Tartalom

[1. - A vizsgaremek - A frontend bemutatása 1](#_Toc134053816)

[1.1 – Technológiai háttér 1](#_Toc134053817)

[1.1.1 – Szoftverarchitektúra 1](#_Toc134053818)

[1.1.2 – Fejlesztői környezet 1](#_Toc134053819)

[1.1.3 – Verziókezelés 2](#_Toc134053820)

[1.1.4 – Node.js és rá épülő tartalmak 2](#_Toc134053821)

[1.1.5 – Vue.js 3](#_Toc134053822)

[1.1.6 – Munkafolyamat lépéseinek vezetése 3](#_Toc134053823)

[1.2 – Használati útmutató 4](#_Toc134053824)

[1.2.1 – Alapkoncepció 4](#_Toc134053825)

[1.2.2 – Üzleti logika terve 5](#_Toc134053826)

[1.2.1 – Design-tervek 5](#_Toc134053827)

[1.2.1 – Az alkalmazás leírása 6](#_Toc134053828)

[2. - Tesztelés 18](#_Toc134053829)

[3. - Források, hivatkozások 21](#_Toc134053830)

# 1. - A vizsgaremek - A frontend bemutatása

Az alkalmazás frontendje a Vue.js JavaScript könyvtár illetve a Bootstrap CSS keretrendszer segítségével készült, főleg asztali gépen való használatra, de reszponzív módon lett kialakítva, hogy bármilyen interneteléréssel rendelkező készüléken használható legyen (különböző méretű laptopok, tabletek illetve mobiltelefonok.

A frontend megtervezését, kialakítását és fejlesztői tesztelését Fazekas Réka végezte.

## 1.1 – Technológiai háttér

A frontend oldali alkalmazás kialakításához változatos technológiákat használtam fel, a szoftverfejlesztők aktuális eszköztárából válogatva.

### 1.1.1 – Szoftverarchitektúra

Alkalmazásunk a két jellemző szoftverfelépítést ötvözi, a „vékony szerver, vastag kliens” (thin server, thick client) illetve a „vastag szerver, vékony kliens” architektúra elemei is megtalálhatóak benne. A szerver felel az adatbázissal való kapcsolattartásért, az adatok lekérdezéséért és mentéséért, az adatok feldolgozását, ellenőrzését, megjelenítését és a felhasználó interakciókat pedig a kliens oldali alkalmazás végzi. Ez a felépítés gyakori a modern webes alkalmazások tervezésénél.

Ez a típusú szoftverfelépítés a többihez képest összetettebb tervezést és fejlesztést igényelt, főleg az adatok biztonságának garantálása miatt, de a modern szoftverfejlesztésnek megfelelően fontos szempont volt a felhasználói élmény (UX/UI) követeleményeinek figyelembe vétele is.

### 1.1.2 – Fejlesztői környezet

A frontend kialakítása a Visual Studio Code (VS Code) nevű, nyílt forráskódú fejlesztői környezetben (IDE) történt. Ezt a programot a Microsoft fejlesztette ki, és jelenleg az egyik legelterjedtebb IDE a webfejlesztők körében, mivel ingyenesen elérhető, jó támogatottságú és könnyedén testreszabható a fejlesztői munkát nagyban segítő bővítményekkel. A frontend fejlesztése alatt én a Tabnine AI Autocomplete, a Vue Language Features (Volar), a Git Graph, GitLens, Pretty Formatter illetve a SonarLint ingyenesen elérhető bővítményeket használtam.

### 1.1.3 – Verziókezelés

A fejlesztés különböző állapotainak, változásainak lekövetésre git verziókezelő rendszert használtunk, a Visual Studio Code és a GitHub, valamint a GitHub és a GitLab felületek intgerációját kihasználva.

Mind a VS Code, mind a Gitlab felületére be tudunk jelentkezni a GitHub accountunkal, majd egy gyors hitelesítést követően máris rendelkezésünkre állnak a GitHubon elérhető tárolóink, valamint egy egyszerűen kezelhető CI/CD (Continous Integration/Continous Deployment) platform, amelyen többek közt könnyen megvalósítható az alkalmazás kódszintű automata tesztelése.

### 1.1.4 – Node.js és rá épülő tartalmak

A Vue.js használatához szükség volt a megfelelő környezet kialakítására, ami magába foglalta a Node.js, az NPM, és különböző NPM-en keresztül telepíthető package-ek telepítésére is.

Node.js

Nyílt forráskódú, JavaScript nyelven alapuló programozási környezet, szerveroldali megoldásokra. Aszinkron programozási modelleket használ ki, és a Google Chrome böngészőben támogatott V8 JavaScript motor működteti. Nagy előnye, hogy a fejlesztők készíthetnek és használhatnak saját modulokat, amelyek lehetővé teszik egyes kódok újrafelhasználását így a kódduplikáció elkerülését.

