# Szakdolgozati munkaterv

**Hallgató neve: Sinkó Ábel**

**Neptun kód: YFE5BR**

**Témavezető neve: Balla Tamás**

**Szakdolgozati téma: Alkalmazásfejlesztés WEB alapokon**

## Tartalom

# Téma általános leírása

Szakdolgozatom célja egy orvosi webalkalmazás fejlesztése. Jómagam Crohn-betegséggel küzdök ebből kifolyólag élethosszig tartó kapcsolatot szükséges ápolnom kezelőorvosommal. Ebből ugyanakkor az is adódik, hogy sokszor szükséges egyéb orvosi kezelésekre járnom és fizikai kapcsolatot tartanom az orvosokkal az alapbetegségemen túlmenően, ezért hasznosnak érzem egy olyan rendszer megalkotását amely transzparens mind a betegek, mind pedig az orvosok számára.

A weboldal rendelkezik egy általános, publikus felülettel, amely leírja az adott orvosi céget, pontosan mivel is foglalkozik, milyen szolgáltatásokat nyújt, továbbá az egyes szolgáltatások áráról is nyilatkozik.

A weboldal négy részre bontható. Az első a publikus, tehát mindenki számára elérhető felület, a második betegek, a harmadik az orvosok és a negyedik az adminok számára készülő felület lenne, három különálló bejelentkezési felülettel. Ebből adódik, hogy a publikus felületet kivételével minden autentikáció kötletes.

1. Publikus felület: Bárki láthatja aki a weboldalra érkezik. Tartalmazza az orvosi cég leírását, illetve az árlistát.
2. Páciens felület: A betegek be tudnak jelentkezni, illetőleg képesek regisztrálni a weboldalra. Utóbbihoz szükségük lesz a nevükre, születési dátumukra, TAJ számukra, illetve lakcímükre. Bejelentkezés után a felhasználóknak lehetőségük nyílik a saját felhasználói profiljuk szerekesztésére, mely jelen esetben a lakcím megváltoztatását jelenti. Ezen túlmenően lehetőségük van a betegeknek kezelésekre, konzultációkra jelentkezni az általuk kiválasztott szakorvosnál. A szabad időpontok orvos specifikusak, tehát maguknak a szakorvosoknak kell szabad időpontokat meghatározniuk.
3. Orvosi felület: A szakorvosi bejelentkező felület kizárólag bejelentkezést tesz lehetővé. Az orvos fel tud venni szabad időpontokat amikor biztosan tud pácienst fogadni, illetve ezen túlmenően egy naptári felületen keresztül látja a lefoglalt időpontokat. Az orvos által felvett szabad időpontok a kizárólag páciensek számára publikusak, így képesek lefoglalni azokat.
4. Admin felület: Egy olyan felület melyet adminok használhatnak. Egy dashboard felületet kapnak ahol a felhasználókat és a hozzájuk rendelt jogosultsági szinteket tudják kezelni. Ezen túlmenően képesek orvosokat felvenni vagy eltávolítani a rendszerből. Ennél fogva tőlük függ, hogy egy orvos be tud-e lépni vagy sem, mivel az orvosi felülethez csak bejelentkező felület tartozik.

