

**Dokumentáció**

**Bevezetés**[**1**](#bevezetes)

Fejlesztés célja Páciens felület[1.1](#elso)

Fejlesztés célja Orvosi felület[1.2](#masodik)

Felület tervek[2](#felulettervek)

Asztali alkalmazás:[2.1](#felulettervek_asztali)

Reszponzív Web alkalmazás:2.2

Felhasznált technológiák[3](#felhasznalttechnologiak)

Reszoponzív Web alkalmazáshoz [3.1](#felhasznalttechnologiak_web)

Asztali alkalmazáshoz [3.2](#felhasznalttechnologiak_asztali)

A backend szerverhez [3.3](#felhasznalttechnologiak_backend)

Késítéshez használt programok [3.4](#felhasznalttechnologiak_programok)

Tesztléshez használt programok [3.5](#felhasznalttechnologiak_teszteles)

Fejlesztői környezetek:[3.6](#felhasznalttechnologiak_fejlesztoikorny)

Adatszerkezetek[4](#adatszerkezet)

Kódolási konvenciók[5](#kodolasikonvenciok)

Könyvtár szerkezet:[5.1](#konyvtarszerkezet)

Végpontok[6](#vegpontok)

Általános működés:[6.1](#altalanosmukodes)

Kódok/Algoritmusok[7](#Kodok)

Asztali alkalmazás:[7.1](#kodokasztali)

Reszponzív Web alkalmazás:[7.2](#kodokweb)

Szerver:[7.3](#kodokszerver)

Felhasználói dokumnetáció[8](#felhasznaloidoc)

Asztali alkalmazás:[8.1](#felhasznaloidocasztali)

Reszponzív Web alkalmazás:8.2

Tesztelés:[9](#teszteles)

Asztali alkalmazás:[9.1](#tesztelesasztali)

Reszponzív Web alkalmazás:[9.2](#tesztweb)

Szerver:[9.3](#tesztszerver)

Fejlesztési lehetőségek:[10](#fejlesztesilehetoseg)

Adatbázis:[10.1](#lehetosegadatbazis)

Szerver:[10.1](#lehetosegszerver)

Reszponzív Web alkalmazás:[10.1](#lehetosegweb)

Asztali alkalmazás:[10.1](#lehetosegasztali)

1. **Bevezetés:**

A Medicloud nevű program legfőbb célja az, hogy egy olyan egészségügyi alkalmazást nyújtsunk a felhasználóknak, amely segíthet a vérvételi komponenseik értelmezésében. Reméljük, hogy a program használta megkönnyíti a határértékek és a komponensek fontosságának megértését. Bizonyára már mindenki kapott kézhez A4-es papíron tengernyi vérvételi adatot amely kiértékelés nélkül maradt. Abban reménykedünk, hogy ha egy páciens tisztában van azzal ami a vérében történik, akkor az jelentősen hozzájárul a korai diagnosztikához. Legfőbbképpen ez indokolta a témaválasztásunkat.

**1.1 Páciens felület**

A fejlesztés célja, hogy egy olyan felhasználóbarát egészségügyi felhő alkalmazást hozzunk létre, amely segíti a pácienseket nyomon követni a vérvételi eredményeiket. Tudatja a felhasználókat arról, hogy melyik vérvételi komponens miért fontos.

**1.2 Orvosi felület**

Ugyanakkor a fejlesztésnek az is célja volt, hogy ne csak egy pácienseknek szánt alkalmazás készüljön el hanem egy orvosi felület is amely kiegészítve a felhasználói részt megkönnyíti az orvosok dolgát egy egy beteg adatainak a kezelése során. Ezzel az egész rendszert leegyszerűsítve. valamint részletes vérvételi adatokat megtekinteni. Ezeket törölni,vagy újat felvenni.