NPM

Az NPM (Node Package Manager) a Node.js által használt csomag- illetve modulkezelő rendszer. Ez teszi lehetővé, hogy a fejlesztők egyszerűen és könnyen telepítsék, frissítsék és kezeljék a Node.js-el kapcsolatos függőségeket (dependency-k).

Jelen vizsgamunkában a Bootstrap legfrisebb elérhető verziója, a Bootstrap ikongyűjteménye, az axios illetve az sha256 node package-ek lettek telepítve.

Bootstrap

A bootsrtap egy nyílt forráskódú frontend keretrendszer, az egyik legszélesebb körben webfejlesztésben használt CSS illetve JS framework. Előre elkészített, könnyedén testreszabható elemeket tartalmaz, amelynek segítségével egy kezdő fejlesztő is kiemelkedően gyorsan és hatékonyan tud reszponzív weboldalakat és alkalmazásokat fejleszteni, amelyek így mobilra, tabletre, laptopra, asztali gépre egyaránt optimalizálva vannak.

Bár létezik a Vue.js-hez optimalizált verzió is, jelen projektben az NPM-en keresztül automatikusan a bootstrap legfrisebb elérhető stabil verzióját vettük használatba, hogy minél tovább támogatott legyen az oldalhoz elkészített design.

Axios

Olyan JS könyvtár, ami könnyen kezelhetővé teszi a szerverrel való kommunikációt, azaz a HTTP kérések (get, post, put, delete) és válaszok kezelést, aszinkron módon is.

SHA-256

Alapvetően egy hash algoritmus, amelyet a Secure Hash Algorithm részeként fejlesztettek ki nemzetbiztonsági célokra. Ebben a projektben a felhasználók jelszavának titkosítására használtuk fel, backend és frontend oldalon is, kétszeres kódolással, ami gyakorlatilag felfejthetetlen titkosítást biztosít így jelentősen csökkentve a adatvédelmi kockázatot.

A frontend alapját adó Vue.js-t is az NPM segítségével telepítettem és konfiguráltam.

### 1.1.5 – Vue.js

A Vue.js egy JavaScripten alapuló, könnyen megtanulható és használható keretrendszer jellegű könyvtár, ami 2014-ben jelent meg először. Alapvetően egyszerű, de látványos, felhasználóbarát alkalmazások fejlesztésére szolgál.

A Vue.js egyik előnye a virtuális DOM-ot (Document Object Model) használ, azaz a változó frissülésével egyidőben az igazi DOM is módosul, ami sokkal gyorsabb teljesítményt jelent pl egy jQuery-t használó alkalmazással szemben.

Architektúrája az MVC-hez (Model-View-Controller) hasonló, az adatok, a metódusok és a html kód könnyen szétválasztható, áttekinthető. Az adatok még teljesebb szeparálására a Vuex állapotkezelő könyvtár ad módot, amelynek három fő része az állapot (State), a műveletek (Actions) illetve a mutációk (Mutations), amelyek mind különböző állapotokat meghatározó adatok, egy helyre gyűjtve.

