

# **Adminisztrációs rendszer a doktori iskola számára**

*Szakdolgozati dokumentáció*

*Mogyorósi Hunor*

## Bevezetés:

A mai világ meglehetősen sok területén fel lehet használni a technológiát, mint emberi munkát segítő, megkönnyítő, vagy akár pótló tényezőt. Ilyen folyamatok közé tartozik például az adminisztrációhoz szükséges papírozások, amelyeknek mennyiségét természetesen minél jobban csökkenteni szeretné a társadalom.

A papírok helyett elektronikai eszközökön, vagy szervereken tárolt dokumentumok rendkívül sok szempontból praktikusabbá és egyszerűbbé teszi a munkánkat. Egy papírra írt szerződés helyhez kötött, bonyolultabb a szállítása, mint egy dokumentumnak, amelyet egy kattintással el tudunk juttatni egyik helyről a másikra.

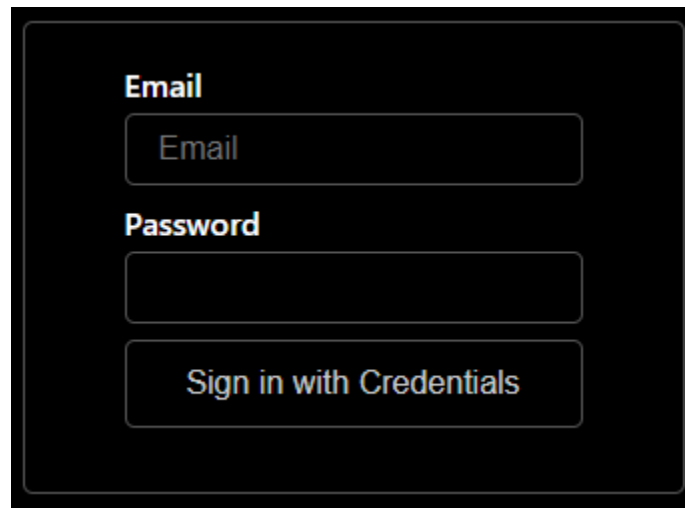
Manapság eszméletlenül sok dokumentumot lehet már tárolni egy teljesen átlagos tárhellyel rendelkező eszközön, mert nagyon kevés a helyigényük. Ezzel szemben a papír egy hosszadalmas és pazarlásokkal tele procedurán megy keresztül, amíg használatba lehet venni. A jelenlegi technológia sokféle adminisztrációs feladat digitalizációját lehetővé tette, ennek egy fajtáját készítettem én is el.

Szaktervezésként választott feladatom egy informatikai szoftver elkészítése, amely számos módon segíti és gyorsítja a doktori iskola adminisztrációs feladatainak elvégzését. A doktori iskola szereplőinek informálódási lehetősége van a tanulási ütemezés és a határidők tekintetében. A program egyik fő eleme az automatikus dokumentum-generálás, amely lehetővé teszi a felkérő levelek, meghívók, jegyzőkönyvek, hirdetmények gyorsabb és hatékonyabb kitöltését. A szoftver támogatja a komplex vizsga valamint a nyilvános védés folyamatait is.

Már középiskolás korom óta fokozottan érdekel a webfejlesztés, ami abban is megnyilvánult, hogy nyolcadikos koromban honlapot készítettem az osztályomnak. Továbbá szeretem az adminisztratív munkák optimalizására készülő programokat, mert meglehetősen sok időt, pénzt, energiát lehet egy ilyen programmal megspórolni, ezért is választottam egy ilyen feladatot. Sokszor szabadidőmben is ilyen szoftvereket fejleszték, például telefonra vagy webre. Nem mellesleg a doktori iskola minden bizonnyal nagy hasznát fogja élvezni egy online adminisztrációs rendszernek.

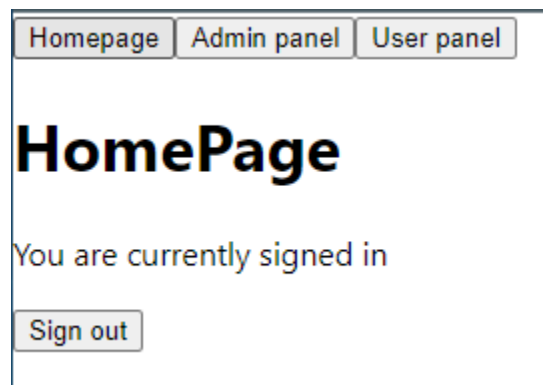
Rendszerterv:

A weblap betöltésekor a felhasználónak lehetősége van e-mail cím és jelszó megadásával bejelentkezni a felületre.