# Téma relevanciája

Azért választottam témának egy orvosi/magánegészségügyi weboldal létrehozását mert úgy gondolom, hogy jelen pillanatban az ország állami egészségügye, és így az emberek is hatalmas nehézségeken mennek keresztül. Az állami egészségügy illetve annak működése azt hiszem nem szorul bemutatásra egyikünknek sem. Nagyon jól ismerjük áldásait és átkait. Hosszú váró listák, melyek akár 1 évet, vagy kevésbé szerencsés esetben akár éveket is felölelhetnek, nem megfelelő minőségű ellátás, ellenszenvet kiváltó kórházi atmoszféra. Ide értem a kórházak belső kinézetét és az ott uralkodó általános hangulatot. Említést érdemel a telefonos elérhetőség problémáját is, hiszen nem ritka jelenség, hogy akár órákig kell próbálkozni telefonon még a recepciót elérjük főmunkaidőben. Nem hiszem, hogy ne lenne legalább egy olyan dolog a felsoroltak közül amivel nem találkozott volna polgártársaink közül bárki is. A COVID járvány megmutatta, hogy szükség volna egy rugalmas és korszerű egészségügyi rendszerre. Vannak változások a jelenlegi rendszerben melyeket előnyösnek ítélek, viszont ezek még így sem tartják az egészségügyet versenyképes állapotban. Észrevehető egy tendencia a magánegészségügyi szektorban. Arról van szó, hogy az utóbbi időben az állami és a magánegészséügyi ellátás színvonala közötti szakadékszerű különbség miatt az emberek egyre inkább nyitottabbak a magánegészségügyi szektor felé, annak ellenére, hogy ez az emberek megtakarításra szánt anyagi forrásait terheli. Ugyanakkor az emberek a saját egészsügket figyelembevéve mégis hajlamosabbak kiadni ezeket az összegeket a jó minőségű ellátásért cserébe. Ebbben természetesen jelentős szerepe van a fentebb említett problémáknak melyek az állami egészügyben jelen vannak. Hoznék a saját életemből egy példát: édesapámnak porckorong sérve lett. Műtétre szorult volna már 2023 februárjában, ellenben időpontja a műtétre valamikor 2023 év végén lett volna az állami egészségügyben. Ezzel szemben a Medicoverben 2 hónapon belül sor került a műtétre, az ország egyik  legjobb idegsebésze által. Érezzük a különbséget, ugye? Egy beteg ember fizetésének a kiesése hosszú távon mindenképpen érezteti hatását. Ebből kifolyólag a munkabért terhelő egészségügyi járulékon túlmenően is vannak emberek akik hajlandóak kifizetni a magánegészségügyi szolgáltatások anyagi vonzatait. Ellenkező esetben az állami egészségügyben lévő hosszú várólisták miatt egy betegség sokkal súlyosabb lefolyású is lehet és jobban is éreztetheti a betegre és családjára gyakorlot hosszútávú hatását. Saját példám, hogy egyes orvosok kevésbé rugalmasak. Egy recepet felírásához több tíz kilométert kellett megtennem, mert az adott kezelőorvos nem élt a telefonos konzultáció adta lehetőségekkel. Ez a fajta rugalmatlanság én úgy ítélem a modern világban kevésbé megengedhető már. A leírtakat összegezve én úgy vélem igenis nagy szükség van egy olyan egységes egészségügyi rendszerre amely képes kielégíteni az ügyfelek igényeit és az orvosok számára is legalább olyan transzparens felületet kínál a betegeik és a munkájuk kezelésére. Ezen problémák többségére igyekszem a webalkalmazásommal válaszolni.

# Elvégzendő munka specifikációja

## Architektúra felépítése:

Kliens-szerver architektúra

- Frontend:
  - React.js & Redux Store & React Router
  - Chakra UI komponens könyvtár, TailwindCSS, React Big Calendar
  - Jest
- Backend:
  - Laravel Sanctum API
  - PHPUnit
- Adatbázis
  - MySQL (relációs adatbázis)

Egy ilyen komolysággal rendelkező weboldal esetében érdemes kliens-szerver architektúrában gondolkodni, hogy különválasszuk a felhasználó számára megjelenített logikát a háttérben zajló eseményektől, üzleti logikától.

A frontend adja a felhasználói felület teljes egészét. Ennek értelmében a felhasználók számára megjelenített bejelentkező és regisztrációs felületek, illetőleg bármilyen olyan felület mellyel a felhasználók közvetlenül érintkezni fognak - szerepkörtől függetlenül – mind a frontend részét fogják képezni.

