

Blog weboldal – dokumentáció

Készítette: Türk Viktor, F5HV4G

1. Bevezető

Ez az oldal azért jött létre, hogy a saját gondolataim leírjam, megosszam illetve a saját hobbijaimhoz tartozó információkat amik relevánsok számomra megtaláljam egy helyen, például a Fyr'alath oldal is pont erre szolgál, egyfajta automatizálása a dolgoknak majd megjelenítése egy weben elérhető felületen. Így akár ha úton vagyok is könnyen elérek bizonyos információkat.

A weblapon 5 oldal található meg:

- Home
- Posts
- Social Credit
- Fyralath
- Review

Ezek mind egy oldalsó menü segítségével érhetők el ahol kiemelve lehet látni az aktuális oldalt.

2. Főoldal

A főoldalon, vagy "Home" szekcióban rövid áttekintést adok az aktuális tanulmányaimról és szakirányomról. Továbbá kiemlem azokat a programozási nyelveket, keretrendszereket és szoftvereket, amelyekkel jelenleg foglalkozom. A tartózkodási helyem is fel van tüntetve, azonban kizárólag városi szinten, tekintettel a személyes adatvédelem fontosságára, és annak elkerülésére, hogy bárki számára könnyen hozzáférhető legyen a pontos lakcímem.

Végül, de nem utolsósorban, pár mondatban kifejtem a blog létrejöttének indítékait és céljait. Ezzel lehetőséget teremtek arra, hogy az olvasók megismerhessenek engem és jobban megértsék a blog hátterét. Azt is megvilágítom, hogy milyen tartalmakra számíthatnak az oldalon, és milyen értékeket szeretnék közvetíteni azáltal.

3. Posts

A második oldalon „Posts” lehet megtalálni az aktuális blogposztjaimat lehet látni amit egy GO backend szolgál ki, a backendről lekért adatot egy CustomMDX komponens dolgozza fel.

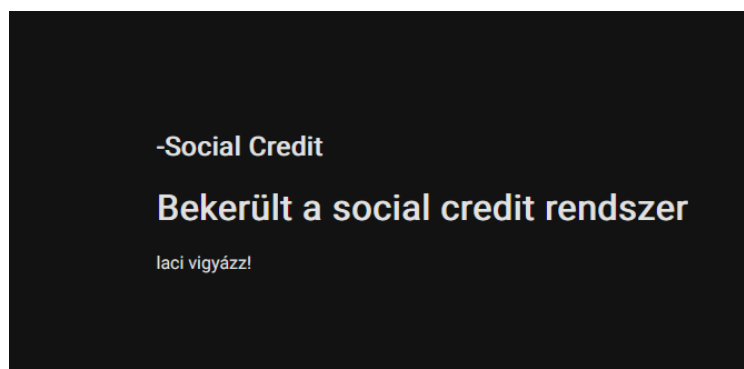
```
import { MDXRemote } from 'next-mdx-remote/rsc'

const components = {
  h1: (props: any) => (
    <h1 {...props} className="large-text">
      {props.children}
    </h1>
  ),
}

export function CustomMDX(props: any) {
  return (
    <MDXRemote
      {...props}
      components={{ ...components, ...(props.components || {}) }}
    />
  )
}
```

Ez annyit csinál, hogy az adatbázisban Stringként tárolt Markdown fájlokat fordítja le így jelenik meg a látogató számára. Ezeket a blog posztokat relatív könnyű szerkeszteni és újakat feltölteni azonban egy blog poszt írását segítő modul még nem készült el.

```
{
  "_id": {"_id"},
  "title": "socialcredit",
  "content": "-Social Credit\n ---\n # Bekerült a social credit rendszer\n laci vigyázz!"
}
```



4. Social Credit

Ezen az oldalon egy táblázat fogadja a látogatót, amelyet a barátaimmal alakítottunk a Kínában működő társadalmi kreditrendszerre alapozva. Sokat szoktunk játszani a Rainbow Six Siege nevezetű játékkal és azért hoztuk ezt létre, hogy ha valaki jól teljesít akkor plusz pontot kap, ha rosszul akkor mínusz pontot és így tudjuk követni mindenkinek a képességeit, illetve nagyon jókat poénkedünk vele amikor találkozunk. A táblázat alatt megtalálható, hogy mikor ki mennyi pontot kapott (vagy éppen veszített el), ezzel tisztán lehet látni a teljesítményét minden embernek. Ezek a frissítések ugyanúgy a CustomMDX Komponens segítségével jelennek meg.

