FELADATKIÍRÁS

Személyfuvarozást támogató alkalmazás készítése

Napjainkban a rohamosan növekvő népesség és igényei nehéz helyzetbe kényszeríti a fővárosi közlekedést, mivel a kialakult strusktúra nehezen változtatható, új utak, parkolóhelyek létrehozása sok helyen nehezen kivitelezhető. Az átlag életszínvonal növekedésével a tömegközlekedés nehezen tud lépést tartani, így az emberek jelentős része járna autóval, ez pedig az utak nagyobb leterheltségéhez vezet. A probléma mérséklésére számos alternatíva jelent meg. Ezek közül a kényelmes de legtöbbször drága taxi szolgáltatások és a kevésbé komfortos de általában olcsóbb közösségi közlekedési opciók (Mol Limo stb.) ötvözése az alkalmazás célja, mely egy megfizethető és kényelmes alternatívát nyújtana a városi közlekedésre.

Egy startup alkalmazás sem indulhat el saját weboldal nélkül, így a feladat elsősorban egy webes felület, illetve egy azt támogató Web Api létrehozása. A backend .NET Core Rest Api formájában készül el, melyről bebizonyosodott, hogy a Single Page alkalmazások készítése során is tartja a színvonalat a versertársakhoz képest. A felhasználói felület React keretrendszer használatával jön létre, ennek előnye, hogy egyszerűen készíthető a webes felületből mobilos Applikáció, mely egy taxi alkalmazás esetében később előnyös lehet.

A diplomaterv célja, hogy a hallgató létrehozzon egy webes alkalmazást mely képes felhasználók kezelésével két féle szolgáltatás kiszolgálására: Az oldalon egy felhasználó foglalhat adott időben adott jármű típussal egy pontból a másikba utat foglalni, mint egy hagyományos taxi szolgáltatás, illetve előre meghatározott időre kocsit bérelni mint egy limuzin szolgáltatás. Az alkalmazás képes kell legyen a szolgáltatott adatok alapján az ár automatikus kiszámolására, a felhasználó felé a foglalás összeállítására majd annak e-mailben való elküldésére, illetve a felhasználó adatainak és foglalásainak kezelésére.

A hallgató feladatának a következőkre kell kiterjednie:

* Egy .Net Core alapú Web Api létrehozására mely képes a kliensek kiszolgálására
* Több típusú adatbázis kezelésére is (pl. PostgreSql, MSSQL)
* A backend képes kell legyen:
  + Automatikus Email küldésére
  + Felhasználók és adataik bizalmas kezelésére
  + Biztonságos fizetési technológiák támogatás
* Egy React keretrendszerben készült webes kliens létrehozása

**Tanszéki konzulens:** Albert István, mérnöktanár

Budapest, 2021. február 7.

Dr. Charaf Hassan

egyetemi tanár

tanszékvezető



Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Automatizálási és Alkalmazott Informatikai Tanszék

Szalkai Krisztián

Személyfuvarozást támogató alkalmazás készítése

Konzulens

Albert István

BUDAPEST, 2021

Tartalomjegyzék

[Összefoglaló 5](#_Toc71324734)

[Abstract 6](#_Toc71324735)

[1 Bevezetés 7](#_Toc71324736)

[1.1 A téma 7](#_Toc71324737)

[1.2 A választott platformok 7](#_Toc71324738)

[1.3 A dokumentum felépítése 7](#_Toc71324739)

[2 Technológiák 8](#_Toc71324740)

[2.1 .NET Core 8](#_Toc71324741)

[2.1.1 Entity Framework Core 8](#_Toc71324742)

[2.1.2 AspNetCore Identity 9](#_Toc71324743)

[2.1.3 Swashbuckle 9](#_Toc71324744)

[2.2 React 9](#_Toc71324745)

[2.2.1 React router 11](#_Toc71324746)

[2.2.2 Material UI 11](#_Toc71324747)

[2.2.3 Axios 12](#_Toc71324748)

[2.3 TypeScript 12](#_Toc71324749)

[3 Tervezés 14](#_Toc71324750)

[3.1 Funkcionális követelmények 14](#_Toc71324751)

[3.2 A weboldal működésével kapcsolatos elvárások 15](#_Toc71324752)

[3.3 Felülettervek 15](#_Toc71324753)

[4 Megvalósítás 17](#_Toc71324754)

[5 Összefoglaló 18](#_Toc71324755)

[6 Irodalomjegyzék 19](#_Toc71324756)

Hallgatói nyilatkozat

Alulírott **Szalkai Krisztián**, 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(k), 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, 2021. 05. 07.