1. **Felülettervek:**

**2.1 Asztali alkalmazás:**

**Graphical user interface

Description automatically generated with medium confidenceRólunk oldal:**

**Text

Description automatically generated with medium confidenceGithub oldal:**

**Graphical user interface, text

Description automatically generatedFőoldal:**

**Bejelentkezés:**

**Graphical user interface

Description automatically generated**

**A picture containing letter

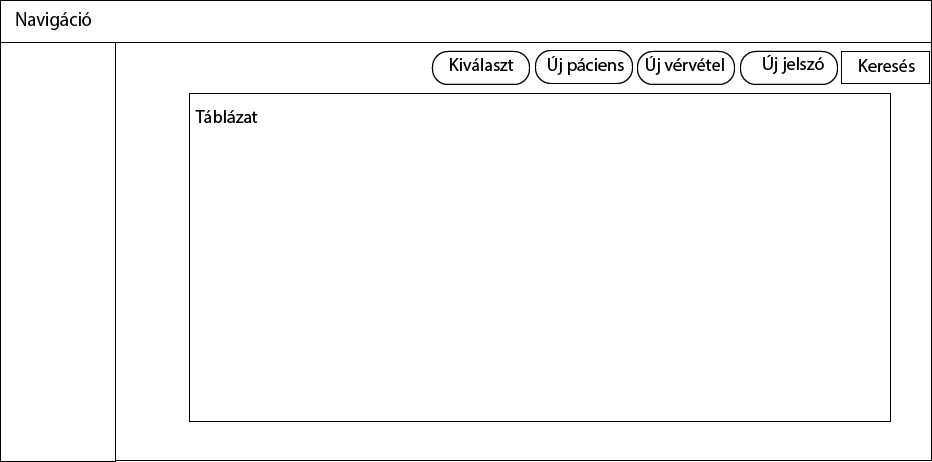
Description automatically generated** **Új jelszó felvétel:**

**Graphical user interface, application

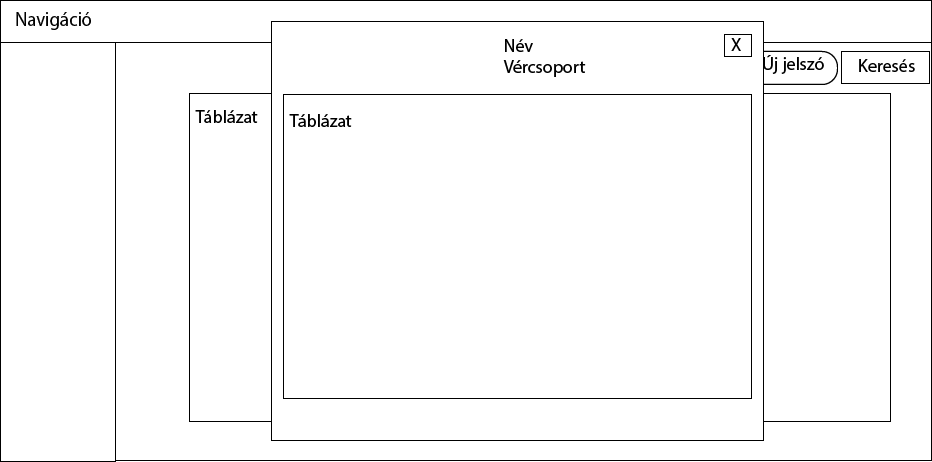
Description automatically generated** **Új páciens felvétel:**

**Diagram

Description automatically generated with medium confidenceÚj vérvétel felvétel:**

**Páciensek:**

**Páciens kiválasztása:**

****

1. **Felhasznált technológiák:**

**3.1** **Reszoponzív webes és mobil alkalmazáshoz:** A felhasználói felület kialakításához a React.js nevű keretrendszert használtunk, mivel ez nyújtott lehetőséget a vizualizációhoz és ennek a szintaktikája felelt meg számunkra a leginkább. A vizualizációhoz a Recharts nevű React.js könyvtárt és ahhoz, hogy a frontend kommunikálni tudjon a backend-del, ahhoz az Axios nevű könyvtárt használtunk.

**3.2** **Asztali alkalmazáshoz:** Ennek a felületnek a kialakításához szintén React.js-t használtunk és ahhoz, hogy a szerverrel kapcsolatba kerüljön, ahhoz Axios-t. Ahhoz, hogy ebből asztali alkalmazás legyen az Electron.js vettük igénybe. Ezt a platformfüggetlen keretrendszert használja, a Visual Studio Code-is.

**3.3** **A backend szerverhez:** A szerver kialakítása igényelte a legtöbb könyvtár beimportálását. A szerverhez tartozó JavaScript kódot a Node.js futtatja, a szervert az Express.js könyvtár. Ahhoz, hogy ez jól működjön szükség volt a Cors, Body Parser, Mysql, CookieParser, JSON Web Token és Bcrypt nevű könyvtárakra. A Cors segítségével lehet szabályozni, hogy melyik hálózatok férhetnek hozzá különböző API requestekhez. Jelenleg a localhost:3002 és a localhost:3000 tekinthető a szerver számára megbízhatónak. A Body Parser-nek köszönhetően tudunk requestekből változókat kinyerni POST requestek esetében. A Mysql-el tudunk az adatbázissal kapcsolatba kerülni. A CookieParser-rel nyerjük ki az azonosítási token-t a clienstől. A JSON Web Token-nel generálunk token-t és a Bcrypt-tel titkosítjuk a jelszavakat.

