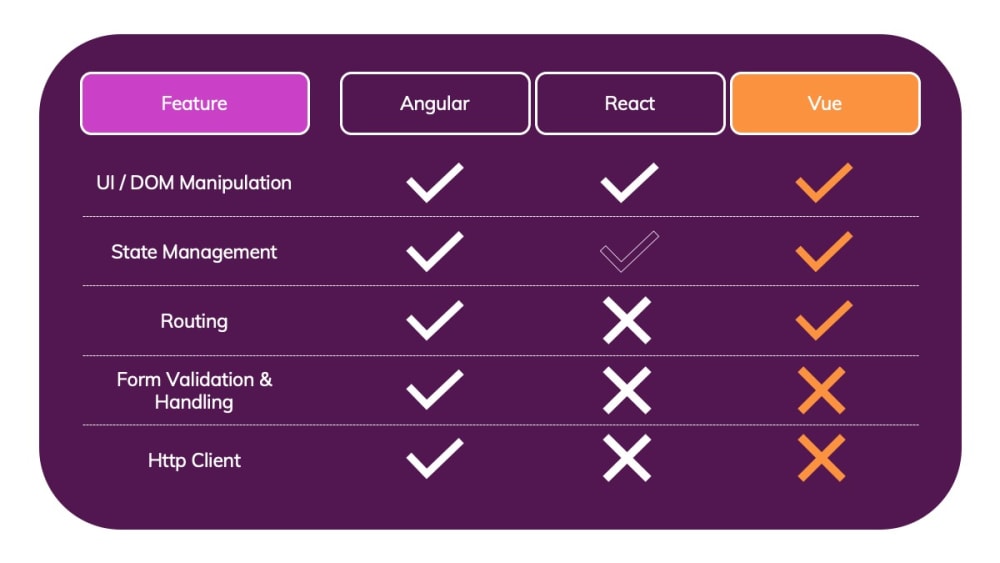
A Vue egy olyan keret, amely a React és az Angular között helyezkedik el. Nem olyan „nagy”, mint az Angular, de határozottan több funkciót tartalmaz, mint a React. A Vue beépített állapotkezelést biztosít, és beépített útválasztóval (router) is rendelkezik. Nem tartalmazza azonban az űrlap-hitelesítést vagy a Http kliens funkcióit. Csakúgy, mint az Angular és a React, a Vue lényege is a felhasználói felületek felépítése az újrafelhasználható komponensek kombinálásával.

### React, Angular és a Vue.js – funkcióinak összehasonlítása

Merül fel a kérdés, hogy sokkal jobb-e az Angular, mint a többi keretrendszer? Több funkciót kínál - és ez fejlesztőként több lehetőséget ad. A válasz: Nem, ezek nem jelentik azt automatikusan, hogy jobb. Nem is rosszabb - egyszerűen a projekten múlik, és a személyes preferencián.

A React egyszerűsége, valamint a komponensekre és a kezelőfelület-megjelenítésre való nagy hangsúlya nagy erőt jelent. Ahol az Angularnak sok mindent össze kell kapcsolnia és gondoskodnia kell arról, hogy zökkenőmentesen működjenek együtt, a React-nek nem kell ezt tennie. A React egy dolgot csinál, és ezt nagyon-nagyon jól csinálja! A többi „esszközt”, amelyekre végül szükség lehet, azokat a nagyon aktív React közösség biztosítja. Ez a közösség olyan megoldásokat fejlesztett ki (azaz extra csomagokat, amelyeket hozzá lehet adni a projektjéhez), mint például a React router, a Redux vagy a Formik. Tehát lehet azt állítani, hogy a React csapata arra összpontosít, hogy a lehető legjobb felhasználói felület-megjelenítő könyvtárat adja, míg a közösség egyedi projektekre összpontosít, amelyek kiegészítik ezt a könyvtárat.

