Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Jankó Júlia

ONLINE IDŐPONTFOGLALÓ FULLSTACK WEBALKALMAZÁS

KONZULENS

Kövesdán Gábor

BUDAPEST, 2024

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Jankó Júlia**, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző, cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2024. 12. 05

...…….
Jankó Júlia

Összefoglaló

A szolgáltatóiparban manapság már alapvető elvárás, hogy a szolgáltató rendelkezzen egy webes felülettel. A szolgáltatóipar minden területén találkozhatunk rengeteg különböző megoldással, melyek ugyanolyan vagy nagyon hasonló funkciókat látnak el, akár kozmetikai, egészségügyi, vagy fitnesz területen. Ezeken az oldalakon tájékozódhatunk a nyújtott szolgáltatásokról és időpontot foglalhatunk ezekre. Így rengeteg szolgáltató kénytelen megoldani olyan problémákat, amit az összes előtte levő szolgáltató már egyszer megoldott. A vevők szempontjából is előnytelen ez a jelenség, mivel ahány szolgáltató, annyi féle különböző weboldallal találkozhat. Az oldalakon a funkciók hasonlók, viszont az oldal megjelenése és navigáció különböző, nehézkes.

Dolgozatomban egy olyan időpont foglaló alkalmazást mutatok be, mely megoldást ad ezekre a nehézségekre. Webalkalmazásomban lehetőségük van a különböző szolgáltatóknak regisztrálni és szolgáltatást nyújtani anélkül, hogy saját weboldalt fejlesztenének. Emellett a felhasználók egy egységes felületen kezelhetik foglalásaikat, valamint válogathatnak a különböző szolgáltatók között az egyéni preferenciájuk szerint.

Megoldásomban a felhasználóknak lehetőség van keresni több szempont alapján a szolgáltatók között. Időpontot foglalhatnak az elérhető szolgáltatóknál, menedzselhetik foglalásaikat és saját profiljukat. Szolgáltatóként az általános felhasználói funkciók mellett lehetőség van az általuk nyújtott szolgáltatások menedzselésére és elérhetőségeik megadására.

Abstract

In the service industry, it is now a basic requirement that the service provider has a web interface. In all areas of the service industry, we can find many different solutions that perform the same or very similar functions, whether in the fields of cosmetics, health, or fitness. On these pages, you can find out about the services provided and book an appointment for them. Thus, many service providers are forced to solve problems that all the service providers before them have already solved once. This phenomenon is also disadvantageous from the customers' point of view, since there are as many different websites as there are service providers. The functions on the pages are similar, but the page appearance and navigation are different and difficult.

In my thesis, I present an appointment booking application that provides a solution to these difficulties. In my web application, different service providers have the opportunity to register and provide services without developing their own website. In addition, users can manage their reservations on a single interface and choose between different service providers according to their individual preferences.

In my solution, users have the opportunity to search among service providers based on several criteria. They can book an appointment with the available service providers, manage their reservations and their own profile. As a service provider, in addition to general user functions, it is possible to manage the services they provide and provide their contact information.

1 Bevezetés

Az internetnek köszönhetően akár mindennapi ügyeinket és teendőinket végezhetjük otthonunk kényelméből. Ilyen például a bankolás, home-office, utazások, események szervezése. Mivel sokan a telefonálás vagy személyes ügyintézés helyett interneten végzik teendőiket, így a szolgáltatási szektor is alkalmazkodott ehhez a változáshoz, ritka az olyan szolgáltató, akinek nincsen weboldala. Ezeken a weboldalakon megtekinthetjük a szolgáltató és a szolgáltatás részleteit, árakat, és gyakran foglalhatunk időpontot is ezekre a szolgáltatásokra.

Mivel rengeteg szolgáltató van a világon, ezért rengeteg különböző weboldallal találkozhatunk. Különbözhetnek stílusukban, elrendezésben, az oldal navigációjában, de általában mindegyiknek az a célja, hogy a látogató tájékozódhasson a szolgáltatásokról, és tudjon időpontot foglalni a szolgáltatásra. Mivel majdnem minden szolgáltatónak szüksége van egy ilyenre, ezért gyakran megoldják újra és újra ugyanazokat a feladatokat, ami idő és pénz igényes.

