

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](#_Toc87124483)

[1.1. Probléma megfogalmazása 4](#_Toc87124484)

[1.2. A dolgozat célja 4](#_Toc87124485)

[2. Koncepció 5](#_Toc87124486)

[2.1. Megoldandó feladatok, elsődleges célok 5](#_Toc87124487)

[3. Követelmény analízis 7](#_Toc87124488)

[3.1. Befizetések kezelése 7](#_Toc87124489)

[3.2. Alap statisztikák megjelenítése 7](#_Toc87124490)

[3.3. Adatok megtekintése és szerkesztése 7](#_Toc87124491)

[3.4. Szakember ellenőrzése 8](#_Toc87124492)

[4. Specifikáció 9](#_Toc87124493)

[4.1. Az felhasználói felületen található funkciók 9](#_Toc87124494)

[5. Technológiák kiválasztása 10](#_Toc87124495)

[5.1. Frontend 10](#_Toc87124496)

[5.1.1. Keretrendszerek áttekintése 10](#_Toc87124497)

[5.1.2. React, Angular és a Vue.js - alap gondolatok 11](#_Toc87124498)

[5.1.3. React, Angular és a Vue.js – funkcióinak összehasonlítása 12](#_Toc87124499)

[5.1.4. React, Angular és a Vue.js – szintaxisának összehasonlítása 13](#_Toc87124500)

[5.1.5. React, Angular és a Vue.js – komplexitásának összehasonlítása 14](#_Toc87124501)

[5.1.6. React, Angular és a Vue.js – teljesítményének összehasonlítása 15](#_Toc87124502)

[5.1.7. Frontend kiválasztása 16](#_Toc87124503)

[5.2. Backend 16](#_Toc87124504)

[5.2.1. React és a Node.js (Express) 16](#_Toc87124505)

[5.2.2. Laravel és a Ruby on Rails összehasonlítása 18](#_Toc87124506)

[5.2.3. A backend kiválasztása 19](#_Toc87124507)

[5.3. Adatbázis 19](#_Toc87124508)

[5.3.1. PostgreSQL 19](#_Toc87124509)

[5.4. A tervezés összegzése 20](#_Toc87124510)