Az Angular csapata természetesen hatalmas tapasztalattal rendelkező fejlesztőkből álló csapat, így valószínűleg nem kell tartani attól, hogy elrontják a dolgokat, csak azért, mert egynél több könyvtáron dolgoznak. Ezen túlmenően, az egész kezelése egy fedél alatt történik, így természetesen ez azt is jelenti, hogy soha nem ütközik verzióváltási problémákba vagy inkompatibilitásokba. A különböző építőelemek mindig zökkenőmentesen működnek együtt, mert egy csapat irányítja őket. Ez mind Angular, mind Vue esetében igaz.



5. ábra Keretrendszerek és funkcióik [2]

### React, Angular és a Vue.js – szintaxisának összehasonlítása

Most, hogy közelebbről bemutattam a különböző funkcióit a keretrendszereknek, áttérek a kódolás formájára. Mert ez nagyon sokat számít! Az Angular projektek a TypeScript-et használják, amely a JavaScript szuperkészlete (azaz kiegészítése). A TypeScript nem fut a böngészőben, de az Angular projektek olyan eszközöket tartalmaznak, amelyek lefordítják a TypeScript kódot a böngészővel kompatibilis JavaScript kódra a háttérben. Angularban, mint fejlesztőnek, soha nem kell olyan kódot írni, amely közvetlenül hoz létre vagy távolít el elemeket a DOM-ból. Ehelyett komponenseket hoz létre, hogy a keretrendszer elvégezze el a nehéz munkát. Meghatározhatja a komponensek bemeneteit (tulajdonságait) és kimeneteit (eseményeit), valamint kezelhet néhány komponens-specifikus vagy alkalmazás-szintű állapotot.

A React általában JavaScript-t használ, és egy speciális JavaScript „funkciót” kezel, amelyet „JSX” -nek hívnak. A JSX valójában nem része a JavaScript nyelvnek, a React projektek csak úgy vannak beállítva, hogy ezt a „HTML JavaScriptben” szintaxist támogassák a fejlesztés során. Csakúgy, mint a TypeScript az Angular projektekben, a JSX is böngészőbarát JavaScript kóddá áll össze a végén.

A Vue normál JavaScriptet használ (bár opcionálisan használhatja a TypeScriptet is), és „Single File Components”-nek hívják, amit kezel. Ez egy olyan keret, amely ugyanúgy komponensekből tevődik össze (csakúgy, mint a másik kettő), de az Angularhoz hasonlóan szétválasztja a HTML és a JavaScript logikát. A Vue támogatja a JSX-t is, de a projektek túlnyomó többségében a fent leírtak szerint alkalmazza a kódot.

Melyik szintaxis a legjobb? Vannak, akik jobban kedvelik az Angular módszerét a kód strukturálására és a TypeScript használatát. Más fejlesztők szeretik a JSX-t és a „csak JavaScript” -et, mint React esetében. Van aki szereti a kódfelosztást, de szeretné használni a JavaScript-et, és így a Vue mellett dönt. 100% -ban szubjektív, és itt nincs objektív érv. Egyetlen szintaxis sem gyorsabb vagy lassabb, mint a többi. Ez teljesen preferencia kérdése.

### React, Angular és a Vue.js – komplexitásának összehasonlítása

Mennyire könnyű megtanulni, és használni ezeket a keretrendszereket? Ez sokat számít! A három közül a Vue valószínűleg a legkönnyebben megtanulható keretrendszer. Ennek két fő oka van:

* Nincs szükség speciális beállításokra. Csak importálni kell a Vue könyvtárat egy HTML fájlba, és hozzáadni néhány JS-t ehhez a fájlhoz. Nincs szükség egyedi projektbeállításra vagy utólagos fordításra
* A Vue csak JavaScript-et és HTML-t használ - nincs különösebb szintaxis, amelyet először meg kell tanulni. A HTML-t olyan direktívákkal javítja, mint a v-for, de ezek a fejlesztések többnyire eléggé magától értetődőek.

Az Angular és a React is mindenképp egy kicsit nehezebb elkezdeni, mint a Vue-t. Mindkettő bonyolultabb projektbeállítást igényel fejlett fejlesztői eszközöket (például Webpack) az elinduláshoz. A projektek létrehozása illesztési beállításokkal mégis egyszerű - megvan az Angular CLI Angular projektekhez, és a create-react-app a React projektekhez. Az Angular esetében ezután meg kell tanulnia a TypeScript-t is, a React-hez pedig meg kell ismerni JSX-szintaxisát, és meg kell értenie, hogyan kell írni a React-komponenseket.