Ha igénybe szeretnénk venni egy szolgáltatást, a nagy választék miatt gyakran optimalizálni kezdünk a saját prioritásaink alapján. Általában szeretnénk minél olcsóbban szeretnénk szolgáltatást kapni. Gyakran fontos, hogy minél közelebb legyen ez otthonunkhoz, milyen minőségű munkát végez a szolgáltató, vagy hogy mennyire szimpatikus. Ehhez igyekszünk sok szolgáltatót összevetni, hogy kiválasszuk az optimálist, azonban ehhez először találnunk kell ilyen szolgáltatókat. Navigálnunk és tájékozódnunk kell a sok különböző oldalon, összevetni a különböző információkat.

Megoldásomban egy olyan webes alkalmazást készítettem, amely lehetővé teszi a felhasználóknak a szolgáltató keresést és időpont foglalást egy egységes felületen. A szolgáltatóknak nem kell saját webalkalmazást készítenie, hanem beregisztrálhatják szolgáltatásaikat, és jobban megtalálhatóvá válnak a felhasználók számára.

Ebben az alkalmazásban szükségem volt egy weboldalra, amely esztétikusan és könnyedén teszi elérhetővé ezeket a funkciókat a felhasználó számára. Ezt a nagyon elterjedt React frontend technológiával oldottam meg. Ezelőtt még nem fejlesztettem React technológiában, ezért fontosnak tartottam, hogy megismerjem és elsajátítsam. A felhasználói felület mellett szükségem volt egy szerveroldali alkalmazásra, mely

kiszolgálta a weboldal kéréseit, és meg tudtam benne fogalmazni az oldalon található funkciók logikáját. Erre a Spring Java alapú keretrendszert használtam, azonban Java helyett a fejlett és népszerű Kotlin programozási nyelven. A szerveroldali alkalmazásom mellé szükségem volt egy adatbázisra is, amelyben tároltam a szükséges információkat. Erre a Microsoft SQL Server által nyújtott relációs adatbázist használtam.

Az időpont foglaló alkalmazásomnál a fodrászati vagy kozmetikai szolgáltatás foglalást tartottam szem előtt, így a dolgozatomban a szolgáltató szó használatánál első sorban fodrászokra vagy kozmetikusokra utalok.

Dolgozatom Felhasznált technológiák fejezetében bemutatom a projektem elkészítéséhez használt technológiákat. A 3. Követelmények fejezetben bemutatom a megoldás elvárt követelményeit. A 4. Architektúra fejezetben röviden áttekintem az elkészült alkalmazás architektúráját. Az 5. Megvalósítás fejezetben az elvárt funkcionális követelmények alapján mutatom be az alkalmazásomat. A 6. Összefoglaló fejezetben pedig röviden összefoglalom tapasztalataimat és az alkalmazás fejlesztési lehetőségeit.

2 Felhasznált technológiák

2.1 Spring

A Spring egy népszerű Java keretrendszer, amely leegyszerűsíti a webes alkalmazások fejlesztését. Egyszerű annotációk használatával könnyen konfigurálhatóvá teszi az alkalmazást. Akár egy REpresentational State Transfer (REST) kéréseket fogadó végpontot szeretnénk, akár szolgáltatásokat beregisztrálni vagy függőség injektálást végezni, ezek mind megoldhatóak egy-két annotációval az osztályokon.

2.2 Gradle

A Gradle egy futtatás automatizáló eszköz Java, Android és Kotlin projektekhez. Segít a projekthez szükséges függőségek telepítésében, projekt konfigurációban, és rugalmasan akár általunk definiált taszkokkal is testre szabhatjuk alkalmazásunk futtatási folyamatait.

2.3 JPA

2.4 Tervezési minták

2.4.1 Domain-Driven Design

2.4.2 Repository

2.5 TypeScript

2.6 React

A React egy könyvtárt nyújt webes felhasználói felületekhez. Segítségével könnyen készíthetünk Single-page alkalmazásokat. Egy Single-page alkalmazás egy olyan webes alkalmazás, amely úgy interaktál a felhasználóval, hogy dinamikusan újraírja a weboldal tartalmát, szemben a klasszikus megoldással, melynél új oldalak betöltésére lenne szükség. Ezáltal gyorsabban képes reagálni a felhasználó kéréseire.

A React könyvtár JavaScript nyelven íródott komponenseket ad, melyeket újra használható, moduláris építőkockaként kezelhetünk webes alkalmazásunkban. Azért választottam ezt a technológiát, mivel nagyon népszerű és elterjedt, és szakmailag sokat fejlődhetnék megismerésével és elsajátításával.