...…………………………………………….

Szalkai Krisztián

Összefoglaló

Jelen dokumentumban bemutatom a megoldás során használt technológiákat, a rendszer megtervezésének lépéseit, feltárom a megvalósításom lépéseit, az aközben felmerülő problémákat, végül leírom az alkalmazás továbbfejlesztési lehetőségeit, a megvalósítás során szerzett tapasztalataimat. A rendszer egy web szerverből és egy webes felületből áll.

Abstract

In this document I will introduce the technologies I used, the steps of designing and implementing the system and show how I solved the problems that occurred while developing the system. Lastly I’ll present some ideas for further development and summarize what I’ve learned during the project. The system consists of a web server, and a web applicatio.

# Bevezetés

A következő fejezetben kifejtem miért ezt a témát választottam, illetve kitérek néhány technológiai/platform választás kérdésére.

## A téma

Napjainkban a rohamosan növekvő népesség és igényei nehéz helyzetbe kényszeríti a fővárosi közlekedést, mivel a kialakult strusktúra nehezen változtatható, új utak, parkolóhelyek létrehozása sok helyen nehezen kivitelezhető. Az átlag életszínvonal növekedésével a tömegközlekedés nehezen tud lépést tartani, így az emberek jelentős része járna autóval, ez pedig az utak nagyobb leterheltségéhez vezet. A probléma mérséklésére számos alternatíva jelent meg. Ezek közül a kényelmes de legtöbbször drága taxi szolgáltatások és a kevésbé komfortos de általában olcsóbb közösségi közlekedési opciók (Mol Limo stb.) ötvözése az alkalmazás célja, mely egy megfizethető és kényelmes alternatívát nyújtana a városi közlekedésre.

## A választott platformok

A téma kiválasztása után el kellett döntenem milyen platformon fogom megvalósítani a szolgáltatásokat. A döntésben figyelembe vettem a jelenlegi szakmai trendeket, a hasonló alkalmazások megvalósításait, illetve szerepet játszott a saját preferenciám, az általam ismert programozási nyelvekben szerzett tapasztalataim is.

## A dokumentum felépítése

A továbbiakban bemutatom a használt technológiákat, használatuk előnyeit. Azután felvázolom a tervezés folyamatát, a megvalósítandó konkrét feladatokat, a létrehozandó adatbázis felépítését. Végül rátérek a megvalósítás tényleges folyamatára, az elkészült program bemutatására.

# Technológiák

Az alábbiakban az általam választott technológiákat mutatom be általánosságban, illetve amennyiben nem egyértelmű a választás, megemlítem a választási lehetőségeket és a mérlegelési pontok felsorolásával alátámasztom a választásaimat.

## .NET Core

Mivel a relációs adatbázis megvalósítására a .Net fejlesztői platformot választottam, így a használt technológia a .Net Core, hiszen a sokak által használt és szeretett .Net Framework továbbfejlesztett utódja (Scott Hanselman, 2019).

A .Net Core egy nyílt forráskódú, platform független keretrendszer, mely ideális szerveralkalmazások elkészítésére. Az ASP.NET Core lehetővé teszi egy állapotmentes Web API készítését, ami egy Hyper Text Transfer Protocol (HTTP) alapú kommunikációra alapuló szolgáltatás, így több platformról is elérhető, legyen az webes, asztali vagy telefonos.