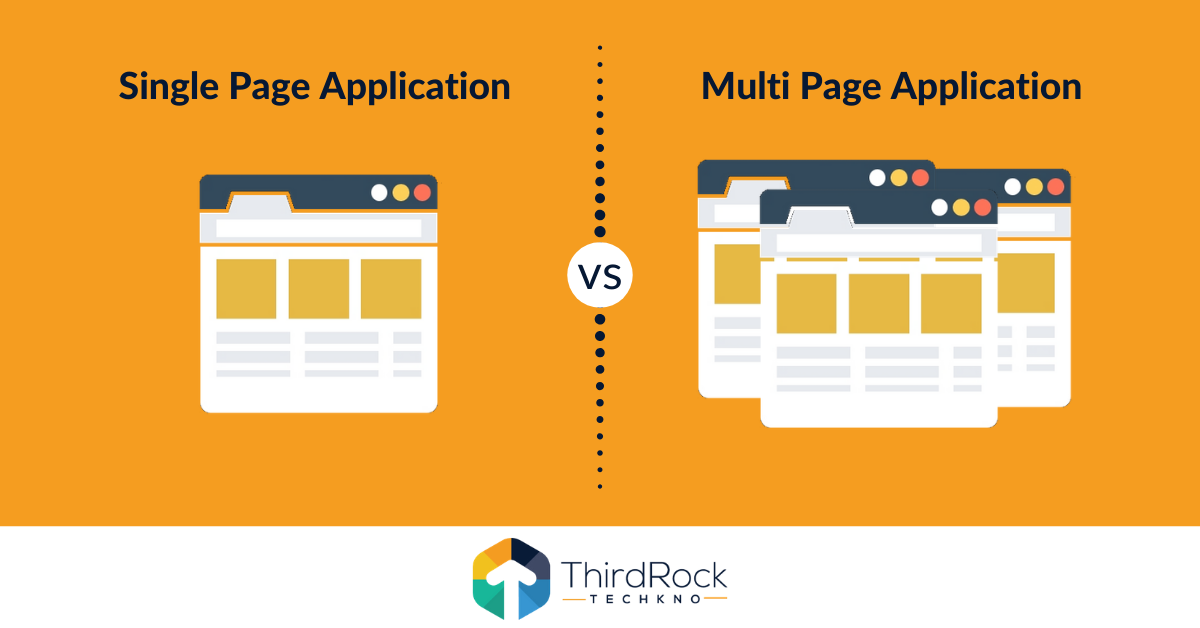
# Bevezetés

## Probléma megfogalmazása

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 könnyedén tudja kezelni. A rendszernek alkalmasnak kell lennie az adatok megtekintésére és módosítására, illetve általános információkat kell tudnia adni a vállalkozásról.

## A dolgozat célja

Egy olyan single-page application (SPA) megvalósítása a cél, 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 megtekinthető kell legyen web-böngészőben.



1. ábra Single Page és Multi Page Applikáció [1]

# Koncepció

Egy már létező szakemberkereső oldal a mestertkeresek.hu, mely vállalkozáshoz tartozó adminisztrátori munkavégzését megkönnyítendő, erre kifejlesztett webes applikáció létrehozása a cél. Szakdolgozatom írásával törekszem ennek az alkalmazásnak a tervezését, létrehozását és működését teljeskörűen bemutatni.

Ahhoz, hogy megoldást találjunk, először mélyebben bele kell ásni magunkat a problémába. A már működő, fentebb 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, elsődleges célok

A cél egy olyan webes alkalmazás létrehozása, amely a mestertkeresek.hu adminisztrációs felületéül szolgál. A software elsődleges feladata, hogy a mestertkeresek.hu-n történő befizetéseket adminisztrálja, kisebb statisztikát készítsen ezekről, illetve a regisztrált felhasználók adatait kezelje.

Regisztráció során sok különféle adat kerül eltárolásra az adatbázisban egy felhasználóról, az ezekhez való hozzáférést könnyedén el kell látni a rendszernek. Továbbá a fizetési tranzakciók ellenőrzést igényelnek, különösképp egy banki átutalás. Ezeknél nagy segítség, ha a program gombnyomásra át tud írni adatokat, akár több táblában is.

Számos esetben 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.

# 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 tehetik meg a felhasználók vagy online bankkártyás fizetéssel.

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. Ez mind az eladások számát, mint pedig a pénzösszeg napi, heti és havi lebontását tudnia kell reprezentálni. Így sokkal egyszerűbb meghatározni a vállalkozás 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.

## Adatok megtekintése és szerkesztése

Az adatok megtekintésére és szerkesztésére való lehetőség egy alapvető feladat, amelyet szintén el kell látni a programnak. Sokkal egyszerűbb egy vizuális felületen kezelni az adatbázis tábláit, mint SQL lekérdezésekkel szemügyre venni, illetve változtatni azokon. Elsősorban a szakemberek és további felhasználók listáit szükséges itt megjeleníteni.

## 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).

# Tervezés

## Funkciók

A 3. fejezetben említett követelmények alapján terveztem meg az alkalmazás funkcióit. A következőkben sorra veszem, hogy mik azok, amikre képesnek kell lennie az alkalmazásnak, illetve mire hogyan kell reagálnia az egyes esetekben. Ezek megvalósítását egy későbbi fejezetben fogom részletesen bemutatni.

### Megrendelések

A programnak átláthatóan meg kell tudni jeleníteni az összes megrendelést, vagyis a mestertkeresek.hu oldalon történő csomagvásárlást, mely feljogosítja az előfizetőt, hogy hirdetést adjon fel. Ezeknek a kezelésére egy olyan megoldásra van szükség, ahol látni lehet a megrendelések részleteit, mellyel az alkalmazásom segítségével egy adminisztrátor képes lesz a felhasználót Ellenőrzött mester státuszba léptetni, amelyet a 3.4. fejezetben fejtettem ki. Továbbá itt kell tudni az ügyintézőnek ellenőriznie a banki átutalásokat.