A backend API szerepkört lát el. Ez azt jelenti, hogy a felhasználók eseményeken keresztül érik el a háttér, azaz a szerver funkcióit úgynevezett API hívásokon (API call) keresztül. Többek között a CRUD műveletek itt kerülnek implementálásra, ahogy a frontendet ellenőrző biztonsági funkciók is.

Adatbázis szinten az adatok fizikai tárolása történik, melyek relációs táblákban lesznek elérhetőek a backend számára.

## Jogosultságok és hozzájuk tartozó funkcionalitások:

- Vendég (guest):
  - A vendég a weboldal publikus felületeit éri el: az orvosi cég leírása, orvosok listája, szolgáltatások listája, árlista.

- Páciens:
    - A már regisztrált páciens képes módosítani a saját adatait, illetőleg képes a kiválasztott szakirányon (pl. Belgyógyászat) egy általa kiválasztott orvosnál időpontot foglalni egy adott vizsgálatra. A felhasználónak szükséges megerősítenie az általa kiválasztott időpontot. Csakis megerősítés után lehet egy időpontot lefoglaltnak tekinteni.

- Orvos:
    - Az orvos bejelentkezés után az irányítópanelre, azaz dashboard-ra kerül. Kiválaszthat egy naptár nézetet melyben látszanak a lefoglalt és szabad időpontok. Az orvos továbbá képes szabad időpontokat felvenni, melyet aztán a páciensek lefoglalhatnak.

- Admin:
    - Kizárólag az adminnak van jogosultsága új orvosokat felvenni a rendszerbe, illetőleg törölni őket. Lehetősége van páciensek törlésére is.

# Használt technológiák ismertetése

Frontend technológiák:

- React: A frontend/felhasználói felület gerincét a React.js fogja adni. Ez tulajdonképpen nem egy keretrendszer hanem egy JavaScript könyvtár. A fejlesztés során reaktív komponensek fognak készülni. Ezek általános célúak azon okból kifolyólag, hogy a kódban újra lehessen használni az egyes komponenseket az igényeknek megfelelő módon. Alapvetően a React egy "SPA", tehát Single-Page Application. Ez azt jelenti, hogy képernyőket és komponenseket jelenítünk meg a felhasználóknak az oldal frissítése, újratöltés nélkül.
- Chakra UI: Ez egy komponens könyvtár amely a felhasználói felületeken felhasznált komponenseket fogja adni. A React kódjában ezek a komponensek meghívásra fognak kerülni, így gyorsabban el lehet végezni a kliens oldali építőelemek elkészítését.
- TailwindCSS: CSS keretrendszer, mely rendkívül modern és népszerű. Nekem is van vele gyakorlati munkatapasztalatom, és alkalmasnak gondolom a felhasználását. Szemben a hagyományos megközelítéssel ahol CSS fájlokat hozunk létre a TailwindCSS-t közvetlenül a HTML elemeken kell használni, például egy DIV elemen. A DIV elemeknek "class" attribútumokat adunk és ebbe soroljuk fel a TailwindCSS által definiált CSS osztályokat. Így tökéletesen személyre szabhatjuk a felhasználói felületet.
- Redux Store: A React egyik alapvető és talán legfontosabb tulajdonsága az állapotkezelés. Az alkalmazásunk JavaScript kódjában lehetőségünk van állapotok tárolására és ennek függvényében a kódunk és a felhasználói felület módosítására. Ezt a célt szolgálja a "useState" hook. Egy kisebb applikációnál nem jelent problémát ha egy adott komponensen belül definiálunk lokális állapotokat. Egy nagyméretű applikációnál, ahol esetlegesen több összefüggésnek is teljesülnie kell ezek az állapotok nagyon könnyen kezelhetetlenné válnak és nem várt működést idézhetnek elő az alkalmazásban. Célszerű egy globális helyen tárolni ezeket az állapotokat, így az állapot az alkalmazás bármely pontján lekérdezhető és esetlegesen módosítható. Ezt a célt szolgálja a Redux-Store.
- React Router: Ahhoz, hogy képesek legyünk megvalósítani a Single-Page Application-t szükségünk van egy kiegészítő könyvtárra amely kezeli a forgalomirányítást és a képernyők megjelenítését, azaz a Routing-ot. Ezt a célt szolgáltatott a React-Router kielégíteni.
- React Big Calendar: Egy olyan kiegészítő könyvtár amely lehetővé teszi egy nagy(obb) naptár integrálását az alkalmazásba.
- Jest: Egy JavaScript keretrendszer amely a kliensoldali kódnak a tesztelésére szolgál. A frontenden használt JavaScript kód helyes működésének tesztelésére használatos.