A backend megvalósításához meggondoltam még a Java használatát, mint a másik vezető backend platform a piacon. A két platform körül folyamatos a „vita” a fejlesztők között, eddig úgy tűnik, hogy mindkét nyelv nagyon hasonló tulajdonságokkal rendelkezik (Patel, 2020), így leginkább a személyes preferencia játszott szerepet a döntésben.

### Entity Framework Core

Az Entity Framework Core (EF Core) egy lecsupaszított, platformfüggetlen verziója az Entity Framework-nek, ami egy Microsoft által fejlesztett objektum relációs leképző (ORM) keretrendszer (Microsoft team, 2018). Segítségével egyszerű .NET-es objektumokon végezhetünk műveleteket C# nyelven, amit az EF Core változáskövetője átalakít az adatbázis által értelmezhető parancsokká és végrehajtatja őket. A Language Integrated Query (LINQ) szintakszissal együtt használva könnyen értelmezhető kódbázist készíthetünk az adatsémánk manipulálása céljából. Ezen felül többféle adatbázis rendszert is támogat, így a fejlesztő számára transzparens módon tud ugyanabból a C# kódból Oracle, MSSQL, SQLite, PostgreSQL vagy akár NoSQL által futtatható kódot generálni.

### AspNetCore Identity

Az ASP:NET Core Identity egy úgynevezett „tagsági” rendszer, mellyel bejelentkezési funkcionalitást adhatunk .Net Core alkalmazásunkhoz. Az Identity nem csak az adott alkalmazásban létrehozott felhasználói fiókokkal történő belépést támogatja, hanem külső szolgáltatóktól származó bejelentkeztetés is támogatható (Ilyen például a Facebook, Google, vagy Twitter) (Rick Anderson, 2019).

Ennek nem csak az az előnye, hogy felhasználóinkat azonosítani tudjuk- hitelesíteni-, a rendszer támogatja a felhasználók meghatalmazási szintjének ellenőrzését is. Az általam fejlesztett rendszer szempontjából például ilyen meghatalmazási csoportok lehetnek a felhasználók adminisztrátorok és a dolgozók.

