**포트폴리오**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 프로젝트명 | .net을 활용한 웹사이트 제작 | 기간 | | 2024.06.22 (6주) |
| 프로젝트 개요 | C# ASP.Net Core API 와 React 및 Azure를 활용한 쇼핑몰 구축 | | | |
| 구축 환경 | Window, Visual Studio, SSMS, Git, npm, Azure, VisualStudio | | | |
| 주요 사용 기술 | C# ASP.NET Core API, EF core, JWT, Swagger, React, TypeScript, MSSQL, GitHub, Azure | | | |
| 고려사항 | API 구현, 사용자인증과 검증, 연결된 사용자 경험, | | | |
| 프로젝트 내용 | **[핵심 구현 기술]**   1. **React를 활용한 사용자 인터페이스** 2. React – typeScript를 사용하여 코드의 안정성과 가독성 높임 3. **Redux Toolkit를 활용한 API서버 통신** 4. Redux Toolkit을 사용하여 상태 관리 5. 같은 라이브러리의 Query를 활용하여 API 요청 처리(효율성 높임) 6. **ASP.Net**   a. ASP.NET Core를 사용하여 RESTful API를 제공  b. 일관된 형식으로 API 응답 반환  c. 적절한 HTTP 상태 코드 반환  .Net 환경과 React을 결합한 풀스택 애플리케이션을 구축했습니다. ASP.NET core 를 활용하여 RESTful API를 제공하는 서버를 구축했습니다. Core Identity를 활용하여 사용자 인증과 권한 관리를 구현했습니다. EF Core 를 활용하여 데이터베이스의 확장성을 높였습니다. 또한 적절한 HTTP 상태 코드를 반환하기 위해서 노력을 기울였습니다.  React와 bootstrap을 활용하여 사용자와 상호작용하는 인터페이스를 구축했습니다. TypeScript를 사용하여 코드의 안정성과 가독성을 높였습니다. Redux Toolkit을 사용하여 전체적인 상태를 관리하고, 같은 라이브러리를 사용하여 API 서버와의 통신을 관리하고 효율적인 데이터의 흐름을 만들고자 하였습니다. | | | |
|  | 상세 구성도 | | | |
| Fonrtend |  | | **[ Redux Toolkit ]**  어플리케이션의 전역 상태를 관리하기 위해 사용했습니다.  **[ Redux Toolkit Query ]**  라이브러리의 통합성을 높이고 API 엔드포인트를 정의.  **[ TypeScript ]**  코드 타입 안정성 보장을 위해 사용  **[ Bootstrap ]**  앱의 빠른 빌드업과 반응형 디자인을 위해서 사용했습니다. | |
| Backend |  | | **[ ASP.NET Core ]**  웹 어플리케이션 백엔드를 구축  **[ MVC 아키텍처 ]**  표준화된 패턴을 따라 코드의 유지보수성을 높이고 재사용성을 증가  **[ EF Core ]**  EF Core를 사용하여 데이터 모델 정의와 데이터베이스 접근 간소화  **[ Swagger ]**  API 문서 생성 및 테스트 인터페이스 | |
|  | **[ Controllers ]**  모든 응답을 ‘ApiResponse’객체를 통해 구조화된 형식으로 반환  JWT를 활용한 인증 및 권한관리 기능  데이터베이스 CRUD기능  **[ Azure ]**  Image container , MSSQL 사용을 위해 Azure 이용.  **[ Blobs ]**  이미지 저장을 위하여 Blobs 사용 | | | |