Web technológiák 1 - Beadandó dokumentáció

SunComplex kft. cégnek készített “Single page application” weboldal
React környezetben

Miliczki József
Y86I0I
2022. 10. 01.

Bevezetés

A weboldalt Grebely Zoltán - és egyben a SunComplex Kft cég - megrendelésére készítettem, abból a megbízásból, hogy legyen egy webes felület amely bemutatja a cég működését, valamint alapvető árajánlatokat nutasson a potenciális vevők számára. A célokból kiindulva a single page megközelítést választottam, mivel az elvárások nem kívánták meg a több oldalas megközelítést, valamint nyári gyakorlatom során rengeteget programoztam React-ban (javascript és typescript is!), amely ezen stílusú weboldalak készítésére specializálódott. A weboldal 2 hét munkát vett igénybe, ez a tervezéstől kezdve a megrendelői változtatások implementálásáig mindent tartalmaz. A weboldalt a Rackhost.hu szerverein működtetem, itt vásároltam hozzá domain-t is. Az oldal ezen a linken elérhető: www.suncomplex.hu

Az oldal felépítése

A gyökéren és InfoSection-ön kívül minden oldal 2 javascript fileból tevődik össze: Egy index.js, amely a szekció felépítését írja le, valamint az xyElements.js, (ahol xy a komponens neve), amelyben definiáljuk a CSS stílusait azon elemeknek, amelyeket felhasználjuk az index.js-ben, mint építőelemeket. Én szeretek azzal a hasonlattal élni, hogy a lego építőkockákat itt definiáljuk! Az Elements file-okban 2 érdekes dolgot érdemes megemlíteni: Néhány helyen boolean változók alapján változtatjuk meg a stílust, ez viszonylag ritka a kódban, viszont például az összes InfoSection ezen alapszik, így nem kell 2 CSS file-t definiálnunk sötét és világos panelek rendereléséhez. A másik érdekesebb rész pedig a mobil optimalizálás, ezt @media taggel oldjuk meg, ahol a képernyő szélességét vizsgáljuk. Bizonyos szélességnél átváltunk telefon módra. Egy szépséghiba: Nem számolhatok minden eszközzel, ezért például a tabletek nem túl stílusosan mutatják az oldalt, viszont széles körben teszteltem a weboldalt, és kielégítő eredményekkel találkoztam telefon optimalizálás terén.

A gyökér (`pages/index.js`)

Ez a file fogja össze az egész projektet, itt vannak sorba téve azon komponensek, amelyek az oldalt felépítik. Itt kezeljük a navigációs fejléc és a telefonos oldalsó menü megjelenítését, valamint itt adjuk át a szöveges - valamint konfigurációs - tartalmát azon komponenseknek, amelyek ezekkel az adatokkal dolgoznak. Ilyen komponensekből áll a projekt nagy része, mivel egyszerűbb külső fájlban definiálni a konfigurációt és a string-eket, mint a komponensen belül ezzel bajlódni. A következőkben sorban fogok menni a komponenseken amelyek ezen file-on belül kerülnek meghívásra.

Sidebar és Navbar

Ezt a 2 komponenset együtt veszem, mivel funkcionalitásukat véve ugyanazt a szerepet töltik be: Az oldal különböző szekcióniak egy kattintással való elérését valósítják meg, úgynevezett navigációs felülettel. A Navbar a számítógépes felülethez tartozik, míg a Sidebar a telefonos weboldal megjelenítésnél lép életbe.

A Navbar vizuálisan látványosabb, mivel a React smooth scroll-t kihasználva az oldal “csúszkál” a különböző szekciók között, ahelyett hogy rögtön odaugrana az oldalon. A Sidebar hasonlóan működik, ám az animációt nem lehet megfigyelni telefonon, mivel mire eltűnik a Sidebar, már ott vagyunk a kiválasztott menüponton. A Navbar még abban is kitűnik, hogy egy logót alkalmazok, mint lap tetejére mutató navigációs gomb. Ezt sok weboldalon így csinálják, gondoltam itt is jó hatással fog bírni. Ez a funkció hiányzik a Sidebar-ból. Amit érdemes még megemlíteni a Navbarral kapcsolatban, az a fixált elhelyezkedése: erről már gyakorlaton is esett szó, pontosan az ott bemutatott módszerrel oldottam meg a navigációs menü folytonos “fentmaradását”.