### React, Angular és a Vue.js – teljesítményének összehasonlítása

A teljesítmény számít! De könnyű túl nagy hangsúlyt fektetni rá, vagy rossz viszonyítási pontokat alapul venni, és így rossz következtetéseket levonni. Különböző típusú teljesítmények is léteznek:

* Indítási teljesítmény: Milyen gyorsan töltődik be és vált interaktívvá a webalkalmazás. Az indítási teljesítményt elsősorban a létrehozott kódcsomag mérete befolyásolja. Vagyis a saját kód + a keretrendszer kódja együttesen.
* Futásidejű teljesítmény: Mennyire gyors az alkalmazása, miután elindult és működik. A futásidejű teljesítményt elsősorban a választott keret belső elemei befolyásolják, valamint az, hogy miként közelíti meg a DOM manipulációját és frissítését.

Indítási teljesítmény: A legfontosabb a teljes csomagméret, azaz a létrehozott és a telepítési kiszolgálóra feltöltött kód. Gyakran egy alapvető „Hello World” alkalmazást vesznek igénybe, hogy összehasonlítsák a három fő keretrendszer kötegméretét. Angular lassabb, mint a Vue és a React alkalmazások - a Vue pedig valamivel gyorsabb, mint a React.

Lighthouse [3] eszközzel mért indítási teljesítmény ezredmásodpercben. A mérésekhez a következő specifikációval ellátott számítógépen voltak futtatva: i7-8750H, 64 GB RAM, Linux 5.14.14, Chrome 95.0.4638.54 (64-bit):

|  |  |  |  |
| --- | --- | --- | --- |
| Keretrendszer | angular-v13.0.0 | react-v17.0.1 | vue-v3.2.21 |
| Betöltési idő | 2783,2ms | 2580,3ms | 2105,0ms |

Nagyobb alkalmazások esetén mindhárom keretnek nagyjából egyforma méretű kódcsomagokat kell készítenie - feltételezve, hogy olyan optimalizálási technikákat használ, mint a „Lazy Loading”, amelyet mindhárom keret támogat.

Futásidejű teljesítmény: Végső soron a felhasználói élmény számít. Ha egy oldal gyorsnak érződik a végfelhasználó számára, akkor nem számít, ha technikailag kicsit lassabb a motorháztető alatt. A React nem csak arra koncentrál, hogy minden lehetséges JavaScript trükköt felhasználjon az összes szükséges felhasználói felület frissítésének hatékony észlelésére és végrehajtására, hanem olyan technikákat is felhasznál, amelyek során bizonyos feladatokat prioritássá tesz, hogy a betöltött oldal gyorsabbnak érződjön a végfelhasználó számára. Előfordulhat például, hogy elsőbbséggel kezeli a felhasználói bevitelt, és késlelteti a képernyőn megjelenő szövegek frissítését.

|  |  |  |  |
| --- | --- | --- | --- |
| Keretrendszer | angular-v13.0.0 | react-v17.0.1 | vue-v3.2.21 |
| 1000 sor létrehozása | 122,2ms | 127,1ms | 106,5ms |
| Mind az 1000 sor frissítése | 105,1ms | 105,5ms | 92,7ms |
| Minden 10. sor frissítése 1000 sor erejéig (16x CPU lassítás) | 181,8ms | 231,0ms | 201,1ms |
| Egy kiválasztott sor kiemelése (16x CPU lassítás) | 72,7ms | 110,9ms | 31,0ms |
| 2 sor cseréje 1000 soros táblázatban (4x CPU lassítás) | 331,2ms | 332,8ms | 51,5ms |
| Egy sor eltávolítása | 20,9ms | 20,9ms | 22,4ms |
| 10 000 sor létrehozása | 1105,4ms | 1105,4ms | 999,3ms |
| 1000 hozzáfűzése egy 1000 soros táblázathoz (2x CPU lassítás) | 244,4ms | 244,4ms | 206,6ms |
| 1000 soros táblázat törlése (8x CPU lassítás) | 155,4ms | 155,4ms | 72,0ms |
| Átlag | 259,9ms | 270,3ms | 198,1ms |

Összességében mindhárom keretrendszer nagyszerű futásidejű teljesítményt nyújt, de a Vue kimagasló teljesítményt ad két sor cseréje és egy kiválasztott sor kiemelésénél.