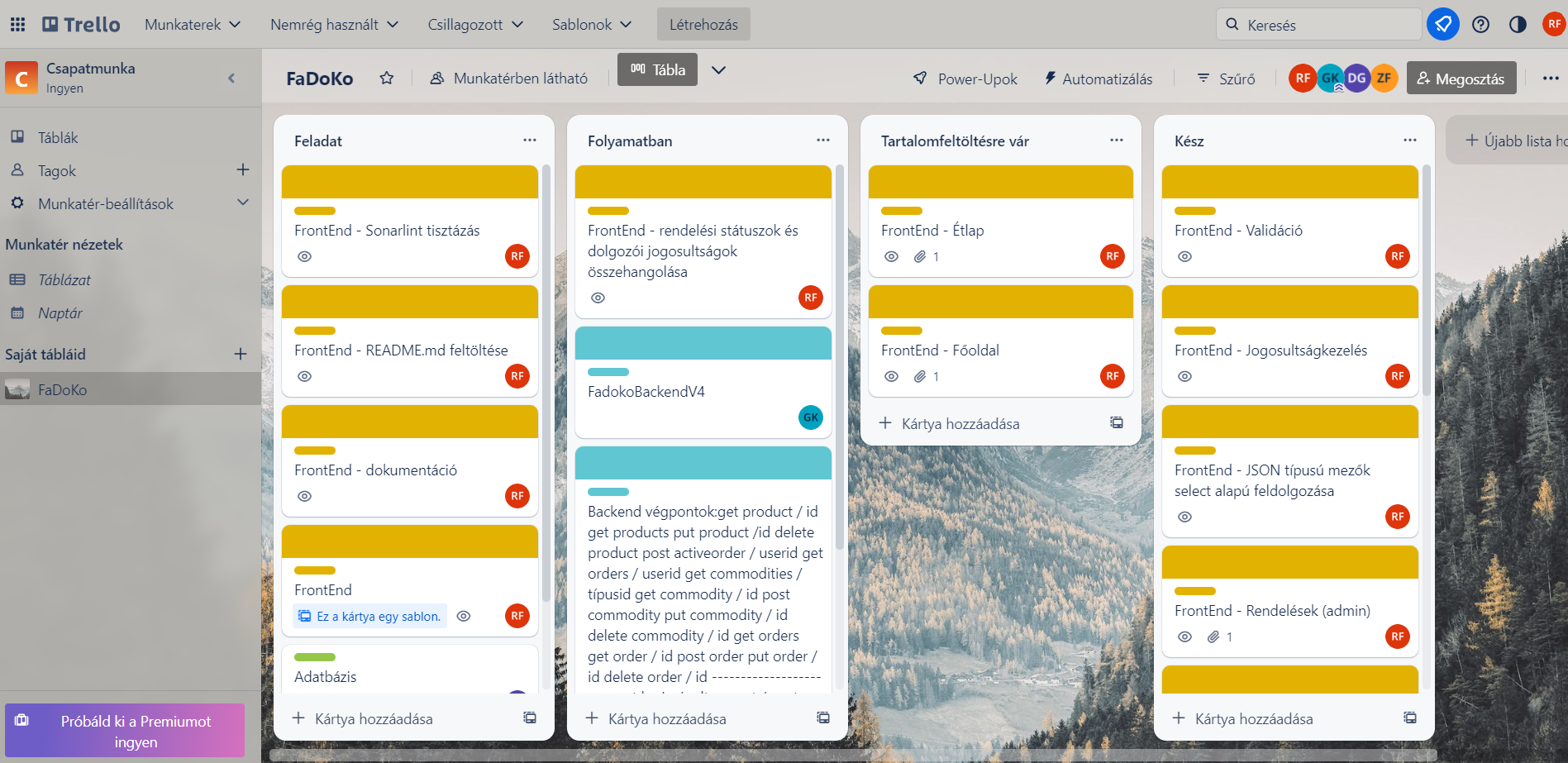
A Vue.js alapelemei a komponensek. Ezek olyan kódrészleteket tartalmaznak, amelyek az oldalon ismétlődnek, és a kiszervezésnek köszönhetően könnyedén újrafelhasználhatóak, így rövidítve és egyszerűsítve a kód egészét.

### 1.1.6 – Munkafolyamat lépéseinek vezetése

Mint a legtöbb modern fejlesztői feladat esetében, a feladatok és az elvégzett munka nyilvántartásához webes feladatkezelő rendszert használtunk. A választás a Trello nevű feladatkezelőre esett, amely ingyenesen is elérhető, és melyben az egy projekten dolgozó felhasználók megosztott táblákra helyezhetik el feladataikat, kártyák segítségével.

Az egyes kártyák címkézhetőek, kommentelhetőek, és fájlokat is tudunk csatolni hozzájuk, a frontend esetében például az egyes oldalakhoz illetve részletekhez kapcsolódó design vázlatokat csatoltam a kártyákhoz.

Módszertanilag az agilis megközelítéshez álltunk a legközelebb, amelyből a változásokra és az új igényekre történő rugalmas reagálást, a szinte állandó kommunikációt és feedbacket és a kockázatok korai azonosítását és kezelését is sikerült megvalósítani. A feladatkártyák megjelenítése a Kanban rendszert idézte.



1. ábra – A vizsgaremek projektjéhez tartozó Trello tábla, a fejlesztés egy adott szakaszában

## 1.2 – Használati útmutató

### 1.2.1 – Alapkoncepció

Az alkalmazás alapötlete a dolgozat korábbi szakaszában is tárgyaltan egy olyan felület megvalósítása, amelynek egy vendéglátóipari egység, konkrétan egy pizzéria vendégei, valamint dolgozói lesznek a felhasználói.

A weboldal megjelenését mobil, tablet, és laptop/asztali számítógép megjelenítésének megfelelően is optimalizáltam. Ennek érdekében a design tervek eleve úgy készültek, hogy Bootstrap 5.3-as keretrendszer segítségével könnyen megvalósíthatóak legyenek.

A frontend logikáját Vue.js JS könyvtár alapon alakítottam ki, így az felel a étlap tartalmának beolvasásáért, a bejelentkezésért, az oldalak közti váltásért és még sokminden másért is.

### 1.2.2 – Üzleti logika terve

Hosszasabb tervezést követően végül arra jutottunk, hogy az alkalmazást 2 alap felhasználási módra készítjük fel: egyrészről egy weboldal, ahol a vásárlók/vendégek tájékozódni tudnak az étterem alapinformációval kapcsolatban, mint az elhelyezkedés, az elérhetőségek, illetve az étlap.