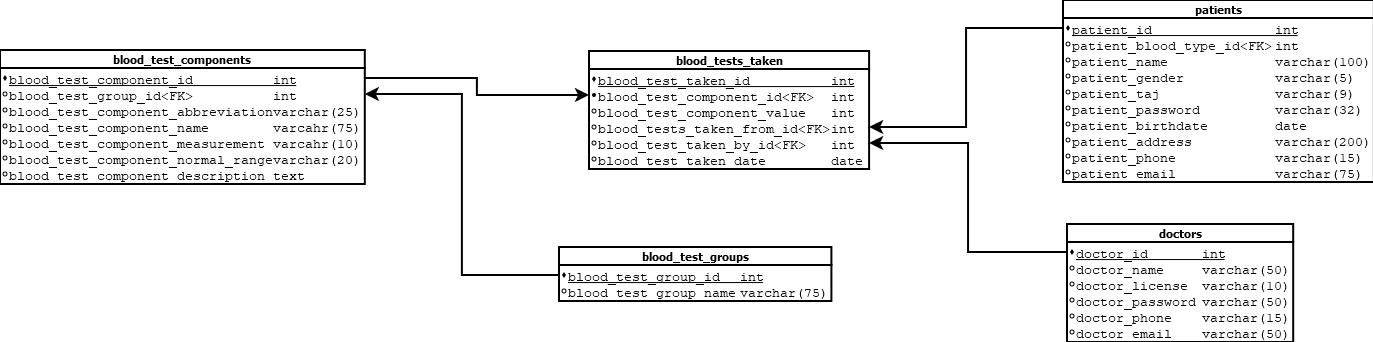
* 1. **Késítéshez használt programok:** dia, Visual studio code, Isomnia, Mariadb server, phpmyadmin
  2. **Teszteléshez használt prgoramok:** Postman, Insomnia ezeknek köszönhetően ellenőrizzük, hogy a szerver, hogy reagál abban az esetben, ha hiba lépne fel.
  3. **Fejlesztői környezetek:**

**Operációs rendszer:**

Windows 10, Windows 11

Szoftverek:

Visual Studio Code, XAMPP, Insomnia, Postman

**4.Adatszerkezetek:**

**5.Kódolási konvenciók**

A fejlesztés során végig git verzió követőt használtunk.

**5.1** **Könyvtár szerkezet:**

medicloud/

+---database

| medicloud.sql

|

+---desktop-app

| | .gitignore

| | package-lock.json

| | package.json

| | README.md

| | yarn.lock

| |

| +---dist

| | Medicloud.VBS

| | MedicloudSetup.bat

| | MedicloudStart.bat

| |

| +---public

| | index.html

| | main.js

| | snake-icon.png

| |

| \---src

| | App.css

| | App.js

| | index.js

| |

| +---assets

| | background.png

| | github.png

| | icon.ico

| | icon.png

| | medicloud-text-white.png

| | medicloud-text.png

| |

| +---components

| | Button.css

| | Button.js

| | Modal.css

| | Modal.js

| | Navbar.css

| | Navbar.js

| | SidebarData.js

| |

| +---pages

| | About.css

| | About.js

| | AddNew.css

| | AddNew.js

| | Github.css

| | Github.js

| | Home.css

| | Home.js

| | Login.css

| | Login.js

| | Patients.css

| | Patients.js

| |

| \---utils

| API.js

| authorization.js

| ProtectedRoutes.js

|

+---doc

| fejlesztoi-dokumentacio.docx

|

+---responsive-app

| | .gitignore

| | package-lock.json

| | package.json

| | README.md

| |

| +---public

| | index.html

| | snake-icon.png

| |

| \---src

| | index.css

| | index.js

| |

| +---assets

| | background.png

| | medicloud-icon.png

| | medicloud-icon333333.png

| | medicloud-text.png

| |

| +---components

| | +---footer

| | | footer.css

| | | FOOTER.js

| | |

| | \---sidebar

| | sidebar.css

| | SIDEBAR.js

| |

| +---pages

| | +---login

| | | login.css

| | | LOGIN.js

| | |

| | \---patient

| | | patient.css

| | | PATIENT.js

| | |

| | +---bloodtest

| | | bloodtest.css

| | | BLOODTEST.js

| | | component.css

| | | COMPONENT.js

| | | INFO.js

| | | visualization.css

| | | VISUALIZATION.js

| | |

| | \---setting

| | PASSWORD.js

| | setting.css

| | SETTING.js

| |

| \---utils

| API.js

| authorization.js

| dataRefinery.js

| PROTECTED\_ROUTES.js

|

\---server

| .gitignore

| config.json

| package-lock.json

| package.json

| server.js

| setup.bat

|

\---SQL\_COMMANDS

DOCTOR\_SQL.json

PATIENT\_SQL.json

A database tartalmazza az adatbázis importálásához szükséges sql filet. A desktop-app tartalmazza az asztali alkalmazást. A doc könyvtárban a felület tervek és a dokumentációk találhatók a programról. A responsive-app könyvtárban található a webalkalmazás. A server könyvtárban a backend server.