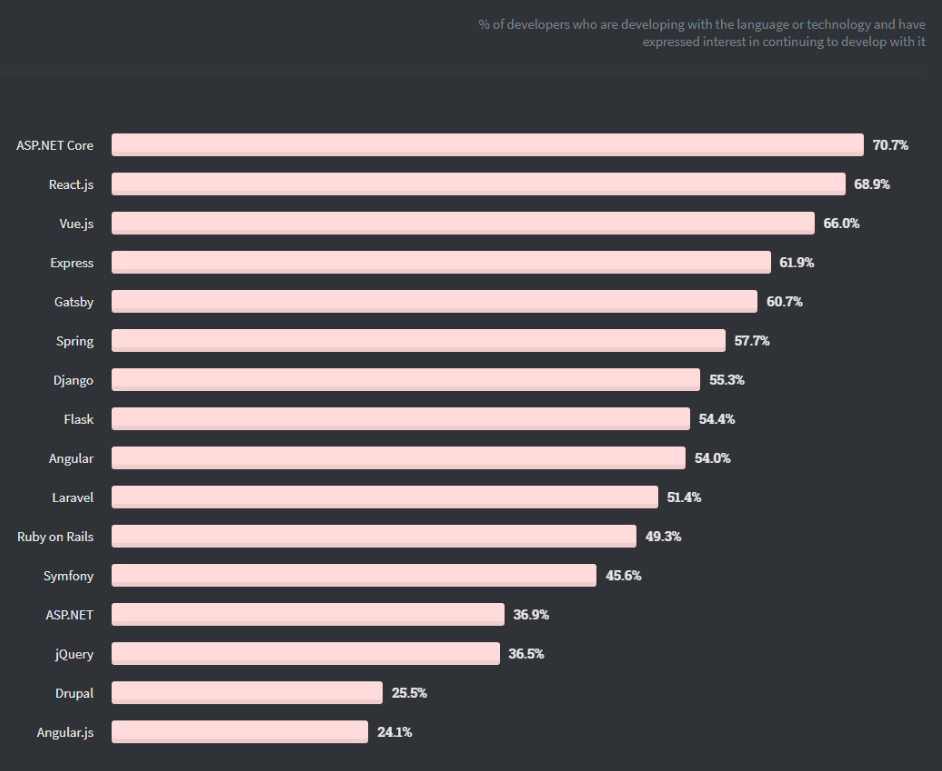
### Swashbuckle

A Swashbuckle egy olyan könyvtár mely segítségével a fejlesztő könnyebben dokumentálhatja, tesztelheti Representational State Transfer (REST) architektúrájú alkalmazását. Képes a megírt C# Apllication programming interface (API)-t felhasználva automatikusan egy olyan felületet generálni, melyről a kérdéses végpontok tesztelhetők, illetve a kód megfelelő kommentezésével az alkalmazás dokumentálását is nagyban elősegíti.

## React

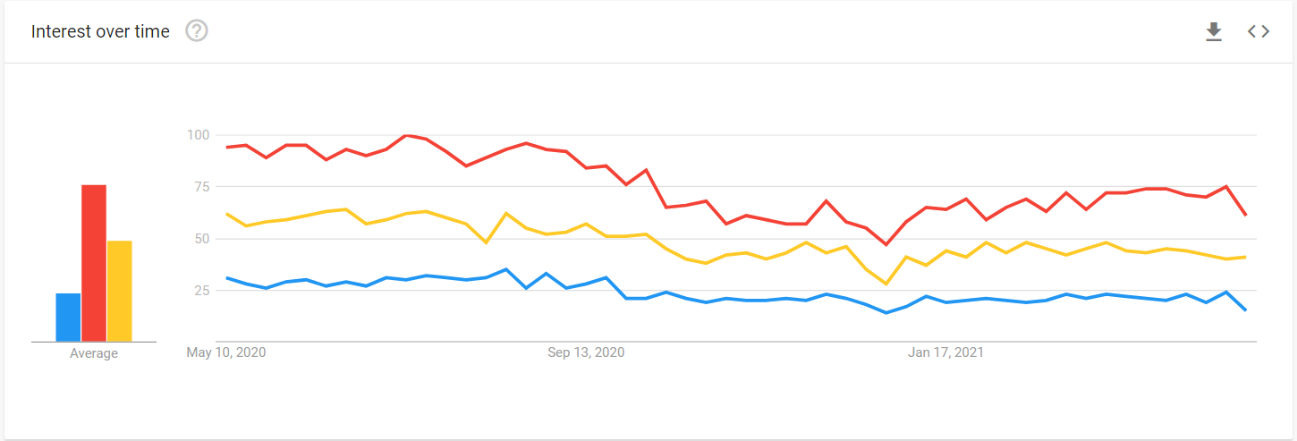
2021-ben elképzelhetetlen a web fejlesztés Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), és Javascript nélkül. Manapság a Javascript a frontend fejlesztés lelke, felvetül tehát a kérdés, hogy melyik User Interface (UI) könyvtárat válasszuk a számos lehetőség közül (Jigar Mistry, 2021). A választás nem könnyű, hiszen a piac és a vélemények folyamatosan változnak, azonban az utóbbi időben kialakuló statisztikák alapján három nagyobb keretrendszer emelkedett ki. Ezek a React, az Angular és a Vue.

A stackoverflow online lap kutatása szerint (Stack Overflow, 2021) a legkedveltebb web keretrendszer – az Modell-View-Controller (MVC) alapokon működő ASP.NET Core után – A React volt, azt pedig a Vue követte, az angulár pedig igencsak lemaradva jelent meg a listában.



2.1 A stackoverflow 2020-as statisztikája

A Google keresési statisztikái szerint (Google, 2021) viszont, ugyan szinten a React vezet, de az Angular követi, a Vue pedig utolsóként jelenik meg érdeklődést tekintve (keresések száma).



