TheOld – Crafters szervercsoport weboldala

FullStack webapplikáció a TheOld – Crafters szervercsoport számára. (https://oldcrafters.net)

Stack:

- NGINX
- PostgreSQL
- Django (API: DRF)
- React (Vite + TS)
- Docker konténerizált környezetben

Az alkalmazás szolgáltatásai Docker konténerekben futnak, ezeknek kezelését a Docker Compose program végzi. (A backend konténere megtalálható a Docker Hubon)

A program célja (kliens elvárásai)

A weboldalnak keresőmotor-optimalizáltnak kell lennie, igényes, letisztult stílussal kell prezentálnia a megbízó termékét, alkalmasnak kell lennie a blogbejegyzések, dinamikus oldalak létrehozására, szerkesztésére és stilizálására webszerkesztő ismeretek nélkül is.

Egy hitelesítéssel védett adminisztrátori felületen szerkeszthetőek a főoldalon megjelenő hírek, illetve a dinamikus navigációs sávba megjelenítésre kerül az itt készített oldalak is, amelyekhez egyedi elérési út is definiálható.

Backend

Megemlítendő csomagok

- Django
- Pillow (képek feldolgozásához)
- Django TinyMCE (az admin felületen a hírek törzsének szerkesztéséhez használt beépülő Rich-Text szövegszerkesztő)
- Django Summernote (az admin felületen az oldalak törzsének szerkesztéséhez használt beépülő Rich-Text szövegszerkesztő, amin keresztül fel lehet tölteni, és be lehet illeszteni médiafájlokat, illetve az általa generált HTML kód is szerkeszthető)
- Daphne (ASGI webszerver applikáció)
- Twisted (HTTP2 integráció a könnyebb kommunikációhoz)

Adatbázis

Az adatbázisról részletes dokumentáció található meg itt, ami a dbdocs programmal készült. A táblákról készült vizuális modell a web/backend/database_model.pdf állományban található.

Az adatbázis műveleteket a Django kezeli, az SQL lekérdezéseket a backendben definiált modell műveletek alapján generálja automatikusan.

Az adatbázis szerver eléréséhez szükséges adatok a docker-compose.yaml állományban módosíthatóak.

Admin felület

A Django beépített adminisztrációs felülete a szintén beépített hitelesítéssel, kiegészítve néhány módosítással a gördülékenyebb felhasználói élmény érdekében.

A főoldalon megjelenő hírek, ahhoz tartozó kategóriák és a navigációs sávban megjelenő oldalak szerkesztésére itt kerülhet sor. Emellett újabb adminisztrátorokat és csoportokat lehet felvenni, amelyeknek jogköröket is meghatározhatunk. A Django Summernote szerkesztő által feltöltött médiafájlokat is itt kezelhetjük.

A hírek és oldalak szűrhetőek bizonyos szempontok szerint, és a felhasználók által végzett műveletek listája is megtekinthető.

A híreknél feltölthető elemenként egy kép, amit feltöltéskor WEBP formátumba konvertál, majd lecsökkenti a méretét és minőségét a kevesebb tárolókapacitás igény és a frontenden történő gyorsabb betöltés érdekében.

API

A Django REST Framework (DRF) segítségével generált API-on keresztül érhető el az adatbázisba felvett hírek és oldalak. Az API-on keresztül az adatok nem módisíthatóak, csak lekérdezés lehetséges.

A HTTP kéréstől függően a válasz vagy egy HTML oldal, amelyen prezentálva van az API endpoint leírása, illetve a válaszul kapott adatok formázva, vagy az egyszerű JSON formátumú adatok.

Az adatbázisból lekérdezett adatok a DRF által biztosított szérializálók által JSON formátumban kerül elküldésre. A /news /navpages /categorys endpointokon az elemek listázása történik.
A /navpages/:id endpointon az oldal törzse is lekérdezhető. Ez a listázásra adott válasz csökkentése érdekében csak itt érhető el. A hírekre lapszámozás (pagination) van alkalmazva, egy oldalon 10, max 50 hír jeleníthető meg.

Frontend

A frontend a Vite, TypeScript, React és TailwindCSS technológiákkal készült.

A TypeScript adta lehetőségeket kihasználva és saját típusok létrehozásával áttekinthetőbb a kód, a beszédes változó és eljárásneveknek köszönhetően dokumentáció nélkül is érthető a kód.(kontextus)

A webszerver minden kérést ugyan arra a HTML oldalra irányít, ami a React Router program segítségével a kérés URL-je alapján eltérő tartalmat jelenít meg.

API kérés

Az oldal betöltése alatt kérést küld az API-nak, amiben lekérdezi a navigációs sávba kerülő oldalakat és a főoldalon a híreket, majd megjeleníti ezeket. A navigációban a választól függően kerülhet oldal az „Egyéb” nevű legördülő menübe is. A főoldalon egyszerre 10 hír jelenik meg, a böngésző URL mezőjében a „?page=<szám>” kereséssel lapozhatunk a következő oldalra. (A frontenden történő lapozás még nincs implementálva.)

CSS preprocessor és framework

A TailwindCSS segítségével osztálynevek meghatározásával lehet stílusokat alkalmazni a dokumentumra. A build során egy olyan minimalizált CSS állományt hoz létre, amiben csak a szükséges definíciók vannak, ezzel csökkentve a hálózati forgalmat az oldal látogatásakor.

Keresőmotor optimalizálás és hozzáféhetőség

A navigációs mezők el vannak látva a megfelelő „aria” jelzőkkel, a képek alternatív szöveggel, a HTML5-ös szemantikai elemek megfelelően vannak használva, így a képernyőolvasók és egyéb kisegítő lehetőségek könnyebben tudnak navigálni az oldalon.

A megfelelő „meta” címkék alkalmazásával a keresőmotorok könnyebben tudják beazonosítani az oldalt szerepét és a tartalmát könnyebben tudják kategorizálni, így nagyobb eséllyel kerül előre a keresési találatok között.

Egyéb

NGINX

A Docker Hub-on található legfrissebb képfájlt használja a program bármiféle változtatás nélkül. A konfigurációs fájlja a „config/nginx.nossl.conf” állományban található. Gyors, biztonságos, megbízható, könnyen konfigurálható webszerver, fordított proxy és terhelés elosztó. A statikus fájlok szolgáltatása, jövőbeli HTTPS támogatás, biztonság és fordított proxyzás a feladata jelen esetben. A kéréseket az index.html filera irányítja, a /static és /media utakra érkező kérésekre statikus fileokkal válaszol, a /admin és /api kéréseket pedig továbbítja a backendnek. Effektív tömörítéssel csökkenti a hálózati forgalmat és az oldal betöltésének idejét.

Egyéb mappák, amelyek futáskor jönnek létre

A db.postgres.data mappában az adatbázis fájljai találhatóak.

A logs mappában pedig az NGINX hozzáférési és hiba naplófájljai, illetve a Daphne hozzáférési naplófájlja található.

Jövőbeli tervek

- A GitHub Actions segítségével automatizálni a backend Docker képfájljának buildelését, illetve a frontend buildelését is
- Implementálni a frontenden a főoldalon a hírek oldalai közötti lapozást

- Implementálni a backenden egy Discord botot, ami a TOC Discord szerverére is kiírja az admin felületen felvett híreket
- 404 és egyéb hibák kezelése mind a front mind a backenden
- Részletesebb kód dokumentáció készítése
- HTTPS implementálása (Let's Encrypt)