**6.Végpontok**

**Páciens:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Végpont** | **Metódus** | **Azonosítás** | **Leírás** |
| */*patient-authetication | post | nem | Páciensek bejelentkezése a webalkalmazásba |
| /patient-profile-data | get | igen | A bejelentkezett páciens adatainak lekérdezése |
| /patient-blood-test-dates | get | igen | A belépett páciens vérvételi időpontjait kérdezi le |
| /patient-blood-test-results | post | igen | Dátum alapján belépett páciens vérvételi adatait kérdezi le |
| /patient-blood-test-statistics | get | igen | Vérvételi adatokat kérdezi le |
| /patient-blood-test-results | get | igen | A belépett páciens vérvételi adatait kérdezi le |
| /patient-change-address | post | igen | A belépett páciens lakcímét változtatja meg |
| /patient-change-phone | post | igen | A belépett páciens telefonszámát változtatja meg |
| /patient-change-email | post | igen | A belépett páciens email címét változtatja meg |
| /patient-change-password | post | igen | A belépett páciens jelszavát változtatja meg |
| /patient-components | get | igen | Ha bejelentkezett a páciens a vérvételi komponenseket kérdezi le |
| /doctor-authentication | post | nem | Doktorok bejelentkezése az asztali alkalmazásba |
| /add-patient | post | igen | A belépett doktor a patients tábla minden adatábol tud újat rögzíteni |
| /selectPatient | get | igen | A belépett doktor ha kiválaszt egy pácienst neki a vérvételi adatait kérdezi le |
| /patients | get | igen | A teljes páciens tábla lekérdezése |
| /delete-patient | post | igen | A kiválasztott pácienst törli |
| /delete\_bloodtest\_taken | post | igen | A kiválasztott páciens vérvételi adatai törli |
| /delete\_Auth | post | igen | A kiválasztott páciens bejelentkezési adatait törli |
| /addPatientBloodTestData | post | igen | A kiválasztott páciens vérvételi adatait adja hozzá |
| /addPassword | post | igen | A kiválasztott páciens jelszavát adja hozzá |

**6.1 Általános működés**

A kéréseket fogad, melyek tartalmazzák a műveletekhez szükséges megfelelő adatokat. A kényes műveletek végpontjai védettek, authentikációt kell használ azok eléréséhez (orvosi végpontokhoz orvosi bejelentkezés szükséges(verifyDoctor), páciens végpontokhoz a páciens bejelentkezése szükséges(verifyPatient)).Az adatokat JSONformátumban fogadja és dolgozza fel. Az adatfeldolgozása a requestekben történik ezek kérik le az adatot az adatbázisból vagy módosítják, törlik azt. Az adtok felvétele is itt történi ugyan ilyen módon.

**7. Kódok/Algoritmusok**

**7.1 Asztali alkalmazás:**

**index.js**: Itt történik a route-ok bekötése. A Protecte\_Routes.js komponensnek köszönhetően, a Home.js komponens csak a bejelentkezés után tud megjelenni. A bejelentkezésig a Login.js komponens látható.

**utils/API.js:** Az Axios beimportálása és configurálásáért felel.

**utils/authorization.js**: A token-nel kapcsolatos műveletek itt történnek. A bejelentkezéskor kapott token document.cookie-ként tárolódik. Ennek az élettartama 1 óra, ahhogy a szerver oldal által generált token-nek is. Egy watchExpiration nevű függvénnyel figyeljük, hogy amikor lejár, akkor visszavezeti a felhasználót a bejelentkezési felületre.

**Components:**

**Button.js:** Visszaad egy gombot ami több oldalra is komponensként beemelhető az onclick és a title pedig változtathátó.

**Modal.js:**

toggleModal(): A modalt láthatóvá teszi vagy eltünteti. Attól függően hogy éppen megvan e nyitva vagy nem.

**Navbar.js:**

showSidebar(): A navbart láthatóvá teszi vagy eltünteti. Attól függően hogy éppen megvan e nyitva vagy nem.

A SidebarData ból a map() függvénnyel kilistázzuk az adatokat Link elem formájában amik megkapják a path, icon, title változókban a JSON be irt adatokat.

A logout elem nem tartozik ezek közé mert ott történik a kijelentkezés kezelése amit nem lehet elvégezni az előző módszerrel.

**Sidebardata.js:** JSON formátumba itt lehet felvenni a navbar-nak az elemeit aminek itt tudjuk meg adni a nevét, ikonját, útvonalát, osztálynevét.