2.7 Material UI

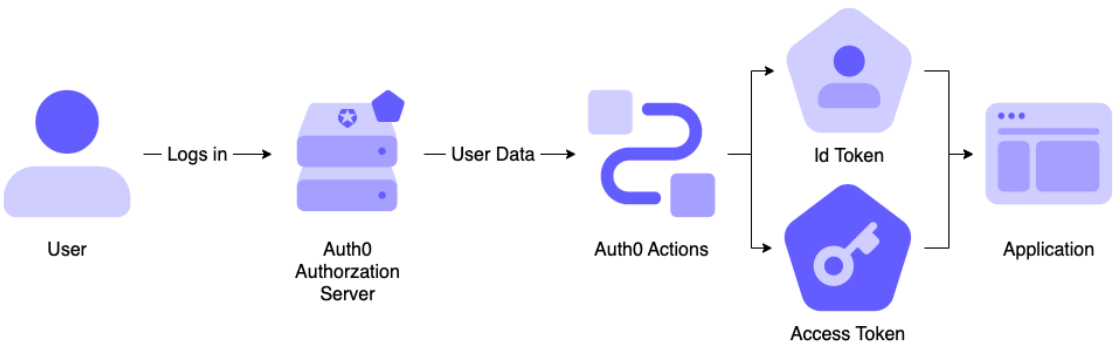
A Material UI egy UI fejlesztői eszköz gyűjtemény, mely egységes és könnyen kezelhető alkotóelemeket nyújt a webes alkalmazások fejlesztéshez. A Material Design stílusirányelvek miatt egyszerűbb az esztétikus weboldalakat létrehozni, illetve az elemek stílusa rugalmasan módosítható, ezzel teret adva az egyedi megoldásoknak. Mivel ez egy React technológiához tervezett UI könyvtár, így egyszerű volt az elemek használata. A Material UI komponensek nagyon hasonlítanak a sima React komponensekhez, emellett jól kidolgozottak.

2.8 Axios

Az Axios egy HTTP kliens, mely lehetővé teszi az aszinkron HTTP kéréseket. Támogatja a JavaScript Promise technológiáját, mely megkönnyíti az aszinkron kérések használatát az async és await kulcsszavakkal.

2.9 Auth0

Az Auth0 egy OpenID protokoll alapú autentikációt biztosít. Az OpenID egy autorizációs szerver segítségével azonosítja a felhasználót, és adataiknak lekérését. Meg szerettem volna ismerni egy biztonságos és megbízható felhasználó kezelési módot, melynél a jelszó és regisztráció kezelés delegált, és lehetővé tesz egyéb bejelentkezési módszereket, például Google vagy Facebook fiókokkal.



1. ábra Auth0 bejelentkezés

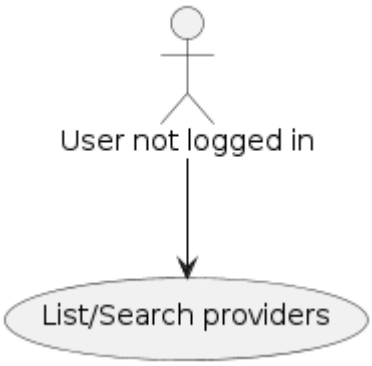
Az Auth0 esetén egy alkalmazás szervert kell létre hoznunk a webes felületen, amely kezeli a beérkezett autentikációs kéréseket. Itt lehetőség van a már regisztrált felhasználók kezelésére és aktivitásuk megtekintésére. Amikor egy felhasználó be szeretne jelentkezni (1. ábra), akkor a frontend elnavigál az Auth0 által nyújtott bejelentkező felületre. Itt a felhasználó azonosítja magát, és a megadott címre visszairányítja az Auth0. Ekkor a frontendünkön el tudjuk érni a felhasználó adatait (ID Token) és a használatra jogosító hozzáférési tokent (Access token).

3 Követelmények

Az alkalmazás felhasználóit három csoportra bontottam. Van lehetőség pár limitált funkcióra bejelentkezés nélkül is, így az első csoport a nem bejelentkezett felhasználók. Mivel a szolgáltatóknak külön felület kell, ahol felvihetik adataikat a szolgáltatásaikról, ezért ezt a két felhasználói típust vettem még fel: általános felhasználó és a szolgáltató felhasználó.

Eleinte teljesen elkülönülő szerepkörökként akartam létrehozni a vevő és szolgáltató felhasználókat, azonban ekkor a szolgáltató szerepű felhasználóknak nem lenne lehetősége időpontot foglalni más szolgáltatókhoz. Készíteniük kellene egy különálló profilt csak a foglalásra. A szolgáltatói és vevői profil közti váltogatás erősen negatívan hatna a felhasználói élményre. Végül úgy döntöttem, hogy a szolgáltatók is használhatnak minden funkciót, mint egy általános felhasználó, és emellett képesek egyéb extra funkciók igénybevételére is.