A dark-themed login form with a black background. It features two input fields: one labeled 'Email' and another labeled 'Password'. Below the password field is a button labeled 'Sign in with Credentials'. The labels and button text are in a light blue color.

(Ez az ábra a NextAuth alapvető bejelentkeztető felülete)

A felhasználói fiókok adatbázisban vannak tárolva, amelynek minden bejelentkezési próbálkozásnál kérést küld a rendszer. Ha a bejelentkezési adatok megegyeznek, akkor a felhasználó átirányítódik a bejelentkezett oldalra, ahol kiválaszthatja, hogy az admin, vagy a user felületen szeretne tevékenykedni.

A header section for a user dashboard. It contains three navigation buttons: 'Homepage', 'Admin panel', and 'User panel'. Below these buttons is a large heading 'HomePage'. Under the heading, it says 'You are currently signed in' followed by a 'Sign out' button.

A weblap kerete a user és admin felületeken alapszik, melyek jogosultságokhoz vannak kötve. Az adminisztrátorok mind a user, mind az admin panelre el tudnak navigálni, valamint létre tudnak hozni új felhasználókat (témavezetőket, hallgatókat egyaránt), a beviteli mezők és a legördülő listák megfelelő kitöltésével.

[Homepage](#) [Admin panel](#) [User panel](#)

# Admin felület

## Személy létrehozása

☐ Témavezető / oktató ☐ PhD hallgató ☐ PhD hallgató (egyéni felkészüléses)

Általa használt nyelv:

☐ magyar ☐ angol

Azonban ha nincs megfelelő jogosultság, tehát ha egy szimpla user szerepkörrel rendelkező felhasználó szeretne bejelentkezni az admin felületre, annak megtiltásra kerül ez a folyamat.

[Homepage](#) [Admin panel](#) [User panel](#)

## You dont have permission to the admin interface.

Az újonnan létrehozott felhasználó természetesen az adatbázisban tárolódik, ahol el vannak különítve a témavezetők és a hallgatók. Az adminisztrátoroknak továbbá lehetőségük van komplex vizsgát létrehozni, amelyhez a hallgató, témavezető(k), dátum, tárgyak, bizottság adatainak bevitele szükséges. A vizsgabizottsághoz új bizottsági tagot létre lehet hozni, akit természetesen hozzá lehet rendelni egy bizonyos komplex vizsgához.

## Komplex vizsga

Hallgató:

Témavezető:

A téma címe:  Dátum:  óra:  Helyszín:

alaptárgy:

melléktárgy:

[Új bizottság felvétele](#) [Felkérő levelek generálása](#) [Meghívó generálása](#)

[Jegyzőkönyv generálása](#) [HTML generálása](#)

[Mentés](#) [Mégsem](#)

Nyilvános doktori védés meghirdetésére szintén a megfelelő beviteli mevők kitöltésével van lehetősége az adminisztrátoroknak, amelyre a hallgatók jelentkezni tudnak.

A komplex vizsgáról:

A doktori iskola hallgatói számára legkésőbb a doktori tanulmányuk negyedik félévében kötelező letenniük a komplex vizsgát. Ez a vizsga egyrészt egy tantárgyi részből, másrészt egy tudományos előrehaladást bemutató részből áll. A vizsgát egy 5 főből álló bizottság értékeli, melynek összetételére vonatkozóan a doktori szabályzat rendelkezik. A Pannon Egyetem doktori szabályzata kritériumokat fogalmaz meg a komplex vizsga vonatkozásában.

