**포트폴리오**

**지성욱**



|  |  |  |  |
| --- | --- | --- | --- |
| **프로젝트명** | **Mini ERP \_Program** | | |
| **개발기간** | 2018.01.04 ~ 2019.03.04 | **개발인원** | 5명 (중도 하차 1명) |
| **개발 환경** | Windows10 | | |
| **IDE** | Visual Studio | | |
| **개발 언어** | C# | | |
| **DBMS** | MS-SQL | | |
| **Server** | Microsoft Azure SQL Database | | |
| **Framework** | .NET Framework 4.6.1 | | |
| BarcodeLib | | |
| .NET Core Libraries  <System.Drawing.Common> | | |
| Tulpep.NotificationWindow | | |
| file-transfer-p2p | | |
| **개발배경** | 다양한 생산 산업 환경의 영업, 물류, 재고, 거래 와 생산 등 여러 업무의 처리를  편리하게 하기 위한 공장 관리 프로그램을 개발하려 하였습니다. | | |
| **개발내용** | 재고와 견적 상품관리 등을 유용하게 하며,  여러 머신 의 상태를 실시간으로 모니터링 해 에러에 대한 추적과,  자재 투입 현황을 실시간으로 알 수 있다.  또한, 그룹 메시지 기능을 이용해 작업자간 원활한 의사소통이 가능해져  생산의 효율성이 더욱 높아 지는 효과를 얻을 수 있다. | | |
| **아키텍처** |  | | |
| **담당업무** | **대시보드**   * 진행중인 모든 ‘거래를’ 조회해 실제 업무가 이루어 질 수 있도록   하는 기능을 구현했습니다. (출고, 생산, 출하, …etc)  **머신 클라이언트**   * 생산 공장의 ‘기계를’ 관리하기 위한 클라이언트입니다 * 투입된 자재의 정보와 개수를 수집합니다. * ‘모니터링 클라이언트’의 명령어를 전달받고 명령을 처리합니다.   **머신 모니터링 클라이언트**   * 여러 머신 들을 제어가 가능합니다. * 여러 머신 들의 실시간 상태 확인이 가능합니다. * 해당 머신 에 실시간 투입 자재 현황을 알 수 있습니다.   **품목의 바코드 출력**   * 모든 품목을 DB에서 가져와 품목코드를 ‘바코드’ 화 시키고 출력을 위해   이미지를 생성합니다.  **자재 계획서 출력 프로시저**   * 각 견적의 품목들의 재고량을 조회해 재고량만큼의 개수를 뺀 필요 자재 개수를 출력하는 프로시저   **구매 계획서 출력 프로시저**   * 자재 계획서 출력 프로시저에서 반환된 데이터를 사용해   자재의 재고량에서 빼 ‘구매해야 할 개수를’ 출력하는 프로시저 | | |
| **고찰** | 머신 을 제어하는 구조를 생각 할 당시 하나의 서버에 하나의 머신 을 접속하는 것과, 하나의 서버를 두고 클라이언트 들이 접속하는 구조를 생각했습니다.  결국 하나의 서버를 두고 여러 클라이언트들이 접속하는 방식으로 가닥을 잡고 구현하였습니다.  이로 인해 얻어낸 효과는 클라이언트들의 관리가 쉬워졌다는 것입니다.  모니터링 클라이언트가 서버에 메시지를 보내면 머신 클라이언트들이 메시지를 확인 후 커맨드를 실행시키는 구조를 가지게 되었습니다.  또한 로그관리가 유리해졌습니다. | | |
| 결국 서버 클라이언트 구조에서는 클라이언트가 질문을 하면 서버가  질문에 대한 응답을 하는 구조로 만들어 져야 한다고 생각 했습니다.  하지만, 서버는 다른 팀원이 만든 것을 그대로 사용했기에 원하는 구조로  수정을 할 수가 없었습니다.  이러한 점을 해결하기 위해 결국 서버를 명령어의 전달에만 사용하기로 정하고, 서버는 메시지를 전달하기만 하며 클라이언트간 메시지를  교환하는 구조를 이루게 되었습니다. | | |

|  |  |
| --- | --- |
| * 실행 화면 * 모니터링 클라이언트 * 대시보드 | |
| * 머신\_클라이언트 |  |