

**Dokumentáció**

**Bevezetés1**

Fejlesztés célja Páciens felület1.1

Fejlesztés célja Orvosi felület1.2

Felület tervek2

Felhasznált technológiák3

Reszoponzív webes és mobil alkalmazáshoz 3.1

Asztali alkalmazáshoz 3.2

A backend szerverhez 3.3

Késítéshez használt programok 3.4

Tesztléshez használt programok 3.5

Fejlesztői környezetek:3.6

Adatszerkezetek4

Kódolási konvenciók5

Könyvtár szerkezet:5.1

Végpontok6

Általános működés:6.1

Funkciók7

Asztali alkalmazás:7.1

Web alkalmazás:7.2

Fejlesztői környezetek8

Felhasználói dokumnetáció9

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. **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.

**3.3** **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.

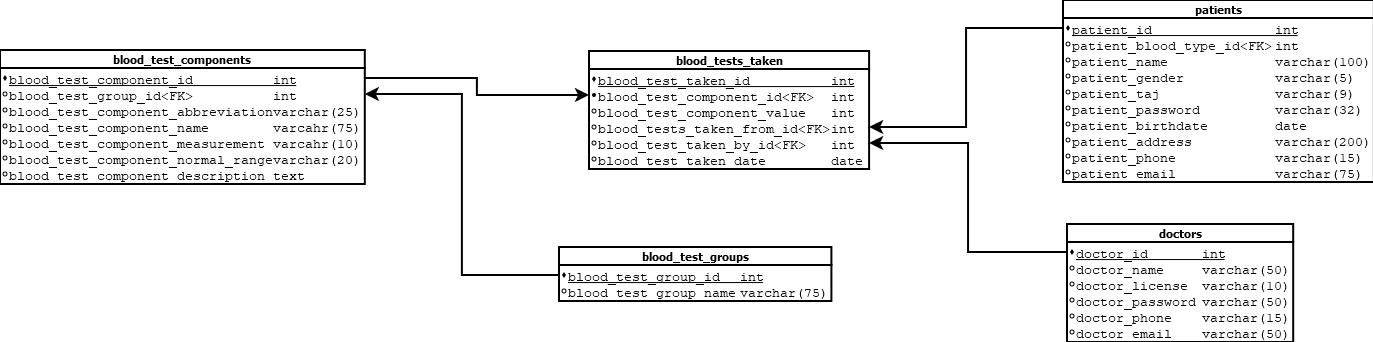
* 1. **Késítéshez használt programok:** dia, Visual studio code, Isomnia, Mariadb server, phpmyadmin
  2. **Teszteléshez használt prgoramok:** Postman, Insomnia ezeknek köszönhetően ellenőrizzük, hogy a szerver, hogy reagál abban az esetben, ha hiba lépne fel.
  3. **Fejlesztői környezetek:**

**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 | 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 REST API http 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. Funkciók**

**7.1 Asztali alkalmazás:**

**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.

1. Algoritmusok
2. Tesztelés
3. Ismert hibák
4. Készítők:
   1. Oláh János Gergely
   2. Hegedűs György