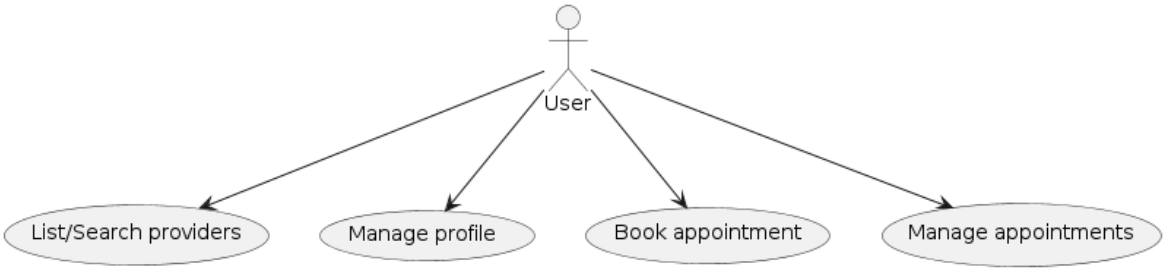
3.1 Böngészés



2. ábra Be nem jelentkezett felhasználó

Az egyetlen funkció, amely egy be nem jelentkezett felhasználónak elérhető, az a szolgáltatók megtekintése. Lehetőségük van kilistázni őket, megtekinteni adataikat. Ebben a listában tudnak név alapján keresni, illetve szűrni az eredményeket szolgáltatások alapján.

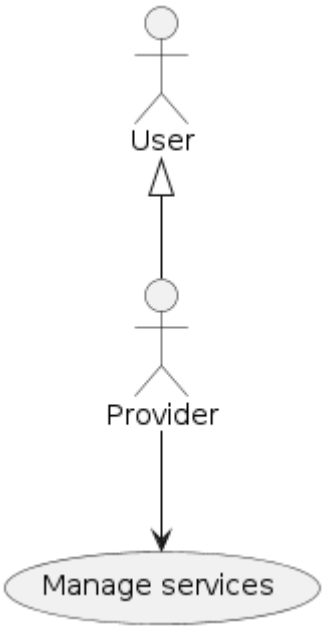
3.2 Általános felhasználó



3. ábra Általános felhasználó

Bejelentkezés után hozzáférhetünk az általános felhasználóknak elérhető funkciókhoz. Ez a szerep képes kilistázni, keresni a szolgáltatók között. Miután talált egy szimpatikus szolgáltatót, elindíthatja az időpont foglalás folyamatát. Ehhez ki kell választania a szolgáltatást és az időpontot is. Emellett képes megnézni a profilját, ezen módosításokat végezni, valamint menedzselni a már lefoglalt időpontjait.