Hero Section

A Hero Section-t mint fogalmat az interneten láttam, ennek nevezték el azt a fogadó felületet amely minimális szövegből és egy háttérből állt. Ez a szekció nem más, mint a felhasználót fogadó felület, ezért cél a letisztultság. A háttér a működésbe helyezett weboldalon egy kép, ám a fejlesztés elsősorban videóval történt. Sajnos szerver oldali limitációk miatt a videó túl sok ideig töltött, így ezt az ötletet le kellett cserélnem az adott videóból kivágott képkockára. A változtatás mai napig visszakövethető a mappa nevéből, ami a szerver oldalon még mindig "video"-nak van elnevezve, viszont már csak egy háttérképet tartalmaz Ezt az elnevezési lustaságot a GitHub-os verzióban már fixáltam! A háttérképen kívül definiáltam még egy személyre szabott H1 és P elemeket is, amelyek a fogadó szöveget írják ki.

Start

Ez a komponens szolgál kezdőoldalként, már ha a Hero Section-t nem vesszük annak. Itt mutatjuk be a fő árajánlatokat, kisebb "kártyákba" téve. Ezek a kártyák igazából a Services résznél kerültek implementálásra először, a Start section a fejlesztés utolsó negyedében készült, és úgy alakult, hogy újra felhasználható a kód egy része, így egy kis változtatással újra implementáltam a Services bizonyos részeit. Ebből alakult például a StartCard komponens, valamint a StartFooterCard! Ezt a komponenset volt a legnehezebb mobilos eszközökre implementálni, mivel a kártyák ide-oda csúszkáltak, így sok idő elment az optimalizálással.

InfoSection

A weboldal legnagyobb tartalmi részét ez a komponens teszi ki. Ez magyarázza el a napelem beszerelés folyamatát. Lényegében egy InfoSection tartalmaz egy képet, egy kisebb információs szöveget (az oldalon ez a "folyamat"), egy címet, valamint egy rövid szöveges tartalmat. A képeket az unDraw illusztrációkat tartalmazó felületről

szereztem be, itt meg lehet adni milyen színnel szeretnénk kitölteni a képek színes részét. Rengeteg választás van, csak be kell írnunk pár kulcsszót, és kapunk a megadott szavakhoz releváns képeket.

Mivel az InfoSection teszi ki az oldal legnagyobb százalékát, így legnagyobb fontosság a kód újrafelhasználhatósága. Az InfoSection tartalmaz egy bónusz file-t, amit Data.js néven találhatunk meg. Ebben írjuk le az összes InfoSection tartalmát és konfigurációját, és a gyökérben levő index.js-ben adjuk át ezeket a paramétereket az InfoSection-öknek. A Data.js-ben leírjuk, hogy a jelenlegi panel az "dark mode"-ban van-e (tehát sötét kék), vagy nem (tehát fehér), megadhatjuk hogy a szöveg sötét legyen-e vagy világos, de itt adható meg a szöveges tartalom, a képek elérési útja és alternatív szövege is. Ezzel a megoldással nem kell új InfoSection-öket gyártani, szimplán csak meg kell írni az újabb panelt a Data.js-ben, és máris van egy újabb InfoSection részünk. A megbízó elvárásai szerint a weboldalt tovább is bővíteni fogjuk, ezért ez lényegesen megkönnyíti az új információk felvitelét a jövőben.

ExtendedInfoSection

A folyamat leírása után egy újabb panelt kell létrehoznunk, ahol leírjuk, hogy a vevők milyen szolgáltatásokat várhatnak el a cégtől. Erre személyes véleményem szerint nem volt megfelelő az épített InfoSection, ezért egy kibővített verziót hoztam létre, amely egy 2x2-es grid-en valósítja meg az InfoSection-t. A trükk az szimplán az, hogy több sort is használhatunk egy InfoSection-ön belül. Mivel a kopmponenst az InfoSection-re alapoztam, így nem meglepő, hogy a Data.js-ben definiált paraméterek itt is használhatóak, ezzel csökkentve az újabb kódsorokat.

RolunkPage és RolunkPage2

A szolgáltatások után a cég történetéről kell beszámolnunk. Erre szolgál a RolunkPage, és ennek változata. Ezek is az InfoSection-re alapozott komponensek, szeretnénk a Data.js-t minél

többször alkalmazni mint centrális információ forrás. A szokásos kép-cím-szöveg hármas megmarad, ám lényegesen több hely jut a szövegnek, mivel ez a rész terjedelmesebbre sikerült. 2 RolunkPage van, a másodikat később implementáltam amikor szembesültem a kéréssel, miszerint legyen 2 kártya amely a cég teljesítményét írja le. Ezeket be kell szúrnom az egyik “RolunkPage”-be, viszont a másikba nem. Az általam választott megoldás praktikusnak talán nem nevezhető, de kézenfekvőnek igen. Végül arra a döntésre jutottam, hogy az egyik RolunkPage-en a cég logója legyen, a másikon pedig kép helyett a 2 kártya.