### Frontend kiválasztása

Az általam tervezett webalkalmazás könnyebben és gyorsabban elkészíthető valamelyik keretrendszer használatával, hiszen elég összetett problémát szeretnék meg vele megoldani és az előreláthatólag tovább bővíthető lesz különféle funkciókkal. Így az nem is kérdés, választok egy keretrendszert ennek elkészítésére. A fent leírtakból jól látszik, hogy sok esetben csak a személyes preferencia dönt a kiválasztásban. Én személy szerint a React-et kedveltem meg leginkább, igy ez a választottam a projekt létrehozására frontend terén.

## Backend

Az alkalmazásához megfelelő backend technológia kiválasztása [4] az egész webalkalmazás egyik legfontosabb lépése. Mivel már eldöntöttem, hogy React-ben szeretném írni a frontendet ezért ehhez „illő” backend kiválasztása a cél. A technológiai ipar annyit fejlődött, hogy számtalan eszköz és keretrendszer közül lehet választani, amelyek zavart okozhatnak. Ennek kiválasztása olyan döntés, amelynek előre nem látható következményei lehetnek a jövőben. Ezért elengedhetetlen, hogy alapos kutatás. A backend nem érhető el a végfelhasználók számára, és láthatatlan marad. Az alkalmazás gerinceként működik, és olyan programozási nyelvek segítségével fejlesztették ki, mint a PHP, a Ruby, a Python, vagy olyan keretrendszereket, mint a Django, a Spring Boot vagy a NodeJS.

### React és a Node.js (Express)

Ki használja a Node-ot React háttérként? A Walmart [5] az első nagyvállalatok között volt, amely profitált a React.js és a Node.js együttes használatáról. 2015-ben a vállalat mintegy 25 alkalmazást, amelyek e-kereskedelmi webhelyüket működtették, Java-ról a React és a Node technológiai csomagra költöztette.

Az Express egy gyors és minimalista keretrendszer, amely a NodeJS-re épül. Ez lehetővé teszi a REST API-k egyszerű létrehozását azáltal, hogy meghatározhatja a kiszolgáló különböző kéréseihez futó kódot. A NodeJS egy nyílt forráskódú, Javascript futásidejű környezet, amely a Google Chrome V8 JavaScript motorjára épül. Más szavakkal, a NodeJS a Google V8 Javascript motorját használja a kód végrehajtására. A Node.js és a React.js együttes használatának több előnye is van egy single page applikáció megírásánál. A technika ezen kombinációja lehetővé teszi, hogy:

* SEO-barát (a Google és más keresőmotorok hatékonyan feltérképezhetik a webhely minden oldalát, a keresett témák alapján a legrelevánsabb és legértékesebb weboldalakat szolgálhatják ki felhasználóik számára) single page applikációkat hoz létre, amelyek kihasználják az alkalmazás nézetek szerveroldali megjelenítését.
* Ugyanazt a V8 JavaScript-motort használja kliens és szerveroldali megjelenítéshez.
* Használja az npm kódcsomagokat a webalkalmazás-fejlesztési ciklus felgyorsítása érdekében.
* Használja a Node.js modulokat a React alkalmazás egyetlen fájlba csomagolásához.