A másik felhasználási mód az étlap szerkesztését, a rendelések megtekintését és szerkesztését illetve a felhasználók megtekintését és szerkesztését teszi lehetővé jogosultságfüggően.

A képen diagram látható

Automatikusan generált leírás

2. ábra – Üzleti terv kialakulása. A vizsgaremek végül a 2. üzleti tervet követve készült el

### 1.2.1 – Design-tervek

A tervezett alkamazás összetettsége nem tette indokolttá grafikus tervezőprogramok használatát, így a szabadkezes vázlatok készültek, figyelembe véve hogy a terveket a Bootstrap 5.3-as verziójában található elemekkel könnyen fel lehessen építeni.

A képen diagram látható

Automatikusan generált leírásA képen diagram látható

Automatikusan generált leírás

3. ábra – Szabadkezes design tervek, Bootstrap elemekből felépíthető komponensekkel

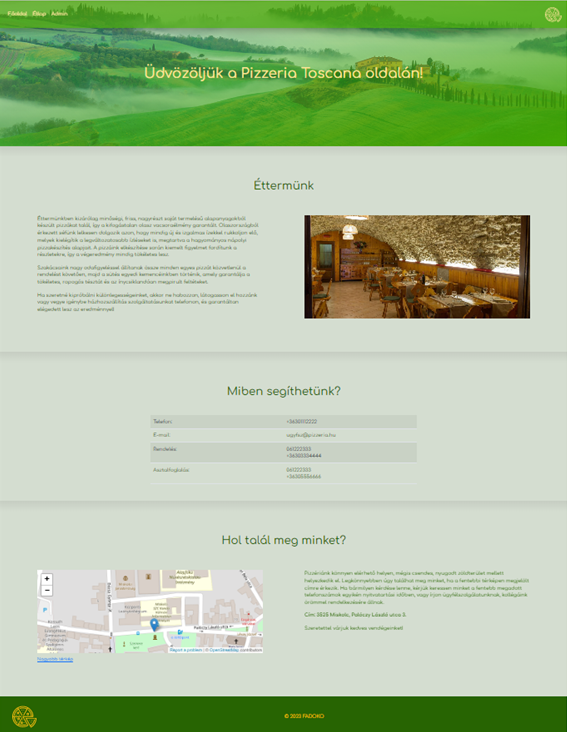
### 1.2.1 – Az alkalmazás leírása

BEJELENTKEZÉS NÉLKÜL ELÉRHETŐ OLDALAK

Az alkalmazás elsődleges elérési módja. Ennek megtekintéséhez nincs szükség bejelentkezésre, bárki szabadon elérheti és tájékozódhat róla az elérési út ismeretében. Elsődleges célja, hogy a felhasználók megtalálják az éttermet illetve ki tudják választani a megrendelni kívánt termékeket az oldalról. A főoldal és az étlap érhető el alapvetően ebben a formában, valamint az admin felület bejelentkezési ablaka.

Főoldal

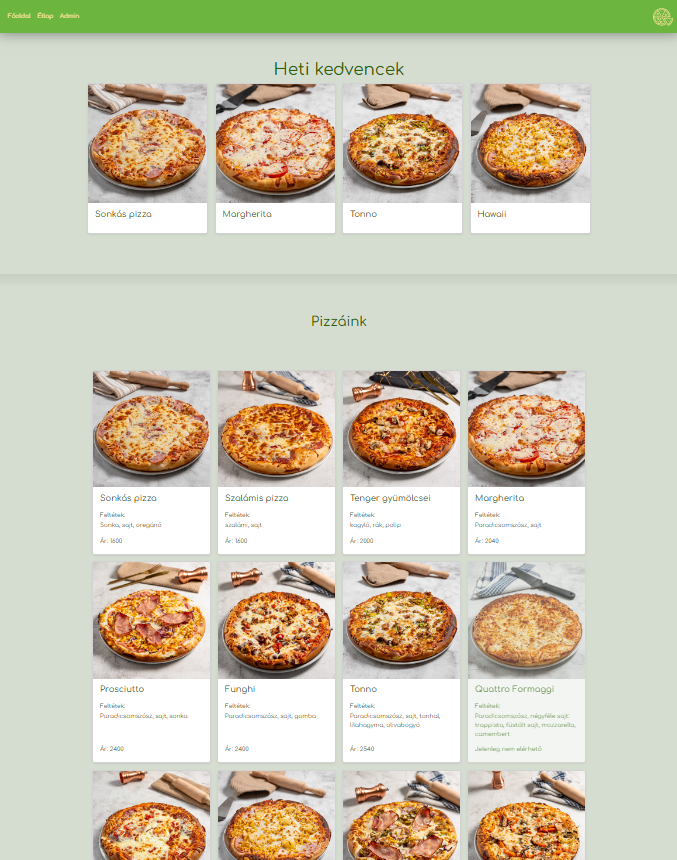
Erre az oldalra érkezik a felhasználó az alkalmazás indításakor illetve a kijelentkezést követően. Innen tud tájékozódni a hely jellegzetességeiről, elérhetőségeiről, elhelyezkedéséről. Az oldalt marketingszövegekkel töltöttem fel.