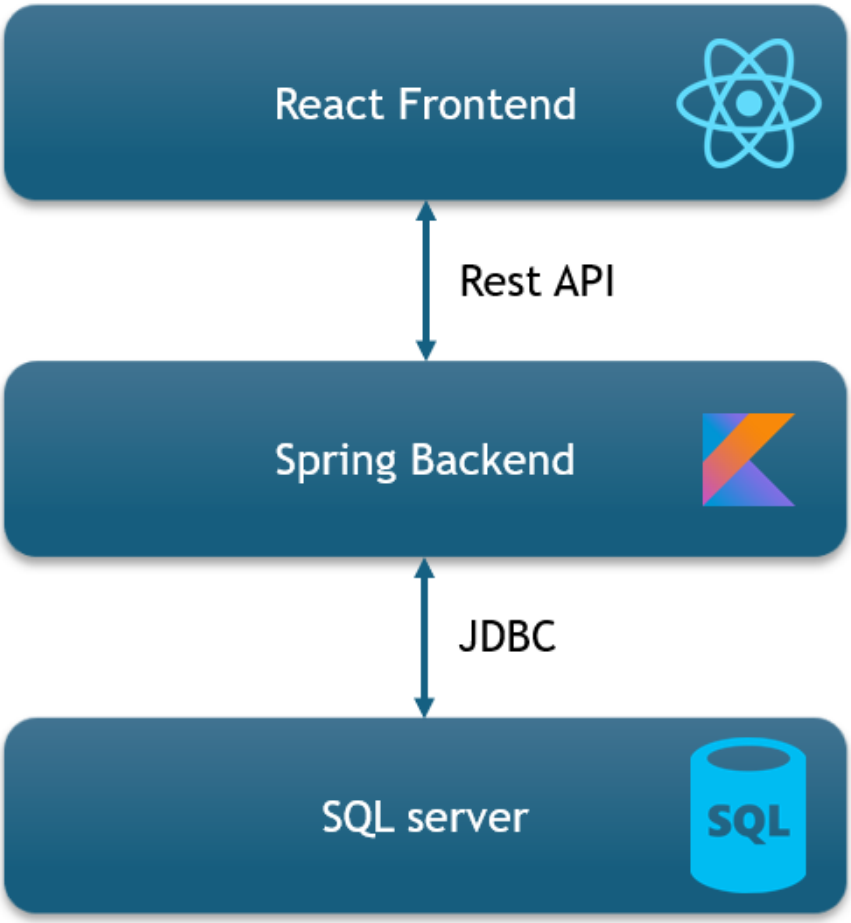
3.3 Szolgáltató felhasználó



4. ábra Szolgáltató felhasználó

A szolgáltatók számára elérhető az összes általános felhasználói funkció. Emellett képesek felvenni az általuk nyújtott szolgáltatásokat és ezeket menedzselni. Emellett meg tudják adni, hogy mikor érnek rá foglalásokat fogadni.

4 Architektúra



5. ábra Architektúra

Webalkalmazásom felépítése a megszokott három rétegű architektúrát követi. A 5. ábra tetején látható a React alapú frontend szerverem, mely biztosítja az oldal megjelenítését, a felhasználók autentikációját és a kérések delegálását a Spring boot backend szerverem felé. Ez a kommunikáció egy REST API interfészen keresztül megy. A backendre beérkező kérések átmennek az üzleti logikán, és a szükséges adatokat a JPA elkéri az SQL adatbázis szervertől.

4.1 Frontend architektúra

Router-el a navigáció

Auth0 kérések

Backend felé Axios

4.2 Backend architektúra

Rest kontrollerek

Spring service-ek

JPA repository-k

4.3 Adatbázis szerkezete

Eleinte a H2 relációs adatbázist használtam, mivel ez eleve bele volt integrálva a JPA konfigurációba. Ez egy könnyen használható memóriát használó adatbázis, azonban ezt főképp fejlesztő és tesztelő környezetekben használják. Ezért is ki akartam próbálni egy adatbázis konfigurációt, melyet az éles környezetekben használnak, így egy MS SQL szerver mellett döntöttem.

A JDBC kapcsolat paramétereit kellett konfigurálnom, illetve az adatbázis szervert, hogy fogadjon TCP/IP kapcsolatokat. Miután ezzel végeztem, néhány módosítást kellett végeznem a backend által definiált entitásokban és ismét gördülékenyen működött az összes JPA repository használata.

5 Megvalósítás

Mindenhova kép a végleges feature-ökről Egyelőre nem teszek, mert át tervezem alakítani a ui-t.

5.1 Profil

A profil megtekintéséhez először a bejelentkezett felhasználónak a jobb felül található a profil ikonra kell kattintania. A legördülő menüből kiválasztva a profil menüpontot kiválasztva tud navigálni saját profil oldalára.

5.2 Szolgáltatók listája

5.3 Időpont foglalás

insert picture here

Itt látha

Időpont foglalást a bejelentkezett felhasználó képes kezdeményezni. Nem csak a React frontendemben van korlátozva ez az út, de maga a gomb sem jelenik meg egy nem bejelentkezett felhasználó számára.

5.4 Foglalt időpontok kezelése

5.4.1 Általános felhasználó

5.4.2 Szolgáltató

5.5 Szolgáltatások kezelése

5.6 Foglalható időpontok kezelése

6 Összefoglaló

7 Irodalomjegyzék

[1] P. Koopman, „How to Write an Abstract,” október 1997. [Online]. Available: https://users.ece.cmu.edu/~koopman/essays/abstract.html. [Hozzáférés dátuma: 20 október 2015].

[2] W3C, „HTML, The Web’s Core Language,” [Online]. Available: http://www.w3.org/html/. [Hozzáférés dátuma: 20 október 2015].

[3] K. Nahtkasztlija, „Az idegen szavak toldalékolása,” június 2009. [Online]. Available: http://www.pcguru.hu/blog/kredenc/az-idegen-szavak-toldalekolasa/5062.

Függelék