**Pages:**

**About.js:** Rövid leírás az oldalról annak bemutatása Logók megjelenítése**.**

**AddNew.js:**

addNew(): Egy post kérést küld az /add-patient végpontra (nevet, vércsoportot, nemet, tajszámot, születésidátumot, lakcímet, telefonszámot, emailt) majd meghívja a handleRest nevű függvényt.

Az adatok kiszedése:

az onChange={(e) => setVáltozó e.target.value)} kerszetül történik.

handleReset(): A document.getElementById nevű függvénnyel elkéri az inputnak az id-ját majd a value értékét egy üres stringre cseréli.

**Github.js:**

Az oldal készítőinek a github linkjei találhatok itt meg amire ha rákattintunk a github oldalukra irányit át minket azalkalmazás a window.location.href függvény segítségével.

**Home.js:**

navigateToPtients(): felveszünk egy navigate változót ami egyenlő a useNavigate függvénnyel majd átadjuk neki az útvonala amire menni szeretnénk. Majd a függvényünket átadjuk a gomb onClick funkciójának.

A másik gomb a webalkalmzás linkjére tud minket irányitani a window.location.href függvény segítségével.

**Login.js:**

handleLogin(): Egy post kérést küld a /doctor-authentication végpontra ahol a (licence, password) adatokat várja amit az onChange={(e) => setVáltozó e.target.value)} kerszetül szed ki az inputból. Majd az ablakot újra tölti.

**Patients.js:**

**Modal():** Itt hozzuk létre a Modalt ami megjeleníti a kiválasztott páciensek részletes vérvételi adatait.

getPatientData(): egy post kéréssel kérdezzük le a /selectPatient végponton a páciensek részletes vérvételi adatai a blood\_tests\_taken táblából. A useEffectnek átadva a getPatientData-t táblázatban jelenitjük meg az adatokat. A map függvény segítségével.

selectOne(): bejövő paraméterként az arg-ot kapja meg amit átad a setSelectednek a táblázat tr elemének az onClick metódusára hívjuk meg igy a sorra kattintás esetén kilehet olvasni az adatokat a táblázatból.

deleteBloodtests(): Bejövő paraméterként fogadja a selectedet innen kapja meg a kiválszatot sor id-ját. Egy post kérést küld a delete\_blood\_tests\_taken végpontra ahol átadjuk a kiválaszott vérvételnek az id-ját is. Ezt követően végrehajtja a getPatientData függvényt, hogy lefrissüljön a táblázat. A gomb onClick metódusán kerül meghívásra.

**Patients():** Itt hozzuk létre magát a Patients oldalt ahol a páciensek alap adatai jelennek meg egy táblázatban, amik között lehet keresni a hozzáadás gombra kattintva pedig navigálni a hozzáadás oldalra. A sorba belekattintva kiválasztás gomb megnyomása utána ugrik elő a Modal a kiválasztott Páciens vérvételi adataival.

selectOne(): bejövő paraméterként az arg-ot kapja meg amit átad a setSelectednek a táblázat tr elemének az onClick metódusára hívjuk meg igy a sorra kattintás esetén kilehet olvasni az adatokat a táblázatból.

openModalUp(): a selectOne metódus álatal a sorból kiválasztott nevet ha megkapja akkor fenyitja a modalt aminek a h1 sorába a név és a vércsoport töltődik be. Más esetben nem nyílik meg.

closeModal(): a Modal megjelenítését false-ra állítja igy bezáródik a felugró ablak.

openModalDataUp: a selectOne metódus álatal a sorból kiválasztott nevet ha megkapja akkor fenyitja a modaltDatát. Ellenkező esetben nem.

closeModalData(): a ModalData megjelenítését false-ra állítja igy bezáródik a felugró ablak.

openModalPasswordUp(): a selectOne metódus álatal a sorból kiválasztott nevet ha megkapja akkor fenyitja a modalPasswordot. Ellenkező esetben nem.

closeModalPassword(): a ModalData megjelenítését false-ra állítja igy bezáródik a felugró ablak.

getPatients(): egy post kéréssel kérdezzük le a /patients végponton a páciensek adatait a patients táblából. A useEffectnek átadva a getPatients a táblázatban jelenítjük meg az adatokat. A map függvény segítségével.

filterData(): az input mezőbe beírt adatok value-ját ellenőrizzük le amit String-ként kérünk be. És kiskarakterekre alakítjuk igy nagy és kis betű beírásánál is működni fog. Majd a pacienseket a táblába seteljük ha nagyobb a value érték mint nulla akkor a tableFileterre hívjuk meg a map függvényt ha nem nagyobb mint nulla akkor meg a patients-re.

navigatetoAddNew(): Át navigál a /patients/Addnew url re.