Az oldalt minden aloldalon ismétlődő fejléc és lábléc foglalja keretbe. Utóbbiban az oldal szerkesztőiről és az alkalmazás fejlesztésének idejéről helyeztem el információt, a előbbi pedig a mindenhonnan könnyen elérhető navigációt segíti. A felhasználó-interakciót egy egyszerű hover funkció könnyíti, a kurzor pozíciójának megfelelően a fejléc változtatja az áttetszőségét: lejjebb tekerve láttatni engedi a mögé kerülő tartalmat, amikor a kurzor a navigációs linkek közelébe kerül, akkor viszont elveszíti átlátszóságát, a menüpontok olvashatóságát növelve. Szerkezetileg a fejléc és a lábléc bootsrtap elemek, az oldal blokkjai pedig Vue komponensek, ez lehetővé teszi a gyors variálást, ha szeretnénk egy blokkot kivenni vagy hozzátenni, vagy csak a sorrendjükön módosítani.

Étlap

Bejelentkezés nélkül az étlap funkciója a pizzák összetételéről, áráról illetve elérhetőségéről történő tájékoztatás.



A nem elérhető pizzákat enyhe áttetszőséggel jelöltem, valamint szöveges visszajelzést is nyújtok a felhasználónak, hogy a tárgyban forgó termék nem elérhető aktuálisan.

Az Étlap oldal 2 fő blokkból áll, a heti kedvenceket bemutató blokkból, melyek a példában véletlenszerűen kerültek kiválasztásra, de tartalma könnyen testreszabható a frontend felületen a pizzák betöltésekor. A másik blokk a fő tartalom, a pizzák felsorolása.

Mind a kedvenc pizzák, mind az étlapot felépítő pizzák megjelenítéséért egy-egy Vue komponens elem felel, dinamikusan feltöltve adatokkal. Az egyes komponesekhez a bootstrap card elemét használtam, a reszponzív oldalmegjelenítést pedig a grid rendszer tette lehetővé, amelyben oldalszélességtől függően jeleik meg 1, 2, 3 vagy 4 komponens egy sorban.

Admin (bejelentkezés)

Ez az oldal már azoknak a felhasználóknak van fenntartva, akik rendelkeznek az oldalhoz belépési azonosítóval, ami esetünkben felhasználónév és jelszó.

A képen szöveg látható

Automatikusan generált leírás

A bejelentkezés oldal tablet nézetben, színvilágban testreszabott Bootstrap hamburgermenüvel

Szerkezet szempontjából a bejelentkezés funkcióját, a kijelentkezést és az felhasználók kezelését ugyanaz a felület végzi, aminek a tartalmát dinamikusan töltöm be, jogosultságfüggően.

A bejelentkezéssel kapcsolatos hibaüzeneteket JS alert üzenetek formájában hozza a felhasználó tudomására az oldal, amely tartalmaz a hiba azonosítására alkalmas fejlesztői üzenetet és felhasználóbarát tájékoztatást is (nem létező felhasználó, hibás felhasználónév vagy jelszó).

A képen szöveg látható

Automatikusan generált leírásA képen szöveg látható

Automatikusan generált leírás

A mezők validálása Vue formvalidáló beépített funkciójával történik. A bejelentkezés oldalon csak azt ellenőrzi az alkalmazás, hogy a felhasználó mindkét mezőt kitöltötte-e.

Ha nem, azt a beviteli mezők alatt, pirossal kijelölve, szöveges formában jelenítettem meg:

A képen szöveg látható

Automatikusan generált leírás

A helyes adatok megadását követően a sikeres bejelentkezésről is kap feedbacket a user, hogy milyen felhasználóval lépett be.

A képen szöveg látható

Automatikusan generált leírás

Az üzenet leokézását követően átirányítom a felhasználót a főoldalra, ahonnan kiválaszthatja hogy melyik aloldalt szeretné megtekinteni illetve használatba venni a rendelkezésére állók közül.

A bejelentkezés folyamata a felhasználónév és az sha256-os kódolással titkosított jelszó ellenőrzésével történik, sikeres esetben beállításra kerül a felhasználónév, a felhasználóazonotó és a felhasználó jogosultsági szintje a Vuex változók használatával.