Az Express.js nyílt forráskódú harmadik fél moduljaként kerül kiterjesztésre a Node.js fájlhoz. Sokkal kisebb, mint Django vagy a Ruby on Rails. Azáltal, hogy csak egy vékony absztrakciós réteget ad a Node.js szerver tetejére, kisimítja a webfejlesztést anélkül, hogy bizonyos alkalmazásstruktúrát kényszerítene. Mivel a React könyvtár célja az építészeti szabadság, az Express.js-szel együtt történő használata lehetővé teszi magasan testreszabott alkalmazások készítését. Az Express.js keretrendszer könnyű és moduláris. Alapja csak olyan alapvető funkciókat tartalmaz, mint például az útválasztási (routing) rendszer, a munkamenetek és a cookie-k támogatása, a MIME segítők, a RESTful felület stb., amelyek szükség esetén opcionális modulok vagy köztes eszközök hozzáadásával bővíthetők. Ennek a keretrendszernek egy másik előnye a nagy sebesség és az egyidejű kapcsolatok ezreinek másodpercenkénti kezelése. A teljesítménye emellett javítható gyorsítótárazás vagy több folyamatra skálázás révén. Mivel ez a produktív, az Express.js aligha válhat szűk keresztmetszetévé egy nagy terhelésű React alkalmazás számára.

### Laravel és a Ruby on Rails összehasonlítása

Minden bizonnyal megfelelő választás az Express a feladat megvalósítására backend oldalról, azonban érdemes megnézni a további lehetőségeket is. Két elég gyakori React-hez való backend párosítást szeretnék megvizsgálni, amelyek a Laravel és a Ruby on Rails.

A Laravel egy modern webes framework [6], amellyel egyszerűen és gyorsan lehet tervezni webes alkalmazásokat. A webes alkalmazás testreszabását ugyanolyan könnyedén tudja elvégezni, mint például az Express. Ez egy nyílt forráskódú és gyakran használt keretrendszer PHP-ben. Mivel nyílt forráskódú, forráskódját a GitHub tárolja. Számos olyan funkcióval rendelkezik, amelyek vonzhatják a fejlesztőt egy webes alkalmazás létrehozásához Laravelben. Skálázhatósága és teljesítménye nagy hangsúlyt kapott a fejlesztők közösségében.

A Ruby on Rails egy olyan keretrendszer, amely sok kész funkcióval segíti a fejlesztőt, így a fejlesztő egyszerűen az üzleti logikára tud koncentrálni, ahelyett, hogy elmélyülne a keretrendszer bonyolultságaiban. Ez egy szerver oldali webes alkalmazáskeret, amelyet Ruby-ban hoztak létre. A két keretrendszer különbségeit az alábbi táblázat foglalja össze szemléltetően.

|  |  |  |
| --- | --- | --- |
| **Az összehasonlítás alapja** | **Laravel** | **Rails** |
| **Nyelv** | PHP alapú | Ruby alapú |
| **Szintaxis** | Pontosvesszőt és kapcsos zárójelet használ | Nem támogatja a pontosvesszőt és a kapcsos zárójeleket |
| **Használat** | Vállalati szintű alkalmazás kis oldalaknak | Adatbázis-alapú webalkalmazás, meta-programozás |
| **Létezés** | Első megjelenés 2011 júniusában | 2004-ben érkezett |
| **Sablon motor** | A Blade a sablonmotor | Az ERB az alapértelmezett motor |
| **ORM** | Eloquent a használt ORM | Az ActiveRecord a használt ORM |
| **Jellemzők** | Gyors ORM, saját CLI, hitelesítési lehetőségek | Tiszta szintaxis, harmadik féltől származó beépülő modulok, jó MVC-támogatás |

Jól látható, hogy mindkét keretrendszernek megvannak a maga sajátosságai, amelyek egyedivé teszik őket a maguk módján. A Rails olyan keretrendszer, amely alkalmasabb kis méretű projektekhez, jó dokumentációval, útmutatókkal és kiterjedt könyvtárakkal rendelkezik. Érkezése a webes tájra korábbi, mint a Laravel; így közössége erősebb és népszerűbb a fejlesztők körében. A Ruby egyben jól olvasható nyelv; kódolása könnyebben olvasható és írható, mint bármely más hagyományos objektumorientált nyelv. A Laravel viszont tele van olyan funkciókkal, amellyel alkalmas bármilyen webes alkalmazás létrehozására. ORM-eszköze, az Eloquent és a CLI-eszköz, az Artisan, lehetővé teszi a fejlesztők számára, hogy különböző feladatokat hajtsanak végre a gyorsítótár törlésével, így könnyedén migrálják az adatbázist. Dokumentációja alapos, és a nyelv minden aspektusára kiterjed. A webalkalmazások hitelesítési funkcióját hatékonyan támogatja a Laravel.