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.

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.

### Alap adatok

Alap adatok megjelenítésénél a 3.2. fejezetben leírtakból következik, hogy mire van szükség. Az alkalmazásban szerepelnie kell az eladások darabszámának és a bevétel forint alapú kiírásának, illetve ezeknek ciklusokra bontott reprezentálásának. Mindezeknek az előző periódushoz képesit százalékos megjelenítését is képesnek kell lennie 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.

//todo: csomagok grafikon

### Adatok szerkesztése

# 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 [2] 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 [3]. Mindhárom könyvtár jó okokból nagyon népszerű. Mindegyiküknek megvannak az erősségei és gyengeségei, és általában bármelyik könyvtárat fel lehet használni bármely projekthez. Nagyon fontos, hogy szereted-e a keretrendszert / könyvtárat, vagy sem. Ha tetszik a szintaxisa, a dolgok megközelítésének módja, és ha tetszik, az adott keretrendszerben írt kód az merőben megkönnyíti a munkát. Ha nem tetszik egy technológia, akkor nem tudsz vele sikeresen dolgozni!

* Az Angular a Google által kifejlesztett keretrendszer ami 2010-ben jelent meg. A Google belsőleg is használja az Angulart, ezért nem fogjuk látni, hogy az Angular eltűnik egyik napról a másikra. 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 „fiatalabb” az Angularnál.
* A Vue egy önálló projekt, amelyet egyetlen vállalat sem épít. Korábban egyszemélyes show volt (Evan You, alapítója), de ezek a napok már régen elmúltak. Manapság elkötelezett munkatársa van, 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. Ezenkívül létezik egy hivatalos CLI, amely segít 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 számára, és azt hatékonyan ellenőrzi. Ez az összetevőkről és a felhasználói felületek komponensekből való építéséről szól. 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. Nem tartalmaz útválasztót (router) és nem szállítja a saját Http-kliensét. Ezenkívül nem jár más különleges funkció hozzá, és ebben a tekintetben határozottan „karcsúbb”, mint az Angular. Ezeknél a funkcióknál a React közösségre kell támaszkodni.

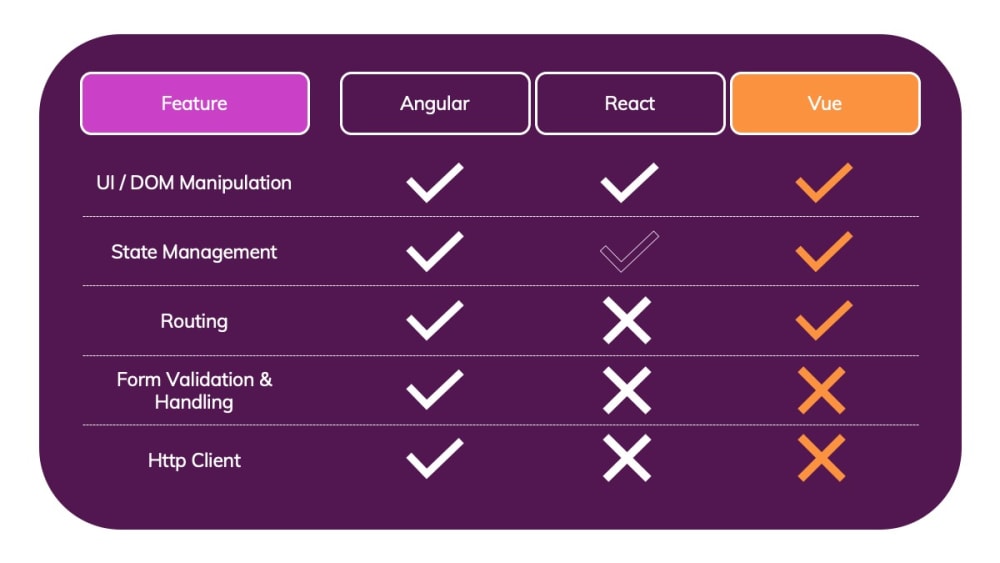
A Vue egy olyan keret, amely a React és az Angular között helyezkedik el. Nem olyan "nagy", mint az Angular, de határozottan több funkciót tartalmaz, mint a React. A Vue beépített állapotkezelést biztosít, és beépített útválasztóval (router) is rendelkezik. Nem tartalmazza azonban az űrlap-hitelesítést 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? Végül is 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 preferenciáitól. Az Angular és a Vue teljes beépített támogatással rendelkezik, hogy a munkát könnyebbé tegye. A React csak beépített állapotkezelési támogatással rendelkezik, és nincs beépített útválasztási támogatása.