2.2 A google 12 hónapos keresési statisztikája 2021.05.07.-én

A statisztikák alapján tehát a React nyeri a versenyt, nézzük is meg, hogy miért lehet népszerűbb a többinél, illetve mi is a React.

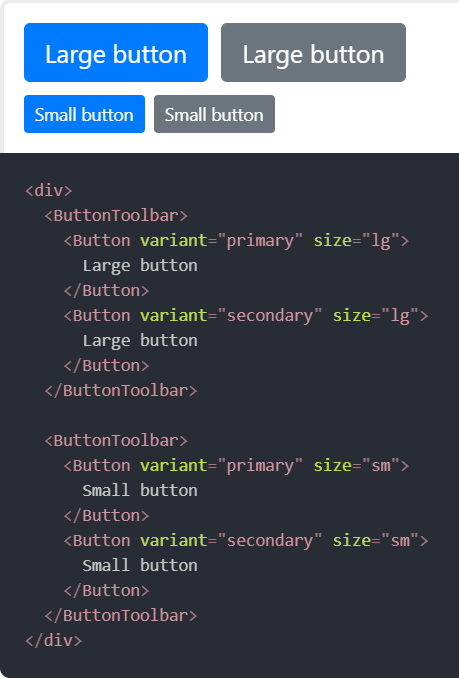
A React ahogy fentiekben említettem egy UI keretrendszer, JS könyvtár, melynek segítségével egy weblap felhasználói felületét készíthetjük el. Lényeges különbség például az Angular-hoz képest, hogy ez egy rendkívül könnyűsúlyú csomag, igazából nem is nevezhető keretrendszernek inkább csak egy könyvtár. Ez azt jelenti a gyakorlatban, hogy minden olyan feladathoz, ami nem kapcsolódik szorosan a megjelenéshez külső könyvtárakat kell használnunk, például a navigációhoz a React routert, a hálózati kommunikációhoz az Axios-t vagy hasonló segédkönyvtárakat. Nagy előnye legtöbb vetélytársával szemben, hogy virtuális Document Object Model-t (DOM) használ, mellyel sokkal jobb teljesítményt nyújt mint azok a megoldások amik valós DOM-ot használnak- Például az Angular. Tanulhatóság és a fejlesztés gördülékenysége szempontjából szintén érdemes olyan eszközt választani, amely nagy elterjedtséggel bír, mivel könnyebb hozzá segédanyagokat találni.

### React router

A legtöbb modern web alkalmazás úgynevezett Single page application (SPA) ami azt jelenti, hogy a megjelenítendő oldalak nem több weblapból tevődnek össze, hanem csak egy lapon változik a tartalom a megjelenítendő komponensek függvényében. Ezt a navigációt segíti elő a react router, anélkül navigálhatunk az oldalunkon, hogy ez az „egy oldal” amivel dolgozunk újratöltődne, fehér töltőképernyőt hagyva a két oldal megjelenítése között.

### Material UI

A React UI elemei komponensekből épülnek fel, ami azért hasznos, mert könnyen készíthetünk újra felhasználható HTML elemeket, akár megadható, felhasználásonként változtatható paraméterekkel melyek megváltoztatják a komponens viselkedését. A Material UI egy olyan könyvtár mely a bootstrapből jól ismert előre elkészített kinézeteket adja az alkalmazásunkhoz React komponensek formájában, így a komponensek tulajdonságainak kihasználásával élvezhetjük a bootstrap előnyeit, például paraméter segítségével adhatjuk meg a komponsens kinézetének tulajdonságait.