A komplex vizsga megszervezése egy összetett feladatot ró annak szervezőjére, jellemzően a Doktori iskola titkárára. A vizsga előkészületeinek és lefolytatásának menetéről a Doktori Iskola komplex vizsga eljárása nyújt útmutatást. Miután a Doktori Iskola vezetője javaslatot tesz a komplex vizsga bizottságra és azt a Doktori Iskola Tanácsa elfogadta, a bizottság tagjait fel kell kérni egy-egy felkérőlevél segítségével. Pozitív válasz esetén, egy időpont-egyeztetés után, már a vizsga szervezése folyamán a vizsga helyének és idejének tudatában meghívóleveleket kell számukra készíteni, illetve az eseményt több fórumon – többek között a Doktori Iskola weboldalán is – meg kell hirdetni. Mindehhez levelek és weboldal-bejegyzések elkészítésére van szükség.

## Felhasznált technológiák:

Szakdolgozatom elkészítéséhez felhasznált technológiák közé tartozik a React, Next.js, Node.js, Firebase, HTML és a CSS. A React egy Javascript könyvtár, amelynek használatával első osztályú dinamikus weblapokat lehet készíteni, nem mellesleg a komponens alapú architektúra lehetővé teszi a projekt tökéletes átláthatóságát. E rendkívül népszerű és hasznos technológiának egy keretrendszere a Next.js, amely a React előnyeit még tovább fokozza. A Next.js egy belső API-t futtat annak érdekében, hogy egyszerűbben lehessen elérni a back-endet. További életbevágó fontosságot élvező funkciók a SSR (Server Side Rendering), SSG (Static Side Generation), Client Side Rendering (CSR), Routing, Authentication, amelyekkel e programozási nyelv kecsegtet. A NodeJS programozási technológia különböző szolgáltatások segítségével teszi lehetővé a kommunikációt az adatbázissal, az adatok mentése, módosítása és törlése érdekében. Mivel web programozásról van szó, természetesen megjelenik a HTML kód is, amelyet a Next.js függvény alapú komponensei állítanak elő. A CSS egy rendkívül modern és optimalizált verziójával, a TailwindCSS használatával készítettem el a weblap stílusát és dizájnját. A szoftver fejlesztéséhez Visual Studio Code fejlesztői környezetet használtam.

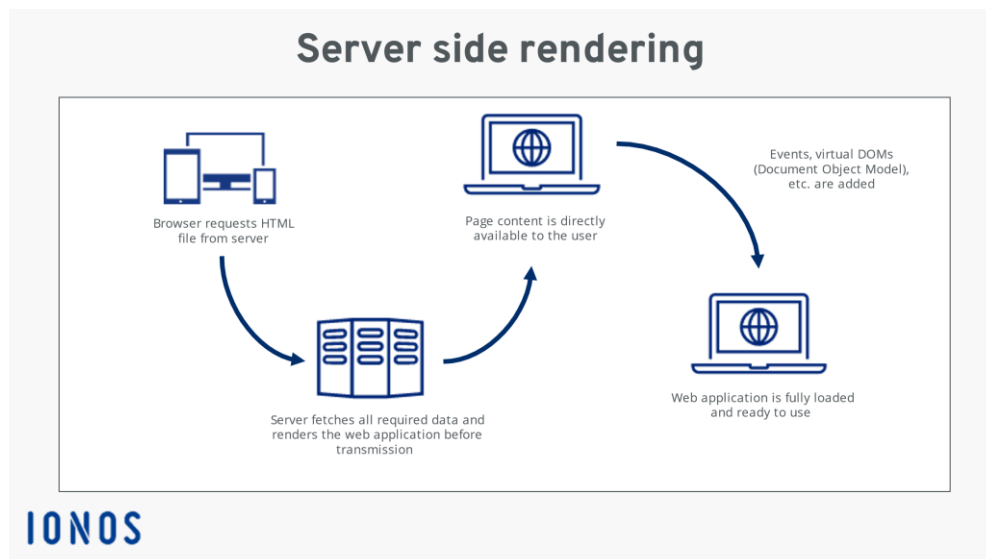
## Next.js:

A Next.js egy Vercel által létrehozott nyílt forráskódú webfejlesztési keretrendszer, amely lehetővé teszi a React alappal rendelkező webalkalmazások használatát szerveroldali rendereléssel (SSR) és statikus webhelyek generálásával (SSG).

A React egy Javascript könyvtár, amelyet hagyományosan olyan weblapok fejlesztésére használnak, amely a kliens böngészőjében renderelődik Javascript-tel. Szakdolgozatom során főként a technológia gyorsasága, tehát a weboldalak villám sebességű betöltése, és a beépített lokálisan futó szerver előnyei miatt választottam ezt a programozási nyelvet.

