

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 3](#_Toc71830085)

[1.1. Probléma megfogalmazása 3](#_Toc71830086)

[1.2. A dolgozat célja 3](#_Toc71830087)

[2. Koncepció 4](#_Toc71830088)

[2.1. Az problémák áttekintése 4](#_Toc71830089)

[2.1.1. Megoldandó feladatok 4](#_Toc71830090)

[3. Tervezés 5](#_Toc71830091)

[3.1. Frontend tervezése 5](#_Toc71830092)

[3.1.1. Keretrendszerek áttekintése 5](#_Toc71830093)

[3.1.2. React, Angular és a Vue.js - alap gondolatok 6](#_Toc71830094)

[3.1.3. React, Angular és a Vue.js – funkcióinak összehasonlítása 7](#_Toc71830095)

[3.1.4. React, Angular és a Vue.js – szintaxisának összehasonlítása 8](#_Toc71830096)

[3.1.5. React, Angular és a Vue.js – komplexitásának összehasonlítása 9](#_Toc71830097)

[3.1.6. React, Angular és a Vue.js – teljesítményének összehasonlítása 10](#_Toc71830098)

[3.1.7. Frontend kiválasztása 11](#_Toc71830099)

[3.2. Backend tervezése 11](#_Toc71830100)

[3.3. React és a Node.js (Express) 11](#_Toc71830101)

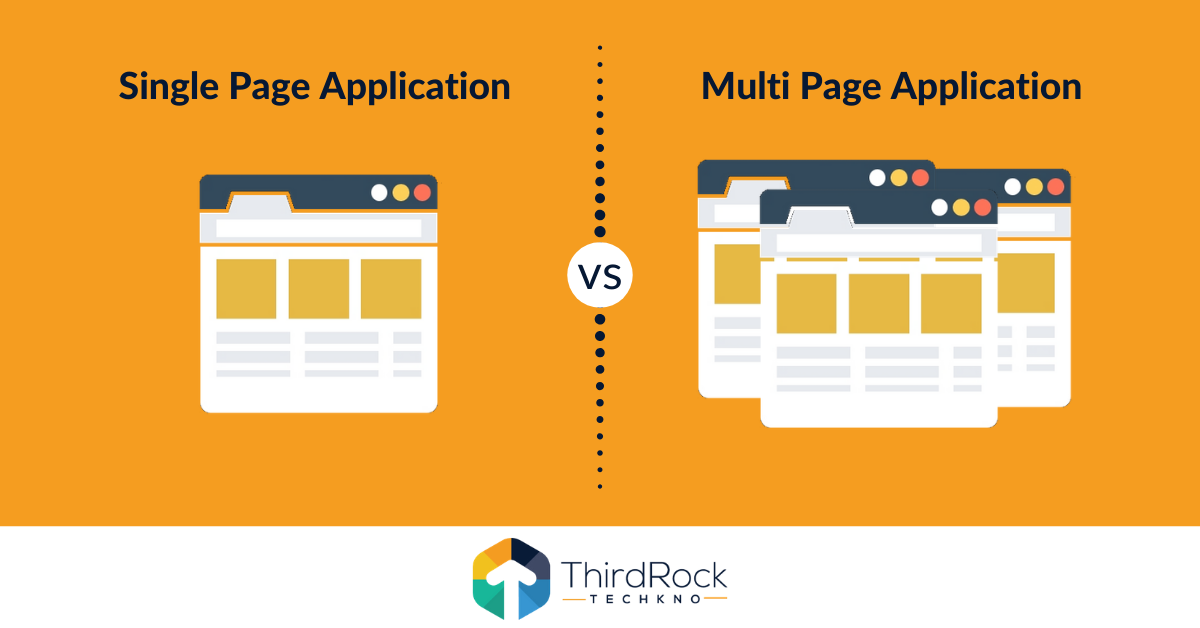
# Bevezetés

## 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

Ahhoz, hogy megoldást találjunk, először mélyebben bele kell ásni magunkat a problémába. A már létező adatbázist felhasználva kell információkat megjeleníteni és szerkeszthetővé tenni a felhasználó számára. De miért is van erre szükség? Az adatok szerkesztése fejlesztői 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. Ehhez könnyedén áttekinthető felületre van szükség, amelynek használatát pár perc alatt el tudja sajátítani a felhasználó. Így a pénzügyi, marketing és adminisztrátori munkavégzés kiadható nem fejlesztők kezébe is.

## Az problémák áttekintése

Nagyon sok cég mai napig Excelben vagy egyszerű CSV fileokban tárolja az adatait. Ez azért rossz, mert sokszor átláthatatlanok és sérülékenyek. Egy szakemberkereső oldal komplex adatbázisban tárolja adatait és ezt felhasználva kell a rendszert felépíteni úgy, hogy az adatok integritása megmaradjon a relációs adatbázisban a szerkesztés után is.

### Megoldandó feladatok

Regisztráció során sok különféle adat tárolódik el az adatbázisban egy felhasználóról. 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 egyszerre is.

Sokszor előfordul az emberekkel, hogy elgépelnek, hibásan írnak le valamit és ezt észre sem veszik. Ezek átírását, javítását is egyszerűen el kell tudni végezni, ezzel a szakemberkereső oldal színvonalát lehet növelni, hogy nincsenek feltűntetve hibásan leírt mondatok.

# Tervezés

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 tervezése

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.

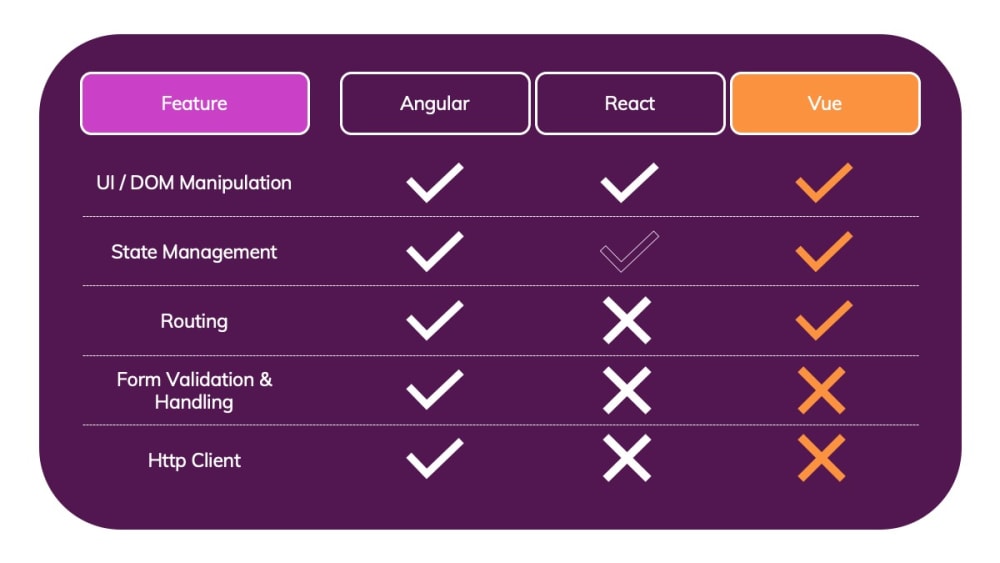
A Vue egy olyan keret, amely a React és az Angular között helyezkedik el. Nem olyan "nagy", mint az Angular, de határozottan több funkciót tartalmaz, mint a React. A Vue beépített állapotkezelést biztosít, és beépített útválasztóval (router) is rendelkezik. Nem tartalmazza azonban az űrlap-hitelesítést 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 tervezése

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.

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. |