2023-11-20

Kevin büntetése

- Kevin nem tud olvasni, és hülyeségeket kérdezget
- -1 pont

2023-11-20

Laci büntetése

- Laci azt mondta ma nem fog R6ozni és ne számítsunk rá de ő mégis R6ozni akart
- -1 pont

2023-11-20

Indulás

- Elindult a leaderboard
- Mindenki 0 ponttal kezd kivéve kristófot mert lefizetett
- A szabályok hamarosan érkeznek

R6 Social Credit Leaderboard

Name	Credit
kristóf	16
kevin	7
roli	7
gabi	4
máté	4
viktor	2
laci	-2














5. Fyralath

Ezen az oldalon ismét egy táblázat található ami a World of Warcraft nevezetű játékban egy ritka legendás baltának a létrehozásához szükséges alapanyagoknak az árát kéri le a fejlesztő cég (Blizzard) API-járól, majd a szükséges darabszámokkal összeszorozza az egyes alapanyagok árait majd a végén összeszummázza, hogy lehessen látni mennyibe is kerül ha szeretnék egy ilyen baltát.

Így nem kell egyesével külön külön kiszámolgatni mindent minden alkalommal amikor kíváncsiak vagyunk rá.

Óránként frissül, ez egy korlátozás a Blizzard felől.

A táblázat alatt lehet látni mikor frissült utoljára.

Name	Price	Needed	Total
 WoW Token	20€	0.4029	8.06€
 Fyr'alath the Dreamrender	139498.66999999998	1	139498.66999999998
 Zaralek Glowspores	9	400	3600.00
 Obsidian Cobraskin	3499.99	3	10499.97
 Dreaming Essence	75	5	375.00
 Mireslush Hide	90.87	50	4543.50
 Shadowflame Essence	3620.87	10	36208.70
 Cosmic Ink	74.24	250	18560.00
 Runed Writhe bark	84.99	50	4249.50
 Awakened Fire	80	150	12000.00
 Awakened Earth	99	100	9900.00
 Awakened Order	209.08	50	10454.00
 Resonant Crystal	145.54	200	29108.00

updates hourly when blizzard api updates

last updated: 2024-01-29 10:00:06 UTC

6. Review

Ezen az oldalon egy űrlap fogadja a látogatót ahol kitudja fejteni a véleményét az oldalról, hogy tetszett-e neki vagy sem.

Összesen 7 darab mező található meg az űrlapon:

Name, Email, Birthdate, Message, Rating, Favorite color, Terms and conditions

Ezek közül mindegyiket kikell töltenie a látogatónak illetve elkell fogadni az Általános szerződési feltételeket.



Please review the site:

Name:

Email:

Birthdate: 

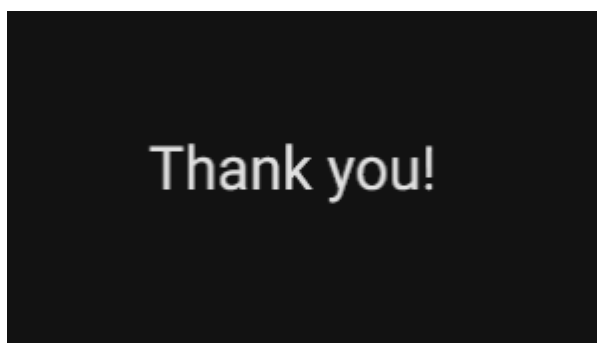
Message: (0/255)

Rating: ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Select your favorite color:

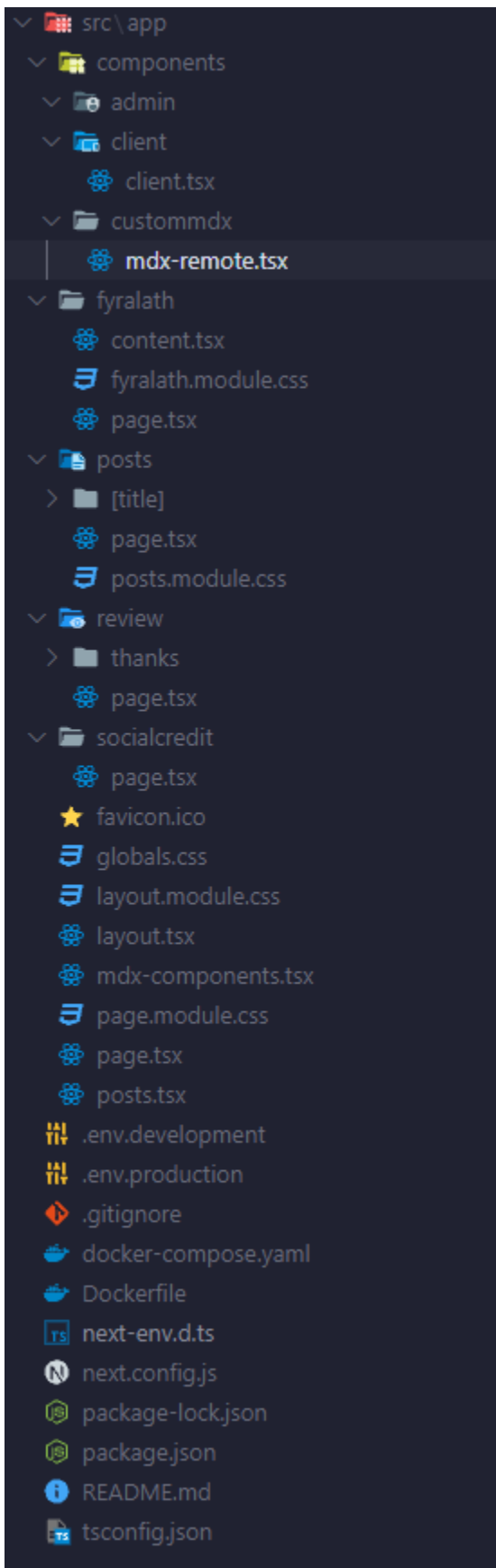
Terms and conditions ☐

Az űrlap kitöltése után egy animált forgó „Thank you!” felirat várja a kitöltőket.



7. Technikai részletek

Az oldal elkészítéséhez a NextJS Keretrendszert használtam segítségül.



A mappa struktúrája:

Az alap src/app mappában megtalálható elősorban a page.tsx ami a főoldal tartalmát foglalja magába.

Ezután a 2. legfontosabb fájl a layout.tsx ami az összes oldalon megjeleníti az oldalsó menüt.

A layout.module.css tartalmazza az oldalon megtalálható összes komponensnek a stílusait.

A global.css-ben található a háttér színezése.

A posts.tsxben található a Posts komponens ami felelős az összes blogposzt megjelenítéséért, egyfajta szerver komponensként működik, ami lekéri az adatok a backendről, majd továbbadja a ClientPosts komponensnek a posztokat, ami egy kliens komponens ami megjeleníti a blogposztokat egy 2. oldalsó menüben ami az 1. oldalsó menüből ágyazódik ki.

A posts almappában található egy page.tsx ami semmit nem jelenít, csak azért létezik, hogy ebből lehessen egy úgynevezett slugot létrehozni, a page.tsx mellett lévő [title] mappa egy slug ami arra szolgál, hogy ha a trkviktor.com/posts/socialcredit oldalra megyünk akkor kiveszi a socialcreditet a posts/ után az útvonalból és az alapján jeleníti meg az adott blogposztot.

Feltűnhet, hogy a page.tsx sokszor szerepel, ez a NextJS-nek a szabványa, csak így lehet a dinamikus útvonalakat működésre bírni ebben a keretrendszerben.

Megtalálható még a fyralth és a review mappa, ezek az adott tartalmat

jelenítik meg amiket a dokumentáció során már említettem.

Végül de nem utolsósorban a client mappában található a client.tsx ami

ténylegesen tartalmazza az összes kliens komponenst ebbe beletartozik a két oldalsó menü.

Erre a külön kliens komponenseket tartalmazó fájlra azért van szükség mert a NextJS-ben kétféle típusú komponens létezik. Egy szerver komponens és egy kliens komponens. Alapjába véve minden komponens egy szerver komponens, a probléma ezzel az, hogy a szerver komponensekben nem lehet semmi hookot, statet használni ezért ez csupán adatok lekérdezésére szolgál a backendről. Egy szerver komponens általában a lekért adatokat továbbítja egy kliens komponensnek ahol feldolgozva a kapott adatok szépen lehet vele dolgozni, tartalmat megjeleníteni.