deletePatient(): egy Post kérést küld és paraméterként kapja a selectedet innen tudja melyik a kiválasztott páciens akit törölni szeretnénk. Az id-nak pedig átadjuk a selected.patient\_id-t. Majd meghívjuk a getPatients() metódust hogy lefrissüljön a táblázat.

deleteAuth(): Az patient\_authentication táblában lévő páciensek adatának a törlésére szolgál hogy páciens törlés esetén azok jelszava is törlődjön. Egy Post kérést küld és paraméterként kapja a selectedet innen tudja melyik a kiválasztott páciens akit törölni szeretnénk. Az id-nak pedig átadjuk a selected.patient\_id-t. Majd meghívjuk a getPatients() metódust hogy lefrissüljön a táblázat és a deletePatientet, hogy törlődjön a páciens a törlésgombra igy ez a metódus kerül majd rá az onClickre().

**ModalData():** Itt hozzuk létre a Modalt ami egy páciens részletes adatainak a felvételéhez szükséges. Ezen a felület szolgál az adatok felvételére.

addNew(): Egy post kérést küld az /addPatientBloodTestData végpontra (komponensId, komponens értéke, a páciens id-ja, a vérvételt végző orvos id-ja, vérvétel dátuma) majd meghívja a handleReset nevű függvényt. A legördülő menük azt az id-t adják vissza ami az adott komponensnek az id-ja.

handleReset(): A document.getElementById nevű függvénnyel elkéri az inputnak az id-ját majd a value értékét egy üres stringre cseréli.

**ModalPassowrd():** Itt hozzuk létre a Modalt ami egy páciens jelszavának a felvételéhez szükséges.

addPassword(): Egy post kérést küld a /addPassword végpontra (páciens id-ja, jelszó) majd meghívja a handleReset nevű függvényt. Az id a patients táblázatból a kiválasztást követően érkezik meg.

**7.2 Reszponzív Web alkalmazás:**

**index.js** > Itt történik a route-ok bekötése. A PROTECTED\_ROUTES.js komponensnek köszönhetően, a PATIENT.js komponens csak a bejelentkezés után tud megjelenni. A bejelentkezésig a LOGIN.js komponens látható.

**utils/API.js** > Az Axios beimportálása és configurálásáért felel.

**utils/authorization.js** > A token-nel kapcsolatos műveletek itt történnek. A bejelentkezéskor kapott token document.cookie-ként tárolódik. Ennek az élettartama 1 óra, ahhogy a szerver oldal által generált token-nek is. Egy watchExpiration nevű függvénnyel figyeljük, hogy amikor lejár, akkor vissza vezeti a felhasználót a bejelentkezési felületre.

**utils/dataRefinery.js** > Az adatbázisból kapott nyers adatokat dolgozza fel úgy, hogy az vizualizálható legyen.

**pages/login/LOGIN.js** > Ez a komponens egy űrlap. A páciens a tajszáma és a jelszavával itt tud belépni. Ezeket a változókat, az API.js segítségével a request body-jába csomagolva elküldi a "/patient-authentication" címre. Abban az esetben, ha az azonosítás sikeres, megjelenik a PATIENT.js komponens.

**pages/patient/PATIENT.js** > Ezen a komponensen sorakozik fel az összes bejelentkezést igénylő komponens ( SIDEBAR.JS, FOOTER.js, BLOODTEST.js, SETTING.js, COMPONENT.js ).

**components/sidebar/SIDEBAR.js** > Egy API request lekéri a pácienshez tartozó vérvételi dátumokat és jeleníti meg. Ezekből választhat a páciens.

**components/footer/FOOTER.js** > Itt tárolódnak a gombok. Az amelyik rendereli a SETTING.js komponenst vagy lefuttatja a kijelentkezés parancsát.

**pages/patient/bloodtest/BLOODTEST.js** > Lekéri a SIDEBAR.js-en kiválasztott dátum vérvételeit és egy map ciklussal átadja ezeket az adatokat a VISUALIZATION.js-nek, aminek köszönhetően megjelenik a vizualizációs csík.

**pages/patient/bloodtest/VISUALIZATION.js** > Recharts könyvtárnak köszönhetően megvalósítja a vizualizációt.

**pages/patient/bloodtest/COMPONENT.js** > Egy vízszintes listát ad vissza az adatbázisból lekért összes vérvételi komponensről.

**pages/patient/bloodtest/INFO.js** > A COMPONENT.js-en kiválasztott komponensről mutat információkat.

**7.3 Szerver:**

**Middleware:**

**PATIENT\_GUARD** függvény > A páciens klienstől kapott token-t ellenőrzi, levédi a hozzá tartozó API requesteket.

**DOCTOR\_GUARD** függvény > Az orvos klienstől kapott token-t ellenőrzi, levédi a hozzá tartozó API requesteket.

**REQUEST-ek:**

POST ("/patient-authentication") > Lekéri a request body-jából a tajszámot és a jelszót. Egy SQL-paranccsal lekéri a tajszámhoz tartozó adatokat. Abból a hash-elt jelszót összehasonlítja az inputból származó jelszóval. Amennyiben megeggyezik, akkor a lekérő kliens cookie-jába betölti a JSON Web Token-el generált tokent.