### A backend kiválasztása

A backend kiválasztása a legfontosabb teendő, [7] mivel ez felelős az oldal teljesítményéért. Hogyan válasszuk ki a megfelelő háttértechnológiát a webalkalmazásához? A választást még megnehezíti, hogy nem lehet megmondani, melyik nyelv, keretrendszer, futtatókörnyezet jobb a másiknál – mindegyiknek megvannak a maga előnyei és korlátai. Egyes projektekben a különböző technológiák egymással felcserélhetően használhatók. Más esetekben ez lehetetlen lenne, ezért olyan fontos a megalapozott döntés. A fent leírtak alapján én a Node.js és Express párosítást választottam, mint az előzetes ismereteim alapján, mint pedig a kutatás által ez bizonyult a legmegfelelőbbnek számomra.

## Adatbázis

A feladatom esetében már meglévő adatbázisból dolgozva kell az adatokat megjeleníthetővé, szerkeszthetővé és áltáthatóvá tennem adminisztrációs feladatok elvégzésének céljából. Ez az adatbázis PostgreSQL alapú, amely mára az egyik legnépszerűbb nyílt forráskódú adatbázis lett.

### PostgreSQL

A PostgreSQL jól működik minden modern webes keretrendszerrel, mint például a backend tervezésnél kiválasztott Node.js. Támogatja a SQL-t és a JSON relációkat, [8]továbbá a fejlett adattípusokat és teljesítményoptimalizálási szolgáltatásokat, amelyek csak drága kereskedelmi adatbázisokban érhetők el, mint például az Oracle és az SQL Server. Tapasztalt fejlesztői közösség támogatja, akik óriási hozzájárulást tettek ahhoz, hogy rendkívül megbízható DBMS-rendszerré váljon. Mellesleg a Postgres megtanulásához nincs szükség sok képzésre, mivel könnyen használható.

## A technológiák kiválasztásának összegzése

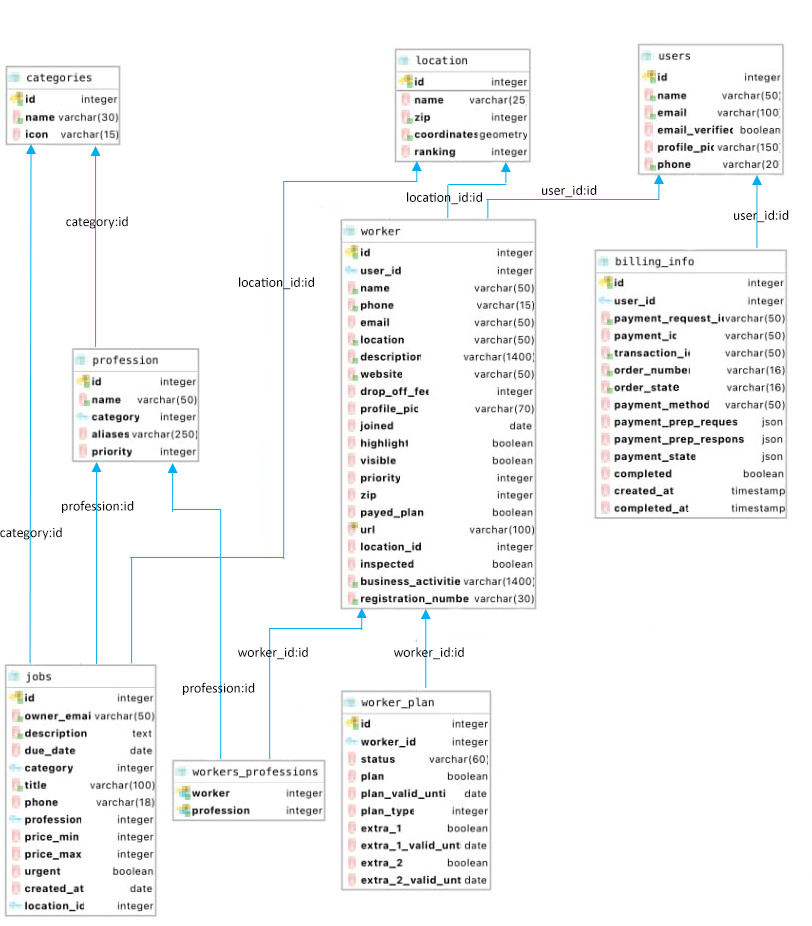
Az adatbázis már adott volt a feladatom esetében, azonban a teljesség érdekében röviden bemutattam a PostgreSQL relációsadatbázis-kezelő rendszert. A legnagyobb kérdés a frontend keretrendszer kiválasztása volt. Itt a React-et választottam, mind az előzetes ismereteim, mind pedig a személyes preferencia miatt. A React mellé az 5.2.1. fejezetben leírtak alapján jól látható, hogy a Node.js- Express.js a leginkább illő backend technológia. Tehát, ezeknek a keretrendszereknek a segítségével fogom az alkalmazásomat implementálni.

