Záródolgozat

Tartalom

Fejlesztői környezet

Bevezetés

A fejlesztői környezetünk a modern webes alkalmazások építésére fókuszál, ötvözve a megbízható technológiákat és a legújabb trendeket a front- és backend fejlesztés terén. A backend részleg alapját a Visual Studio biztosítja, amely ASP.NET technológiával van párosítva, így erős és skálázható szerveroldali logikát kínálunk. A Visual Studio Code-ot választottuk a React alapú frontend fejlesztésére, mely gyors és interaktív felhasználói felületek készítését teszi lehetővé.

Az adatkezelés terén a PostgreSQL-t részesítjük előnyben a MySQL-lel szemben, amely robusztus és megbízható adatbáziskezelő rendszerként szolgál projektjeink számára. Jobban megfelel a komplex adatstruktúrák kezelésére, jobb tranzakciókezelést és fejlettebb adattípusokat kínál. Például a PostgreSQL támogatja a JSON adattípust, amely lehetővé teszi a rugalmas és strukturálatlan adatok hatékony tárolását és lekérdezését, míg a MySQL-ben ez kevésbé intuitív.

A .NET keretrendszert használjuk a backend fejlesztéséhez, amely széles körű funkcionalitást és teljesítményt kínál, míg az ASP.NET és annak modern változata, az ASP.NET Core, különösen alkalmasak webes API-k és alkalmazások fejlesztésére.

A frontend területén a Vite szolgál fejlesztői szerverként, amely rendkívül gyors újratöltést és hatékony modulbetöltést kínál a React projektekben. A TypeScript az előnyben részesített nyelv, mivel szigorúbb típusellenőrzést és jobb fejlesztői élményt nyújt, mint a JavaScript. A Reactot a Tailwind CSS-sel párosítjuk, amely egy modern CSS keretrendszer, preferálva azt a Bootstrap helyett az egyedi és reszponzív design megvalósításához.

Visual Studio

A Visual Studio egy kiemelkedő fejlesztői környezet, amelyet kifejezetten a Microsoft által kínált technológiák, így az ASP.NET alapú webes és vállalati alkalmazások fejlesztésére terveztek. Az ASP.NET projektjeink fejlesztéséhez a Visual Studio biztosítja azt a rugalmasságot és a teljes körű támogatást, amire szükségünk van a hatékony munkavégzéshez. Az IDE integrált környezete lehetővé teszi számunkra, hogy a kódírást, a hibakeresést, az alkalmazások tesztelését, és a telepítést egyetlen, jól összehangolt felületen végezzük.

Az ASP.NET projektek fejlesztésének egyik kulcsfontosságú aspektusa a NuGet csomagkezelő integrációja, amely lehetővé teszi számunkra, hogy könnyen kezeljük a projekt függőségeit és beilleszthessük a legújabb könyvtárakat vagy keretrendszereket a projektekbe. A NuGet segítségével automatikusan kezelhetjük a csomagfrissítéseket, ami jelentősen csökkenti a függőségekkel kapcsolatos problémák kockázatát, és egyszerűsíti a különböző környezetek közötti konzisztencia fenntartását. Innen telepítettük a szükséges csomagokat, mint például az Entity Framework Core-t, amely az adatbázis-kezeléshez szükséges keretrendszer.

Visual Studio Code

A Visual Studio Code (VS Code) az egyik legelőnyösebb fejlesztői környezetünk a React alapú frontend projektekhez. Ennek a könnyű, mégis erőteljes forráskód-szerkesztőnek a kiválasztása elsősorban a JavaScript és TypeScript, két olyan nyelv támogatása miatt történt, amelyek létfontosságúak a React fejlesztésben. A VS Code különösen hasznos funkciókat kínál a React fejlesztők számára, mint például az intelligens kódkiegészítés, a komponensek közötti gyors navigáció és az integrált hibakeresés, ami jelentősen felgyorsítja a fejlesztési folyamatot és javítja a kód minőségét.

A fejlesztői környezetünk további testreszabását és optimalizálását számos, kifejezetten a React és a modern webfejlesztési munkafolyamatokhoz tervezett VS Code kiterjesztés segíti. Ezek közé tartoznak a linter-ek és kódformázók, mint például az ESLint és a Prettier, amelyek segítenek fenntartani a kódbázis konzisztenciáját és olvashatóságát. Emellett a React specifikus kiterjesztések, mint a React Code Snippets, tovább egyszerűsítik a gyakori minták és komponensek kódolását. Ezek a kiterjesztések jelentősen hozzájárulnak a fejlesztési hatékonysághoz.