A jogosultságszintek a következőképpen változtatják az oldalak funkcióit illetve működését:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 4, 5, 6-os szintű jogosultság | 7-es szintű jogosultság | 8-as szintű jogosultság | 9-es szintű jogosultság |
| Étlap | megtekintés | megtekintés | szerkesztés | szerkesztés |
| Rendelések | megtekintés | szerkesztés | szerkesztés |
| Admin | saját adatok megtekintése + kijelentkezés | megtekintés + kijelentkezés | megtekintés + kijelentkezés | szerkesztés + kijelentkezés |

1. számú táblázat – a dolgozói jogosultságok és a velük összefüggő elérhető funkciók

BEJELENTKEZÉS UTÁN ELÉRHETŐ OLDALAK, FUNKCIÓK

Az oldalakról általánosan elmondható, hogy mindegyik megjelenése függ a bejelentkezési állapottól és a jogosultsági szinttől. Amennyiben a bejelentkezési vizsgálat hibát jelez, a bejelentkezéssel elérhető oldalak (Rendelések, Pizzák hozzáadása, Pizzák módosítása) ha felhasználó épp rajtuk tartózkodik mikor véget ér a munkamenet, a következő üzenetet jelenítik meg egységesen:

A képen szöveg látható

Automatikusan generált leírás

A jogosultságoktól függően olyan oldalelemek jelennek meg amik alacsonyabb szinttel nem elérhetőek, illetve ha olyan műveletet próbál végezni a felhasználó akkor JS alert üzenetben kap visszajelzést a jogosultság hiányáról.

A képen szöveg látható

Automatikusan generált leírás

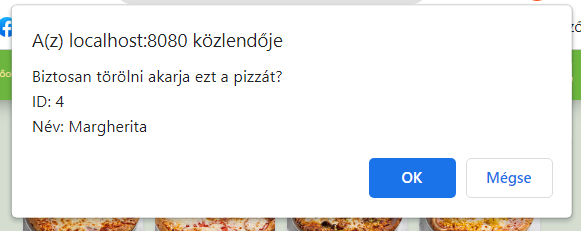
Étlap

Az Étlap oldal az 1.es táblázatnak megfelelően az alacsonyabb jogosultságú felhasználóknak nem mutat változást a bejelentkezés nélkül elérhető oldalképhez képest. 8-as jogosultsági szinttől lehetősége válik a felhasználónak a hozzáadási, módosítási és törlési funkciókra:

A képen Webhely látható

Automatikusan generált leírás

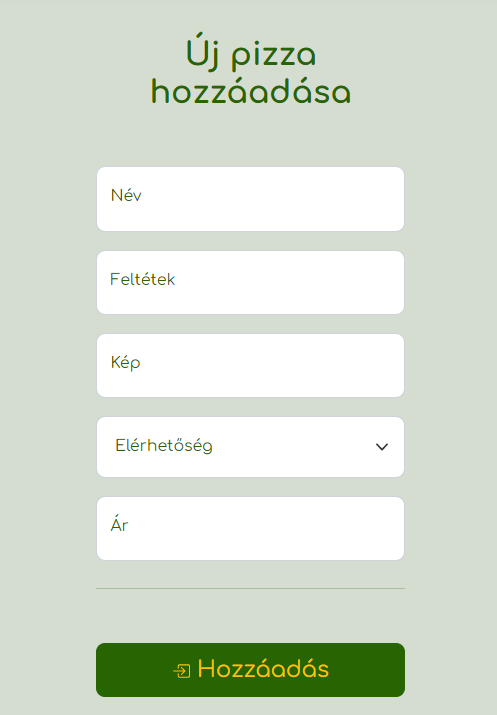
Ezek közül a törlés navigáció nélkül történik, egy confirm típusú JS megerősítéssel:



Az „Új pizza” illetve a „Módosítás” gombok csak 8-as szint felett elérhető aloldalakra irányítanak át.

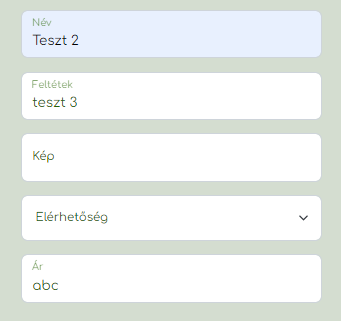
Pizzák hozzáadása és módosítása

A 2 oldal szerkezetét tekintve nagyon hasonló, fő különbségük az oldal címén kívül hogy a hozzáadás oldalon üres beviteli mezők szerepelnek oldalbetöltéskor, illetve felvitelkor ellenőrzésre kerül a bevitt név egyedisége. Módosítás oldalon ezzel szemben a kiválasztott pizza adataival előre fel vanna töltve a beviteli mezők, meggyorsítván a műveletet, és a módosítás mentésekor egy újabb confirm funkcióval ellenőrzöm, hogy a felhasználó biztosan végzett-e minden módosítással.