Akkor miért nem lenne jobb az Angular (vagy a Vue), mint a React? 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 „eszközhözt”, amelyekre végül szükséged lesz, megkapod a nagyon aktív React közösségétől. 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 neked, míg a közösség egyedi projektekre összpontosít, amelyek kiegészítik ezt a könyvtárat. Nem rossz megközelítés.

Az Angular csapat természetesen hatalmas tapasztalattal rendelkező fejlesztőkből álló csapat, így valószínűleg nem kell tartanunk 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.



2. ábra Keretrendszerek és funkcióik [3]

Végül tehát, amint azt korábban említettem, itt nincs jobb megközelítés. Rajtad múlik és a projekteken, hogy melyik megközelítést részesíted előnyben. Bármely projektet fel lehet építeni a három nagy könyvtár bármelyikével.

### 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! Ha nem tetszik az, ahogyan a kódot kell írni abban a bizonyos keretrendszerben (vagy könyvtárban), akkor azzal nem lesz élvezetes dolgozni.

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 összeáll a rendszeres, böngészőbarát JavaScript kóddal a kulisszák mögött.

A Vue normál JavaScriptet használ (bár opcionálisan használhatja a TypeScriptet is), és általában „Single File Components” -nek hívnak, amit kezel. Ez egy olyan keret, amely végül is az komponensekről szól (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 applikálja a kódot.

Melyik szintaxis a legjobb? Már lehet sejteni, hogy: Nincs a legjobb szintaxis. 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. Vagy van aki rajong a kódfelosztásért, 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 „kulisszák mögötti” 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-reago-app a React projektekhez. Az Angular esetében ezután meg kell tanulnia a TypeScript-t is, a React-hez is meg kell ismerni JSX-szintaxisár, 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 alapokat kell futtatni, 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 kódja + 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. Ilyen összehasonlításokban az Angular általában valamivel lassabb, mint a Vue és a React alkalmazások - a Vue pedig valamivel gyorsabb, mint a React. De megint nem szabad túl sokat levonni ezekből. 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 gyorsan érzi magát 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. Ehelyett olyan technikákat is felhasznál, amelyek során bizonyos feladatokat prioritássá tesz, hogy a betöltött oldal gyorsabban erződjön a végfelhasználó számára. Előfordulhat például, hogy elsőbbséggel kezeli a felhasználói bevitelt, és késlelteti a képernyőn megjelenő szövegek frissítését. Összességében mindhárom keretrendszer nagyszerű futásidejű teljesítményt nyújt.

### 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 Reactot 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 Reactben 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 tervezés összegzése

Irodalomjegyzék

|  |  |
| --- | --- |
| [1] | 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/. |
| [2] | Gremmédia, „https://gremmedia.hu/,” 05 01 2021. [Online]. Available: https://gremmedia.hu/angular-react-vagy-vue-melyiket-lehet-erdemes-hasznalnunk. |
| [3] | M. Schwarzmüller, „https://academind.com/,” 19 03 2020. [Online]. Available: https://academind.com/tutorials/angular-vs-react-vs-vue-my-thoughts/. |
| [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. |