Backend technológiák:

- Laravel Sanctum: A backend API megírásához Laravel kerül felhasználásra. Ez egy PHP alapú keretrendszer mely lehetővé teszi az objektum orientált programozást. A Laravelnek a Sanctum változata az előbb is említett SPA frontend rendszerek számára kiváló ugyanis egy kis méretű de annál komplexebb autentikációs rendszert biztos.

- PHPUnit: Backend oldali teszteléshez a PHPUnit keretrendszert fogom alkalmazni, melyet alapvetően a Laravel keretrendszer biztosít, így nincs szükség a rendszer plusz integrálására, nem úgy mint a frontenden a Jest esetében. Természetesen a kód a TDD szabályainak betartásával fog implementálásra kerülni.
- MySQL: Relációs adatbázis modell mely az adatok tárolására használatos. Ehhez az adatbázishoz a Laravelnek van közvetlen hozzáférése melyet az Eloquent ORM (object relational mapper) befolyásol. Ennek segítségével az absztrakción keresztül képesek vagyunk hozzáférni az adatbázisban tárolt rekordokhoz és manipulálni azokat.

Egyéb eszközök:

- Insomnia: Egy pehelysúlyú API tesztelő eszköz, melynek segítségével az egyes API útvonalakra kéréseket lehet küldeni, így tesztelni azokat, illetve a CRUD műveleteket. Rendszerint a gyorsan elvégzendő funkcionális teszteléshez szoktam alkalmazni.
- Swagger UI: Szintén egy API tesztelő eszköz, viszont amég az Insomniát könnyű és azonnali tesztelésre tervezték pl. adatok feltöltésére lekérdezésére, addig a Swagger kódot implementálni kell a backend oldalon. Ez végeredményképp egy komplett dokumentációt eredményez amely szakmailag biztos és hiteles.
- PhpMyAdmin: A MySQL adatbázishoz biztosít egy webes interfészt, így pontosan nyomon követhető az adatbázisban lévő rekordok, illetőleg azok változásai.

Megjegyzés:

Az applikáció kliens-szerver architektúrán alapul. A kliens és a szerver kommunikációs protokollja REST API tervezési mintáját fogja követni és megvalósítani.
A fejlesztés TDD alapon kerül megvalósításra.

# A munka ütemterve

## Szeptember-október:

Elkészülnek a webalkalmazás publikus felületei. Elkészül az adatbázis terv. Kialakításra kerülnek a felhasználók jogosultsági szintjei, továbbá elérhetőek a regisztrációért és bejelentkezésért felelős weboldalak minden jogosultsági szinten.

## November-december:

Kialakításra kerülnek az autentikáció köteles felületek bejelentkezési utáni felületei. Páciens oldalon a regisztráció is működik. Az orvosok időpontokat tudnak felvenni és ezek az időpontok az adatbázisban is megjelennek.

## Január-február:

A pácienseknek megjelennek az orvosok által felvett szabad időpontok és le is tudják foglalni. Elkezdődik a dokumentáció, szakdolgozat megírása.

## Március-április:

A szakdolgozat jelentős része befejezésre kerül.

Eger, 2023.06.18

_____ _____

Hallgató Témavezető