 A képen szöveg látható

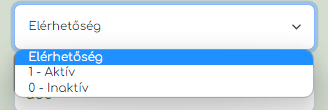
Automatikusan generált leírás

Az adatok ellenőrzése nagyobbrészt kétféle ellenőrzésen is átmegy, megvizsgálom az adat meglétét illetve formai helyességét is, amiről a hibának megfelelő visszajelzést kap a felhasználó:

 A képen szöveg, levél látható

Automatikusan generált leírás

Az elérhetőség, ami az adatbázisban számkóddal van tárolva, ezen a felületen kap szöveges címkét, egy select elemben:



Ez a megoldás az olvashatóságon túl biztosítja, hogy csak az elvárt szerkezetnek megfelelő adat kerüljön felvitelre.

Rendelések

Ez az oldal minden bejelentkezett felhasználó számára megtekinthető, de csak 7-es jogosultságszinttől felfelé szerkeszthetőek a megtekintett adatok. Az oldal főbb részei a rendelések dinamikus táblázata, fordított sorrendbe rendezve, hogy a legfrisebb rendelés szerepeljen az oldal tetején:

A képen szöveg látható

Automatikusan generált leírás

A másik blokk, ami az oldal fő logikáját tartalmazza a rendelések felviteléért és állapotkezeléséért felelős form, amely az adatbázis számára értelmezhető módon, a frontenden kialakított logikával gyűjti össze a rendelések adatait:

A képen szöveg látható

Automatikusan generált leírás

A Rendeléskezelő form elemei

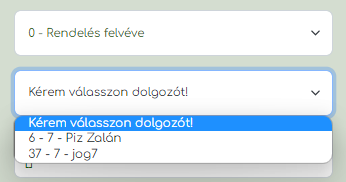
Rendelés azonosítója

Ez a mező manuálisan nem módosítható, mivel automatikusan generálódik. azonban a rendelés azonosításakor a szemléletesebb megjelenítésért ez a mező is feltöltésre kerül a mgelévő azonosítóval.

Státusz és dolgozóválasztás

Ennek a két mezőnek a tartalma szorosan összefügg. A rendelések státusza a frontenden kap olvasható címkét:

A képen szöveg látható

Automatikusan generált leírás 

Az adott rendelési státusz kiválasztásától függően, dinamikusan töltődik fel az adott státuszhoz kapcsolódó jogosultsági szintű illetve munkakörű dolgozók listája, ezzel biztosítom, hogy az adott rendeléssel való feladatot a megfelelő embernek lehessen dedikálni.

Rendelés tartalma

A rendelés tartalma JSON formátumú stringben tárolódik le, illetve kerül megjelenítésben a táblázatban. Mivel azonban ez átlag felhasználónak nem elég könnyen olvasható illetve írható formátum, hogy megtartsuk az adatok formai és tartalmi helyességét, egy olyan logikát dolgoztam ki, a pizzák dinamikusan betöltött listájából kiválasztott pizza és a pizza lehetséges méretének listájából kiválasztása után gombnyomással feldolgozza és a megfelelő formátumban menti az adatokat a rendelés tartalma mezőbe. A törlés gomb segítségével lehet alaphelyzetbe állítani az említett 2 listát.

A select mezőkkel a pizzák hozzáadása és módosítása oldalakhoz hasnolóan azt biztosítottam, hogy csak megfelelő adatok legyenek továbbítva a szerver felé mentésre. A kötetlen bevitelű mezők, a név, a telefon és az email regex kifejezésekkel lettek ellenőrizve, míg a címnél csak a mező kitöltöttségét ellenőriztem, számítva az adatok sokféleségére.

Admin (felhasználók kezelése)

Az admin oldal megjelenése és funkciói a legsokszínűbbek. Ezt az oldalt minden jogosultsági szinten el lehet érni ugyan, de a tartalma a táblázatban is jelzett módokon a következőképp módosulhat:

6-os jogosultságszinttel bezárólag funkciója a bejelentkezett felhasználó adatainak és jogosultsági szintjének ellenőrzése, valamint az alkalmazásból való kijelentkezés