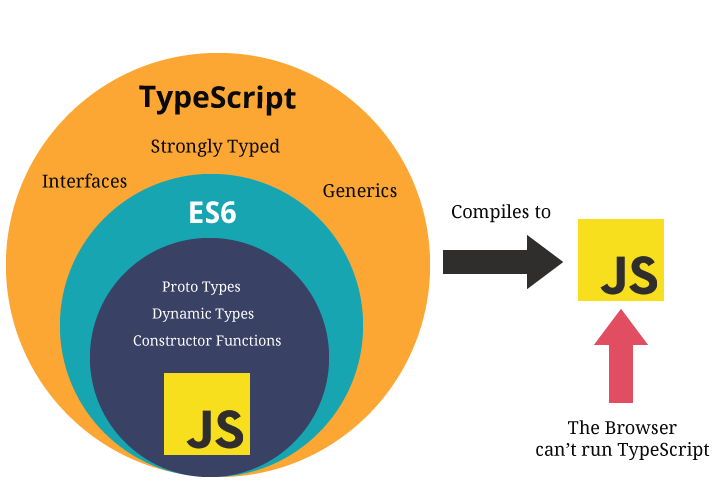
2.3 A Material UI nyomógomb komponensének egy példája

### Axios

Egy web alkalmazás természetesen nehezen képzelhető el valamiféle backend kommunikáció nélkül. Ezt segíti elő az Axios ami egy úgynevezett „promise-based” azaz ígéret alapú JS könyvtár HTTP kérések kezelésére, ami azt jelenti, hogy aszinkron kéréseket valósíthatunk meg, illetve a szerver válasza alapján tudunk elemeket betölteni, hibát kezelni (Flavio Copes, 2018). Az Axios előnyei közé tartozik még az natív fetch API-val szemben, hogy támogatja a régebbi böngészőket is, képes a kérések megszakítására, illetve beállítható rajta számos alapértelmezett beállítás, mint a válasz időtúllépésének korlátja, vagy olyan fejlécek melyek minden kéréssel elküldésre kerülnek – például egy authorizációs fejléc. Más hasonló könyvtárakkal szemben előnye, hogy beépített Cross-site request forgery (CSRF) védelemmel rendelkezik, illetve automatikus JavaScript Object Notation (JSON) adat átalakításra képes (Nnamandi, 2018).

## TypeScript

A TypeScript (TS) egy JavaScripten (JS) alapuló programozási nyelv, melyet nagyszabású webes felületek kialakításakor célszerű használni, úgy is gondolhatunk rá minta JavaScript továbbfejlesztése (Rawat, 2019). A TS kódunk egyenesen JS-re fordul, így egy minden platformra forduló biztonságos kódot kapunk. Leglényegesebb előnye a JS-el szemben, hogy erősen típusos és objektum orientált. Ez rengeteget segít, mikor a kódbázis nagyon nagy, elírások és hasonló hibák esetén a fordító hibát fog jelezni, így nem kevés időt megspórolhatunk magunknak fejlesztés során. Népszerűségének még egy oka, hogy mivel közvetlenül JS-re fordul, nyugodtan használhatunk kizárólag JS-ben írt könyvtárakat (Unknown, 2018).



2.4 TypeScript felépítése, kapcsolata a JavaSriptel

A typescript nyelv ezen kívül lehetővé teszi, hogy az új ECMAScript funkciókat használhassuk régebbi böngészőkön is, mivel a fordító az új funkciókból régi-a böngésző által támogatott- verziókra is képesek lefordulni.

Talán egyetlen hátránya – de legalábbis nehézsége – a JavaScriptel szemben, hogy pont a típusosság miatt nehezebb generikus megoldások alkalmazása, mint például a reflection, vagy az Enum értékek Stringként kezelése, ezekről a későbbi fejezetekben részletesebben is beszélni fogok.

# Tervezés

Ebben a fejezetben ismertetem a funkcionális követelményeket, a rendszer felépítését, a rendszer részei közti kapcsolatokat, illetve bemutatom az adatbázis struktúráját, az adatok elérésének, tárolásának módját.

## Funkcionális követelmények

Az elkészítendő alklamazás egy taxi szolgálat webolda, melyen a felhasználók tudnak regisztrálni, bejelentkezni, időpontokat foglalni a különböző szolgáltatásokhoz, illetve kezelni tudják a saját adataikat, eddigi foglalásaikat. Az alkalmazás ezen kívül lehetőséget nyújt az adminisztrátoroknak az összes leadott foglalás kezelésére, az adminisztrációs tevékenységek elvégzésére, a dolgozók pedig kezelhetik, áttekinthetik a hozzájuk rendelt munkákat (fuvarokat).