# A feladat megvalósítása

## Adatbázis bemutatása

A programom közös adatbázist használ a mestertkeresek.hu-val, amely nagyon is lehetséges, és valójában ez egy normál működési mód. Az adatbázist szempontjából nem lényeges, hogy ki az „ügyfél”. Ez lehet egy webalkalmazásból származó kapcsolat, vagy egy vastag kliens, vagy akár egy szerver alapú folyamat. Az adatbázis számára ezek mindegyike csak egy másik felhasználó, és amint az kliens által kért adatbázis-feladat befejeződött, az adatbázis türelmesen vár egy másik feladatra.

Ez az adatbázis meglehetősen hatalmas és szerteágazó, azonban az alkalmazásom ezekből, csak pár fontosabb táblát használ. Így csak ezekre fókuszálva próbálom bemutatni az adatbázis-táblákat és kapcsolatokat.



6. ábra Adatbázis relációs sémája

## Tervezés

//?

## Implementáció

//backend: config, controller, storage

//frontend: material, komponensek,

# Összefoglalás

//todo: osszefoglaló megírása

Irodalomjegyzék

|  |  |
| --- | --- |
| [1] | Gremmédia, „https://gremmedia.hu/,” 05 01 2021. [Online]. Available: https://gremmedia.hu/angular-react-vagy-vue-melyiket-lehet-erdemes-hasznalnunk. |
| [2] | M. Schwarzmüller, „https://academind.com/,” 19 03 2020. [Online]. Available: https://academind.com/tutorials/angular-vs-react-vs-vue-my-thoughts/. |
| [3] | [Online]. Available: https://rawgit.com/krausest/js-framework-benchmark/master/webdriver-ts-results/table.html. |
| [4] | S. W. Sapnesh Naik, „https://blog.crowdbotics.com/,” 13 10 2020. [Online]. Available: https://blog.crowdbotics.com/most-compatible-frontend-backend-framework-pairings/. |
| [5] | P. H. Kseniia Kyslova, „https://proxify.io/,” 21 01 2021. [Online]. Available: https://proxify.io/articles/node-and-react. |
| [6] | P. Pedamkar, „Educba,” [Online]. Available: https://www.educba.com/laravel-vs-ruby-on-rails/. |
| [7] | A. Yushkevych, „Monovm,” 15 06 2021. [Online]. Available: https://monovm.com/blog/backend-development-how-to-choose-the-right-framework/. |
| [8] | R. Peterson, „Guru99,” 7 10 2021. [Online]. Available: https://www.guru99.com/introduction-postgresql.html. |
| [9] | K. Shan, „https://www.thirdrocktechkno.com/,” 23 06 2020. [Online]. Available: https://www.thirdrocktechkno.com/blog/single-page-apps-vs-multi-page-apps-what-to-choose-for-web-development/. |
| [10] | P. Borrelli, „LogRocket,” 26 10 2021. [Online]. Available: https://blog.logrocket.com/angular-vs-react-vs-vue-js-comparing-performance/. |