A képen szöveg látható

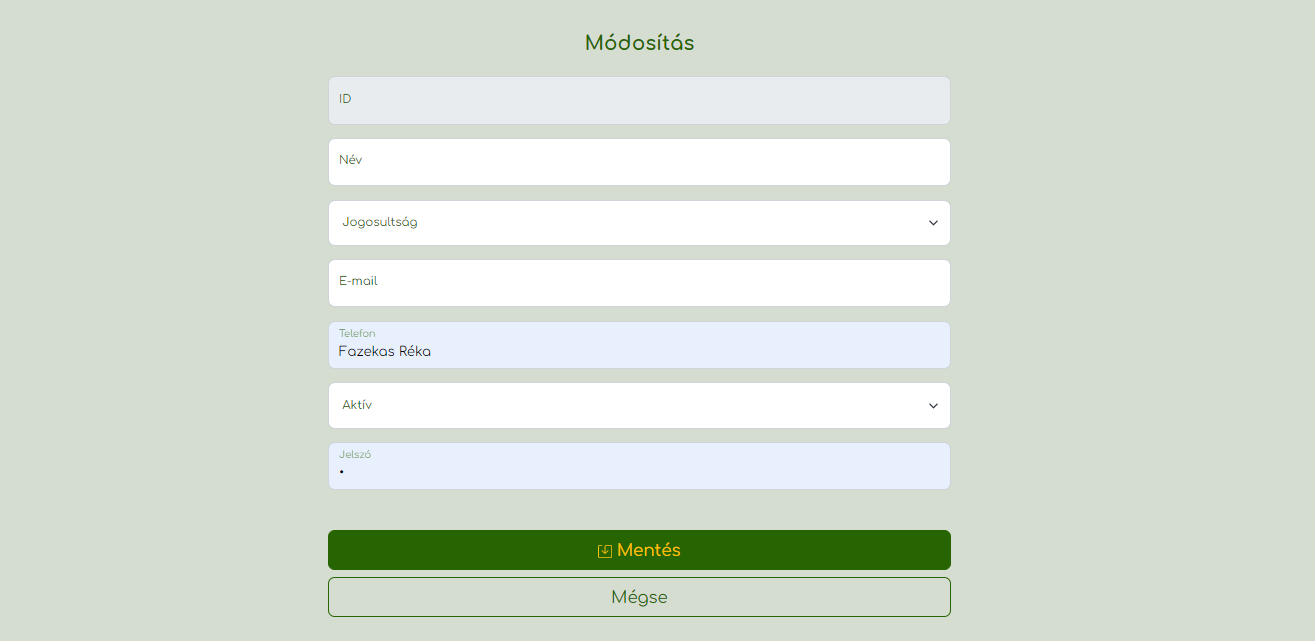
Automatikusan generált leírás

7-es szinttől az oldal kiegészül a regisztrált felhasználók listájával

A képen szöveg, képernyőkép, fedett pályás, számos látható

Automatikusan generált leírás

illetve a csak 9-es jogosultsággal rendelkező felhasználók számára engedélyezett felhasználómódosításért felelő formmal.



A form logikája a Rendelések oldal módosításához hasonló kialakítást kapott illetve használ szerkezetileg, azaz szabad kezes és kötött választási lehetőségekkel feltöltött beviteli mezőkből áll. Szerepel a már korábban felhasznált ativitás mező, amely ebben az esetben a felhasználó érvényességét jelzi (az inaktív felhasználó nem tud bejelentkezni), illetve ezen az oldalon kapnak olvasható címkét a jogosultsági mezők, ezzel együtt az egyes dolgozók munkaköre is:

*A képen szöveg látható

Automatikusan generált leírás*

Mivel rendelések és a felhasználók adatainak megjelenítése esetében előre meghatározott számú adatot kellett megjeleníteni, ezért a táblázatok olyan css tulajdonságot kaptak, amivel oldalirányban görgethetővé váltak kisebb készülékekre optimalizált nézetben:

A képen szöveg látható

Automatikusan generált leírás

# - Tesztelés

Az oldal tesztelése fejlesztés közben is folyamatosan zajlott a böngészők illetve a fejlesztői IDE-k beépített eszközeivel. A frontend elkészültével funkció- és részemről designteszteknek vetettük alá az oldalt, felmérvén az esetleges felhasználásból eredő hibákat. Ezen túl a kód átesett a Clean Code elvei szerinti vizsgálaton is, manuálisan illetve Sonarlinttel. Technikai automata tesztelésnek a GitLap verziókezelő automata tesztelésre alkalmas CI/CD folyamatát használtuk, ami következő sikeres tesztekkel zárult:

A képen asztal látható

Automatikusan generált leírás

build:

A képen szöveg látható

Automatikusan generált leírás

unit teszt:

A képen szöveg látható

Automatikusan generált leírás

lint teszt:

A képen szöveg látható

Automatikusan generált leírás

deploy teszt:

A képen szöveg látható

Automatikusan generált leírás

Az alkamazást feljesztésének szakaszát így sikeresen lezártuk, a program élesítésre illetve általános felhasználásra késznek tekinthető.

# 3. - Források, hivatkozások

1. [https://simplicable.com/IT/thin-client-vs-thick-client - 1.1](https://simplicable.com/IT/thin-client-vs-thick-client%20-%201.1) Szoftverarchitektúra
2. <https://tabnine.com/> - 1.2.1 Fejlesztői környezet, Tabnine
3. <https://vuejs.org/> - 1.2.1 Fejlesztői környezet, Volar
4. <https://gitkraken.com/> - 1.2.1 Fejlesztői környezet, GitLens
5. <https://ak-akademia.hu/mi-az-a-vue-js/> - 1.2.3 Vue.js
6. <https://getbootstrap.com/docs/5.3/getting-started/introduction/> - Bootstrap