.NET

A .NET keretrendszer egy átfogó fejlesztési platform a Microsofttól, amely lehetővé teszi a fejlesztők számára, hogy különféle típusú alkalmazásokat hozzanak létre, beleértve a webes, asztali, mobil-, játék-, és IoT-alkalmazásokat. A platform nyelvfüggetlen, ami azt jelenti, hogy támogatja a különböző programozási nyelveket, mint például a C#, F# és Visual Basic. A .NET keretrendszer különösen erős a vállalati szintű webalkalmazások fejlesztésében, köszönhetően az ASP.NET-nek, egy modell-nézet-vezérlő (MVC) architektúrát alkalmazó keretrendszernek, amely lehetővé teszi a dinamikus weboldalak és szolgáltatások kifejlesztését.

Mi, különösen a C# nyelvet részesítjük előnyben a .NET keretrendszer használatakor, mivel ez a nyelv kifejezetten a .NET-hez lett tervezve. A C# egy objektumorientált programozási nyelv, amely erős típusosságot, memória kezelést, és számos modern programozási paradigma támogatását kínálja, ami lehetővé teszi a fejlesztők számára, hogy biztonságos, hatékony, és könnyen karbantartható kódot írjanak. A C# nyelv szintaxisa egyszerű és könnyen érthető, ami csökkenti a tanulási görbét, és gyorsítja a fejlesztési folyamatot.

ASP.NET, ASP.NET Core

Az ASP.NET egy erőteljes webfejlesztési keretrendszer a Microsofttól, amely lehetővé teszi a fejlesztők számára, hogy dinamikus weboldalakat, alkalmazásokat és szolgáltatásokat hozzanak létre. Az ASP.NET Core, az ASP.NET modern, keresztplatformos, nagy teljesítményű változata, kifejezetten arra tervezték, hogy könnyen kezelhető és skálázható webalkalmazásokat lehetővé tegyen a .NET Core futtatási környezeten. Az ASP.NET Core kínálja az ASP.NET összes előnyét, miközben további előnyöket biztosít, mint például a keresztplatformos támogatás, a könnyebb konfiguráció, valamint a jobb teljesítmény.

Mi az ASP.NET Core-t részesítjük előnyben webfejlesztési projektünkben, különösen az Entity Framework Core integrációjával együtt, ami egy erőteljes és rugalmas objektum-relációs leképező (ORM) keretrendszer. Az Entity Framework Core lehetővé teszi számunkra, hogy adatmodelljeinket közvetlenül C# osztályokban definiáljuk, és az adatbázisműveleteket magas szintű API-k segítségével hajtsuk végre, anélkül, hogy közvetlenül SQL kódot kellene írnunk. Ez nemcsak a fejlesztési folyamatot gyorsítja fel, hanem javítja az alkalmazás karbantarthatóságát is, mivel a kódbázis egyszerűbb és tisztább marad.

GitHub

Projektünkben a GitHubot használjuk a forráskód tárolására, valamint a fejlesztési munkafolyamatok, mint a hibajavítás, funkciófejlesztés és automatizált telepítések kezelésére. A GitHub Actions, egy kulcsfontosságú szolgáltatás a GitHub platformon, lehetővé teszi számunkra, hogy automatizált workflowokat hozzunk létre, amelyek a kódbázisunkkal kapcsolatos eseményekre, mint például a push műveletekre vagy a pull requestekre reagálva aktiválódnak.

A GitHub Actions, egy kiemelt szolgáltatás a GitHubon, lehetővé teszi számunkra, hogy automatizált munkafolyamatokat állítsunk be, amelyek különböző eseményekre reagálva aktiválódnak, mint a forráskódhoz való hozzáadás (push) vagy a pull requestek.

A frontend automatikus telepítését a Cloudflare Pages-re, míg a backend automatikus telepítését a Fly.io-ra konfiguráltuk a GitHub Actions segítségével. Ez azt jelenti, hogy minden változás, amely a frontend vagy a backend kódjában történik, és egy adott branch-be kerül (például egy feature branch merge-elése a main branch-be), aktivál egy workflow-t, amely automatikusan teszteli, építi, és telepíti az alkalmazásokat az előre megadott platformokra. A Cloudflare Pages ideális választás a statikus frontend alkalmazások gyors és biztonságos telepítésére, míg a Fly.io kiválóan alkalmas a backend API-k nagy rendelkezésre állású és skálázható telepítésére.