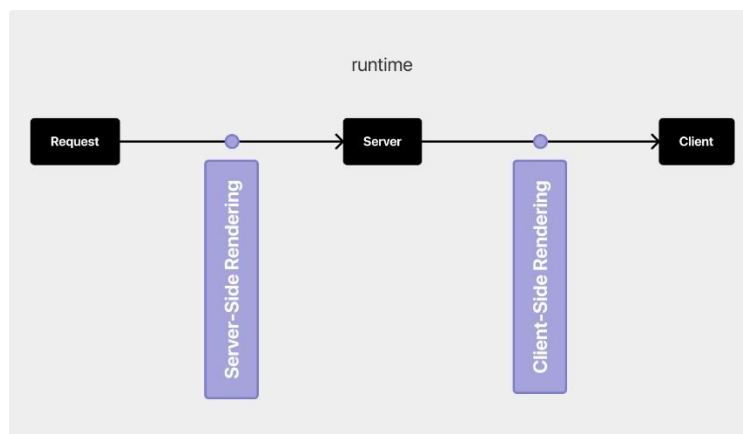
## Server Side Rendering (SSR):

A Server Side Rendering az alkalmazás azon képességére utal, hogy a weblapot a kiszolgálón jeleníti meg, nem pedig a böngészőben. Amikor egy webhely Javascript kódja megjelenik a webhely szerverén, egy már teljesen betöltött oldal kerül elküldésre az ügyfélnek, akinek a Javascript csomagja lehetővé teszi a Single Page Application keretrendszerének működését.



#### Static Side Generation (SSG):

A SSG használata olyan oldalakon ajánlott, ahol az adatokat előre le kell renderelni. Ezeket az adatokat elő lehet állítani a felhasználói kérés előtt. Mindez azt jelenti, hogy a felhasználó adatai az összeállítás időpontjában már rendelkezésre állnak. Tehát minden olyan oldalon, ahol statikus tartalmakat kell megjeleníteni, tökéletes ez a funkció használata.



#### Routing:

A Next.js routing-ja a fájlrendszeren alapszik, amely az oldalak koncepciójára épül. Tehát egy fájl könyvtárba való hozzáadása után az az oldal már automatikusan elérhető lesz útvonalként. A

„pages” mappában található fájlok a legtöbb gyakori minta meghatározására lehetőséget biztosít. Az én projektemben rendkívül sokat segített és temérdek mennyiségű programsort spórolt meg nekem ez a routing technológia.

Next Auth:

A NextAuth.js egy teljesen nyitott forráskódú autentikációs szisztéma a Next.js applikációk számára. Már a kezdetektől a Next.js és a Serverless támogatására készült. Legfontosabb előnyei: egyszerűen, rugalmasan és átláthatóan lehet kezelni; adatbázissal vagy anélkül is lehetséges a használata; rendkívül biztonságos megoldást nyújt. Mindössze néhány sorral meg lehet oldani a felhasználók Google/Facebook/Email (stb...) alapú hitelesítését.

Client Side Rendering: A CSR adatlekérés olyan esetekben tud hasznos lenni, ha a weboldal nem igényel SEO (keresőmotor) indexelést, ha nem igényelt az adatok előzetes megjelenítése, illetve amikor az oldalak gyakori frissítése életbevágó.

Node.js:

A Node.js egy szoftverrendszer, melyet skálázható internetes alkalmazások, mégpedig webszerverek, vagy webapi-k elkészítésére használnak. A programokat teljesen egyszerűen Javascript-ben kell írni. Létezik eseményalapú, aszinkron I/O, amely segítségével a túlterhelés alighanem szóba sem jöhet, és a skálázhatóság prioritása pedig előtérbe van helyezve. A Node.js cross-platform, tehát Windows, Linux, Unix és macOS rendszereken fut.

A fejlesztőknek lehetőséget biztosít a Javascript-ben történő parancssori eszközök írására és a szerver oldali szkriptek létrehozására. A szerver oldali szkriptek a weboldalnak dinamikus tartalmakat produkálnak, amelyek azelőtt futnak le, mielőtt az oldal a felhasználóhoz elküldésre kerülne.