A taxiszolgálat által nyújtott szolgáltatások a következők:

* Pontból pontba történő fuvar meghatározott időpontban.
* Adott időtartamra autó bérlése folyamatos sofőrszolgálattal.
* „Normál” taxi működés, azonnali fuvar kérése.

A két előbbi szolgáltatás esetén meghatározható a foglaláshoz, hogy milyen típusú autóval, milyen extra szolgáltatásokkal (dohányzó/nem dohányzó, légkondi kisállatos, stb) kéri a felhasználó.

A weboldalon a felhasználó a következő funkciókat kell elérje:

* Regisztráció
* Belépés
* Foglalás a háromféle szolgáltatásra
* Saját adatok megváltoztatása, törlése
* Előző foglalások megtekintése
* Foglalás lemondása (Adott határidőn belül)
* Kijelentkezés
* Online fizetés

Adminisztrátori belépéssel:

* Foglalások megtekintése
* Foglalások dolgozókhoz rendelése
* Felhasználók és foglalásaik megtekintése

Dolgozói belépéssel:

* Dolgozóhoz rendelt foglalások megtekintése, kezelése.

## A weboldal működésével kapcsolatos elvárások

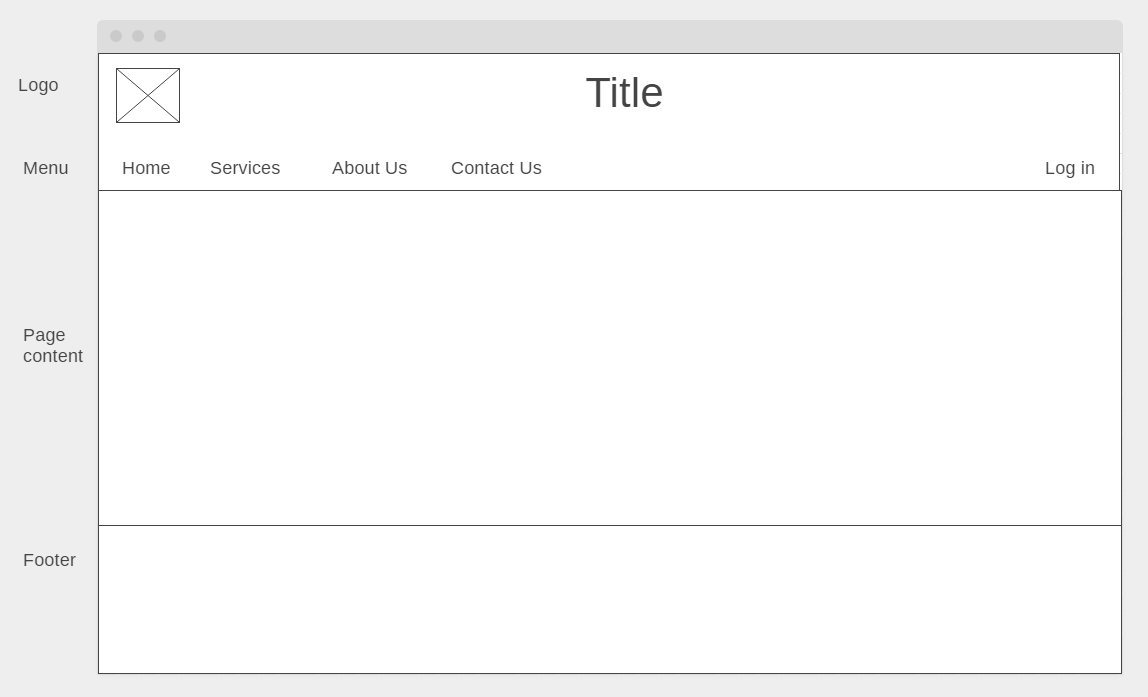
A weboldalnak minden személyes adatot biztonságosan kell tárolnia, az adatokhoz csak a megfelelő jogosultságú felhasználók férhetnek hozzá. A személyes adatok, illetve a felhasználói profil a felhasználó kérésére bármikor törölhető kell legyen.