GET ("/patient-profile-data") > Lekéri a token-ba hash-elt ID-val a páciens alap adatait.

GET ("/patient-blood-test-dates") > Lekéri a pácienshez tartózó egyedi időpontokat.

POST ("/patient-blood-test-results") > Lekéri a vérvételi adatokat a body-ból kapott időpont segítségével.

POST ("/patient-change-address") > A body-ból kapott címre átírja a páciens címét.

POST ("/patient-change-phone") > A body-ból kapott telefonszámra átírja a páciens telefonszámát.

POST ("/patient-change-email") > A body-ból kapott emailre átírja a páciens emailjét.

POST ("/patient-change-password") > A régi jelszó helyes megadásával megváltoztatja a páciens jelszavát.

GET ("/patient-components") > Lekéri az összes vérvételi kompoensre vonatkozó adatokat.

POST ("/doctor-authentication") > Lekéri a request body-jából a licenset és a jelszót. Egy SQL-paranccsal lekéri a licensehez tartozó adatokat. Abból a hash-elt jelszót összehasonlítja az inputból származó jelszóval. Amennyiben megeggyezik, akkor a lekérő kliens cookie-jába betölti a JSON Web Token-el generált tokent.

GET (”/patients”) > Lekéri az összes páciens összes adatát.

POST (”/selectPatient”) > Egy páciens id-jának a megadásával visszadja annak vérvételi adatait.

POST (”/add-patient”) > A body-ból kapott adatokat felveszi a patietns táblába.

POST (”/delete-patient”) > A páciens id helyes megadásával kitörli a kapott id-n lévő pácienst.

POST (”/delete\_bloodtests\_taken”) > A vérévtel id helyes megadásával a kapott id-n lévő vérvételt törli ki.

POST (”/delete\_Auth”) > A páciens authentikációs id helyes megadásával a kapott id-n lévő jelszót törli ki.

POST (”/addPatientBloodTestData”) > A vérvétel id megadásával a body-ból kapott adatokat veszi fel a vérvételi táblába.

POST (”/addPassword”) > A páciens id megadásával a body-ból kiolvasva jelszót vesz fel neki.

**8. Felhasznalói dokumentáció:**

**8.1 Asztali alkalmazás:**

**A picture containing application

Description automatically generatedIndítás:** A repository letöltése után ha fut a mysql akkor importáljuk be a database mappában található teszt adatokkal feltöltött adatbázist. Miután ez meg történt telepítsük a backend szervert a setup.bat-el majd indítsuk el a start.bat el ha a szerver fut utána végezzük el az asztali alkalmazással is ugyan ezt majd a Medicloud.vbs állománnyal indítsuk el azt.

**Graphical user interface

Description automatically generatedBejelentkezés**: A porgram elindulásakor egy bejelentkező felület fogad minket ahol az orvosok az adataikkal betudnak lépni.

Graphical user interface

Description automatically generatedAz oldalak közötti navigációra a egy navigációs sáv szolgál.

Bejelentkezés hiányában csak a rólunk és Github felületek érhetők el egyéb esetben visszairányít minket a program.

Graphical user interface, application, website

Description automatically generatedBejelentkezés után a főoldal fogad minket, ahonnan a páciensekhez vagy a weboldalra lehet navigálni.

**Megtekintés:** A pácienseket kiválaszthatjuk bejelentkezés után már navigációs menüből vagy akár a főoldalól is. Ahol a páciensek listája fogad majd minket a róluk tudott alap adataikkal Név, Email stb..

Table

Description automatically generated

**Graphical user interface, website

Description automatically generatedKeresés:** A páciensek listájában van lehetősége az orvosoknak keresésre, amit megtehetnek akármelyik oszlop adatai alapján.

Egy páciens vérvételi adatai úgy tekinthetjük meg, hogy rákattintunk (1) a kívánt személyre majd megnyomjuk a kiválaszt gombot (2) és felugrik egy ablak, ahol láthatjuk egy táblázatban az adatokat.

Table

Description automatically generated

4

1

2

Graphical user interface

Description automatically generated

3

**Törlés:** Törlésre is van lehetőség ezt úgy lehet megtenni, hogy a kis kuka ikonra kattint. Elsőnek az orvos köteles törölni egy páciens vérvételi adatait (3) és csak utána törölheti az adott pácienst a rendszerből (4).