HTML:

A HTML (HyperText Markup Language) a szabványos programozási nyelv a dokumentumok böngészőben való megjelenítésére. Design-beli segítséget nyújthatnak olyan technológiák, mint például a Cascading Style Sheet (CSS), amelyet a későbbiekben bemutatok, és szkriptnyelvek, mint például a Javascript.



A böngészők egy webszerverről vagy tárolóról kapott dokumentumokat alakítanak át multimédiás weboldallakká.

A HTML oldalak HTML elemekből állnak, amelyek segítségével képek, beviteli mezők, gombok és egyéb objektumok, például interaktív keretben megjelenő űrlapok beágyazására van lehetőség. A HTML-lel létre lehet hozni strukturált dokumentumokat akár a szöveg struktúrális szemantikája alapján is. A HTML elemeket címkék (<, >, /) jelölik.

A HTML olyan programok beágyazására képes, amelyek szkriptnyelven, például Javascript-ben íródnak (a <script> tag segítségével), ami jelentősen pozitív irányba tudja változtatni a weboldalak viselkedését és dinamikáját.

### CSS:

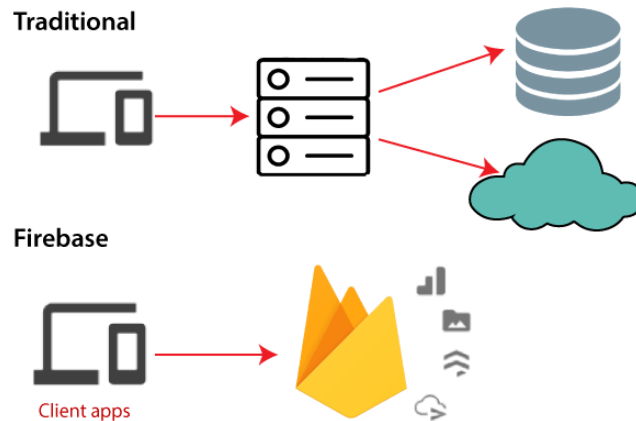
A CSS (Cascading Style Sheets) egy stílusnyelv, amely HTML-ben vagy XML-ben írt dokumentumok kinézetét írja le. Elengedhetetlen építőeleme a World Wide Web-nek a HTML és a Javascript mellett.

### Tailwind CSS:

A Tailwind CSS egy olyan verziója a CSS-nek, amely egycélú segédosztályokat biztosít, melyeket közvetlenül a HTML-ben vagy XML-ben használni lehet a design megtervezéséhez. Ezáltal sokkal egyszerűbb módon el lehet érni ugyanazt a kívánt kinézetet, vagy elrendezést.

### Firebase:

A Firebase egy tárhelyet szolgáltató megvalósítás, amelyet bármilyen típusú alkalmazáshoz lehet használni (iOS, Android, Javascript, Node.js, Java, Unity, PHP, C++ stb.). NoSQL rendszerének köszönhetően valós idejű tárolást nyújt az adatbázisoknak, tartalmaknak, közösségi oldalak által történő autentikációnak illetve értesítéseknek és kommunikációs szervereknek.



Hasonló megvalósítások:

Szakdolgozatom témájához közvetlenül kötődő rendszer megvalósítás nem létezik, azonban hasonló logikát lehet észrevenni a doktori iskola adminisztrációs rendszere és egy neptunban történő szakdolgozati jelentkezés, vagy egy google dokumentumban vezetett adatok rendszere között. Abból fakadóan, hogy az én szoftverem a doktori iskola adminisztrációjára van specializálódva, sokkal kötöttebb, mint az előbb felsorolt módszerek, de emiatt meglehetősen bonyolultabb is a megvalósítása, használata.

Források:

<https://en.wikipedia.org>

<https://nextjs.org>

<https://next-auth.js.org>

<https://static.javatpoint.com/tutorial/firebase/images/firebase-introduction.png>

[https://www.ionos.com/digitalguide/fileadmin/DigitalGuide/Screenshots\\_2022/Server-side-rendering-diagram.png](https://www.ionos.com/digitalguide/fileadmin/DigitalGuide/Screenshots_2022/Server-side-rendering-diagram.png)

<https://pawelgrzybek.com/photos/2020-12-19-1.jpg>