“Rólunk” oldalak

A kártyák animációval is rendelkeznek, bár ez csak annyi, hogy hover-nél megnő szélességük és magasságuk. Ez minden kártyánál előfordul, kivéve a Start komponensnél, ahol ezt a funkciót inkább eltávolítottam. Ízlések és pofonok!

Ceginfo és Services

Bár ez 2 külön komponens, felépítésük és funkcionalitásuk kevésben különbözik. Ezek lényegében egyszerű panelek, amelyeket kártyákkal töltöttem ki, és ezen kártyák különböző információkat mutatnak be, különböző képekkel.

A Céginformációban különleges képeket, pontosabban ikonokat alkalmazok, amelyek a React Icons csomagból származnak. A React Icons egy rendkívül egyszerű eszköz, amellyel több száz ikonból válogathatunk kedvünk szerint. Az egész ingyenes, és minőségi, mivel SVG formátumban tárolnak minden ikont. Egyszerűen csak ki kell néznünk egy ikont és annak a kódját, majd a megfelelő alkönyvtárból ki kell exportálni azt a kódú ikont, amit kiválasztottunk. Az ikon színezhető és méretezhető, ezeket a tulajdonságokat használom ki a céginformáció kártyáin. Telefonos felületen a stílus annyiban változik, hogy minden kártya egy oszlopba van szervezve, a 2x4-es grid helyett.

A Kapcsolat (Services, a refaktorálás talán ráférne erre a kódrészre, mivel ez volt a régi Szolgáltatsok menüpont a megrendelő által megkövetelt változtatások előtt) komponens kicsit egyszerűbb, csak 3 kártyát tartalmaz, viszont a Facebook és az E-mail kártyákra kattintva bónusz funkcionalitást tapasztalhatunk: Egy kis javascript segítségével kimásoljuk az e-mailt, valamint a Facebook kártyára kattintva elvisz minket a cég Facebook oldalára. Ezek úgymond egy soros javascript ínyencségek, de kényelmi szempontból megkönnyítik a felhasználó életét.

Gallery (a kódban Galery)

A weboldal legalján található galéria ad otthont a cég által készített képek prezentálásának. Több koncepció is született ezzel kapcsolatban, ám megint egy React csomagot fogok segítségül hívni: React Image Gallery. Ez a csomag egy (viszonyleg nehezen) személyreszabható galéria interfészt biztosít, amelynek csak meg kell adnunk képeket. A trükkös része a dolognak az a stílus beállítása,

mivel nem túl sok eszköz áll rendelkezésünkre a default stílus megváltoztatásához. Szerencsére a mi célunknak a csomag így is megfelelt, ám rengeteg dokumentációt és Stack Overflow-t kellett olvasni ahhoz, hogy rájöjjek hol kell befolyásolni a galéria stílusát. Egy újabb probléma viszont a képek betöltése: Minden képet egyenként hozzáadjuk? Ez nem tűnt már a fejlesztés korai stádiumában sem jó megoldásnak ezért próbáltam összerakni valami dinamikusabbat:

```
const Inverterek = importAll(require.context('../../images/gallery/Inverterek', false, /\.(png|jpe?g|svg)$/));
const Parkok = importAll(require.context('../../images/gallery/parkok', false, /\.(png|jpe?g|svg)$/));
const Tetok = importAll(require.context('../../images/gallery/tetok', false, /\.(png|jpe?g|svg)$/));
const images = [];

for (const img of Tetok)
{
  images.push({original: img, thumbnail: img})
}

for (const img of Parkok)
{
  images.push({original: img, thumbnail: img})
}

for (const img of Inverterek)
{
  images.push({original: img, thumbnail: img})
}
```

Végül ez a megoldás született, amely beinportál minden képet, majd egyenként bepush-olja egy images listába. Így nem kellett foglalkozni semmi féle név konvenciókkal, valamint a képek kiterjesztésén sem kellett változtatni png-től eltérő formátumoknál. Mivel a kártyákat nagyon szeretem használni, így a galéria is egy nagy kártyában van, amely egy stílusos keretet biztosít a képeknek.