JWT

Projektünkben a JWT-kat használjuk a felhasználók hitelesítésére, ami lehetővé teszi számunkra, hogy biztonságos és hatékony hozzáférést biztosítsunk az alkalmazásainkhoz. Egy JWT három részből áll: a fejlécből (header), az adatokból (payload), és az aláírásból (signature), amelyeket pontokkal választanak el egymástól. A fejléc általában tartalmazza a token típusát, például JWT, és az aláíráshoz használt algoritmust, például HMAC SHA256 vagy RSA.

Projektünkben a JWT frissítéséhez egy refresh tokent használunk, amely lehetővé teszi a felhasználók számára, hogy új hitelesítési tokent kérjenek anélkül, hogy újra be kelljen jelentkezniük. Ha a frontend alkalmazás egy nem engedélyezett választ kap, automatikusan használja a refresh tokent egy új JWT kéréséhez. Ez a megközelítés javítja a felhasználói élményt, mivel a felhasználóknak nem kell gyakran újra bejelentkezniük, miközben fenntartja a rendszer biztonságát azáltal, hogy rendszeresen frissíti a hitelesítési tokeneket.

Vite, TypeScript

A Vite és a TypeScript kombinációja jelentősen felgyorsítja és optimalizálja a frontend fejlesztési folyamatot projektünkben.

A Vite, egy modern építőeszköz, amely a JavaScript modulok natív ES import export szintaxisát használja a böngészőkben, lehetővé teszi az alkalmazások villámgyors indítását és frissítését fejlesztési időben. Ez ellentétben áll a hagyományosabb eszközökkel, mint például a Create-React-App (CRA), amely hajlamos lehet lassabb indítási időkre és frissítésekre, különösen nagyobb méretű projektjeinknél. A Vite által nyújtott azonnali modulfrissítés (HMR) és a konfiguráció nélküli indulás tovább csökkenti a fejlesztési ciklusokat, lehetővé téve a fejlesztők számára, hogy gyorsabban iteráljanak és teszteljenek.

A TypeScript, egy JavaScriptre épülő nyelv, statikus típusellenőrzést ad hozzá a dinamikus JavaScript nyelvhez. A projektünkben a TypeScriptet preferáljuk a JavaScripttel szemben, mivel a statikus típusellenőrzés javítja a kódbiztonságot, elősegíti a hibák korai szakaszban történő azonosítását, és növeli a fejlesztési folyamat hatékonyságát. A TypeScript támogatása az intelligens kódkiegészítéshez, refaktoráláshoz és a jobb kódértelmezéshez vezet, ami csökkenti a fejlesztési időt és javítja a kódbázis olvashatóságát és karbantarthatóságát.

A Vite használata a TypeScripttel különösen előnyös, mivel a Vite kihasználja a TypeScript gyors, inkrementális fordítását, ami lehetővé teszi, hogy a fejlesztők azonnal láthassák a kódjukban végrehajtott változtatások hatását. Ez a kombináció egyaránt támogatja a gyors prototípuskészítést és a nagy teljesítményű alkalmazásfejlesztést, miközben biztosítja a kód minőségét és robusztusságát.

React, Tailwind CSS

A React lehetővé teszi a dinamikus és interaktív felhasználói felületek hatékony építését a weben. Ennek a könyvtárnak az alkalmazása számos előnyt kínál, amelyek közvetlenül hozzájárulnak a fejlesztési folyamat javításához és a végtermék minőségének növeléséhez.

A Bootstrap-tel szemben, amely előre meghatározott stílusú komponenseket és JavaScript plugineket kínál, a Tailwind CSS egy “utility-first” megközelítést alkalmaz, ami azt jelenti, hogy kis, újrafelhasználható stílusosztályokat biztosít, amelyeket közvetlenül a HTML elemekhez rendelhetünk. Ez a megközelítés nagyobb mértékű testreszabhatóságot és finomhangolást tesz lehetővé, anélkül, hogy aggodalmat kellene érezni a felesleges CSS kód vagy stílusütközések miatt. Ezenkívül a Tailwind CSS segítségével könnyen létrehozhatók reszponzív designok, mivel a keretrendszer mobil-first megközelítést követ és rendelkezik számos reszponzív segédosztállyal.

Egy másik fontos szempont, amiért a Tailwind CSS-t preferáljuk a Bootstrap-pel szemben, az a teljesítmény. A Tailwind lehetővé teszi, hogy csak azokat a stílusokat építsük be, amelyeket ténylegesen használunk, csökkentve ezzel az alkalmazásunk végső CSS méretét. Ez a “purge” (tisztítási) funkció jelentősen optimalizálja a betöltési időket, különösen nagyobb projektjeink esetében.

Felhasználói dokumentáció

Bevezetés

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.