Foglalás során a rendszernek automatikusan kell elvégezni a következőket:

* Ár számítása a preferenciák, extra szolgáltatások, szolgáltatás típusa és az esetleges felárak ( pl: Dugódíj) alapján.
* Átirányítás biztonságos fizetési lehetőségekhez (pl: Stripe, PayPal, Barion). Fizetés esetén az esetleges problémák kezelése (Sikertelen fizetés, helytelen összeg, hálózati probléma).
* A foglaláshoz egyedi azonosító generálása és megerősítő email küldése a felhasználónak.
* Határidőn belül történő foglalás lemondása esetén az összeg visszautalása a felhasználónak.

## Felülettervek

Az alkalmazás tervezését a követelményekből kiindulva a felülettervek elkészítésével kezdtem. A felülettervek készítése segíthet a megrendelővel lefektetett követelményekben fennmaradó esetleges félreértések felderítésében, illetve a webszerver és az adatbázis megalkotásánál is hasznos támpontot adnak. A tervekhez létrehozásához a WireFrame online tervező eszközt használtam.



3.1 A webes felület általános kinézete

Az alkalmazás alapvető felépítése a 3.1 A webes felület általános kinézete ábrán látható. A lap tetején mindig egy Header látszik, melyen megjelnik a logó a szolgáltatás neve és a navigációs menü. A lap alján egy Footer található, melyen helyet kaphatnak a kapcsolatfelvételi információk, felhasználási feltételek, oldaltérkép, stb. A kettő között helyezkednek el a különböző oldalak tartalmai. Látszik, hogy a tervezés ezen szakaszában még nem koncentrálunk a konkrét működésre, tényleges felületi elemekre – például nem jelenik meg a navigációs menü különbsége a különböző felhasználó típusok között.

# Megvalósítás

# Összefoglaló

# Irodalomjegyzék

Flavio Copes. (2018. April 11). *Medium*. Forrás: https://medium.com/free-code-camp/simple-http-requests-in-javascript-using-axios-272e1ac4a916

Google. (2021). *Google Trends*. Forrás: https://trends.google.com/trends/explore?cat=31&q=Vue,React,Angular

Jigar Mistry. (2021. March 18). *Monocubed*. Forrás: https://www.monocubed.com/10-most-popular-web-frameworks/

Maki, M. (2017. December 4). *Medium*. Forrás: https://medium.com/@marcellamaki/a-brief-overview-of-react-router-and-client-side-routing-70eb420e8cde

Microsoft team. (2018. 06 19). *Microsoft*. Forrás: https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server?toc=%2Faspnet%2Fcore%2Ftoc.json&bc=%2Faspnet%2Fcore%2Fbreadcrumb%2Ftoc.json&view=aspnetcore-3.1

Nnamandi, C. (2018. November 28). *Medium*. Forrás: https://blog.bitsrc.io/comparing-http-request-libraries-for-2019-7bedb1089c83

Patel, T. (2020. August 31). *concetto labs*. Forrás: https://www.concettolabs.com/blog/java-vs-net-which-is-better/

Rawat, D. (2019. January 2). *DEV.TO*. Forrás: https://dev.to/singhdigamber/what-is-typescript-jig

Rick Anderson. (2019. 07 12). *Microsoft*. Forrás: https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-3.1&tabs=visual-studio

Scott Hanselman. (2019. September 23). *Microsoft*. Forrás: https://dotnet.microsoft.com/learn/dotnet/hello-world-tutorial/intro

Stack Overflow. (2021). *Stack Overflow*. Forrás: https://insights.stackoverflow.com/survey/2020#technology-most-loved-dreaded-and-wanted-web-frameworks-loved2

Unknown. (2018. April 16). *Zero to hero in TypeScript*. Forrás: http://zerotoherointypescript.blogspot.com/2018/04/typescript-introduction.html