**Új páciens felvételének menete:**

A screenshot of a computer

Description automatically generated with medium confidenceTable

Description automatically generatedÚj pácienst az új páciens gombra kattintva (1) tudunk felvenni, ami egy új oldalra navigál minket itt a páciens adatait megadva van lehetőségünk őt rögzíteni az adatbázisban a felvesz gomb megnyomásával. (2) A gomb megnyomása után a mezők ürülnek és ha szeretnénk folytathatjuk újabb páciens felvételével. A felvétel során adott mezőkben például: Taj, Dátum csak számot lehet beírni vagy naptárból választani.

2

1

**Új vérvétel felvételének menete:**

Új vérvétel régebbi vagy újonnan felvett pácienshez egyaránt vehetünk fel. Elsőnek kikell választani a táblázatból a kívánt pácienst a sorába kattintanunk, mint ahogy azt a kiválasztásnál is tettük. (1) Után kattintunk az Új vérvétel gombra a menüsorunkban. (2) Megjelenik egy új ablak, ahol megtudjuk adni a mért értéket és kitudjuk választani a komponenst és az orvos nevét legördülő menükből és a dátumot naptárból. Itt sem lehet meg adni csak számot például az érték mezőben az elgépelések elkerülése végett. A felvétel gomb megnyomására az adatok rögzülnek egyből a mezők pedig ürülnek és folytathatjuk a további vérvételek felvételét. (3)

Table

Description automatically generated

1

2

Graphical user interface, website

Description automatically generated

3

**Új jelszó megadása a páciensnek:**

Table

Description automatically generatedHa rendszerből került felvételre egy új páciens ahhoz hogy ő betudjon jelentkezni az adataival kell neki adnunk egy jelszót is ezt az orvos határozza meg. Az előzökhöz hasonlóan itt is a sorba kattintva határozzuk meg kinek szeretnénk új jelszót megadni. (1) Utána az Új jelszó gombra kattintva jelenik meg az ablak, (2) ahol beírhatjuk a jelszót. Amit a felvesz gomb megnyomásával tudunk Graphical user interface, website

Description automatically generatedrögzíteni. (3)

2

1

3

A navigációs sávban meg találjuk a rólunk és a github oldalakat ahol alap információk vannak.

Graphical user interface

Description automatically generated

Graphical user interface, application

Description automatically generated

**9. Tesztelés:**

**9.1 Asztali alkalmazás:**

[**desktop-app-test.xlsx**](file:///C:\Users\gyuri\Desktop\medicloud\medicloud\doc\desktop-app-test.xlsx)

****

**9.2 Reszponzív Web alkalmazás:**

**9.3 Szerver:**

**Text

Description automatically generatedGraphical user interface, text

Description automatically generatedGraphical user interface, text

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidenceGraphical user interface, application

Description automatically generatedGraphical user interface, text, application

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidenceGraphical user interface, application, Teams

Description automatically generatedGraphical user interface, text

Description automatically generatedGraphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generatedGraphical user interface, application

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidence**

**10. Fejlesztési lehetőségek:**

**10.1 Adatbázis:**

A jövőben szivesen kibővítenénk az alkalmazás adatbázisát a jelenleginél több vérvételi komponenssel.

Ezt követően csoportokra bontanánk ezeket (lipid panel, máj panel, hormonok, metabolikus panel, ülepedési sebesség, stb...).

**10.2 Szerver:**

A szerver jelenlegi formájában is már elég biztonságosan levédi az adatokat, viszont minden létrehozott token minimum 1 óráig él, ennek megszerzésével lekérhetjük 1 órán át mások adatát. Erre az esély nagyon alacsony, viszont létrehoznánk egy feketelistát az összes már használt token-hez. Erre a listára kijelentkezéskor kerülnének fel a tokenek így, ha valaki manuálisan kijelentkezik akkor a token élettartama lerövidül a bejelentkezéstől a kijelentkezésig tartó időre. Ennek ellenörzését beültetnénk a middleware-be. Ahhogy ez a lista elér egy bizonyos mennyiséget, ami már a program számára szignifikánsan lelassítja a middleware-t, akkor törlésre kerül a feketelista és a szerver páciens token secret-je megváltozik ezzel hatástalanodik az összes létező token és helyet ad újaknak.

**10.3 Reszponzív Web alkalmazás:**

A jelenlegi design-t alapul véve egy ennél is felhasználó barátabb felületet alakítanánk ki.

**10.4 Asztali alkalmazás:**

A design tovább fejlesztése annak teljesen reszponzívvá tétele, hogy akár telefonon is működőképess lehessen. Egyes funkciók kibővítése pl: lehessen komponenseket is felvenni. Még jobban felhasználó barátabbá tenni a felületet.

**Készitette:**

Hegedűs György Károly

Oláh János Gergely