

IDG Deep Dive

“개발의 병목을 넘어서라”

로우코드의 가능성과 실용적인 접근 전략

애플리케이션 개발에 걸리는 시간을 단축하려는 시도는 어제 오늘의 일이 아니다. 비주얼 기반의 개발 환경이나 RAD 같은 개발 툴도 같은 맥락에서 볼 수 있다. 하지만 로우코드 플랫폼은 개발자의 범위를 확대하는 데 중점을 두고 있다. 현업 부서에서 필요한 애플리케이션을 개발자와 IT 부서에 의존하지 않고 직접 만들 수 있다면, 속도는 물론 현업 환경을 100% 반영한 애플리케이션을 기대할 수도 있다. 로우코드 플랫폼이 많은 한계와 개발자의 반대에도 지속적으로 확산되고 진화하는 것도 바로 이런 무시할 수 없는 이점 때문이다. 디지털 혁신 열풍과 함께 새삼 주목받고 있는 로우코드 개발의 현주소와 현실적인 접근 전략을 살펴본다.

🔗 Trends

‘클릭 몇 번으로 앱 만든다?’… 로우코드의 가능성과 한계

🔗 Practical Guide

‘약이 되거나 독이 되거나’… ‘로우코드’ 제대로 시작하기
현대적인 로우코드 개발 플랫폼의 4가지 특성

🔗 Low Code for Developer

개발자가 로우코드 플랫폼을 다시 생각해야 하는 이유
개발자가 로우코드 플랫폼을 믿지 말아야 할 때

🔗 Case Study

“속도가 경쟁력” 로우코드 개발의 혁신 가속화 사례



Trends

‘클릭 몇 번으로 앱 만든다?’... 로우코드의 가능성과 한계

Alex Cruickshank | IDG Connect

로우코드는 애플리케이션 구축, 특히 데이터 세트 관리와 처리를 위한 기업 자체 애플리케이션 구축에 필요한 프로그래밍을 완전히 없애거나 크게 줄일 수 있다는 점을 내세운다. 새로운 애플리케이션을 처음부터 새로 만들 필요 없이, 개발자는 물론 심지어 개발자가 아니라도 시각화된 개발 환경 내에서 다양한 리소스를 조합하기만 하면 된다. 버튼 몇 개만 클릭하면 ‘짜잔!’하고 새로운 애플리케이션이 등장한다는 것이다.

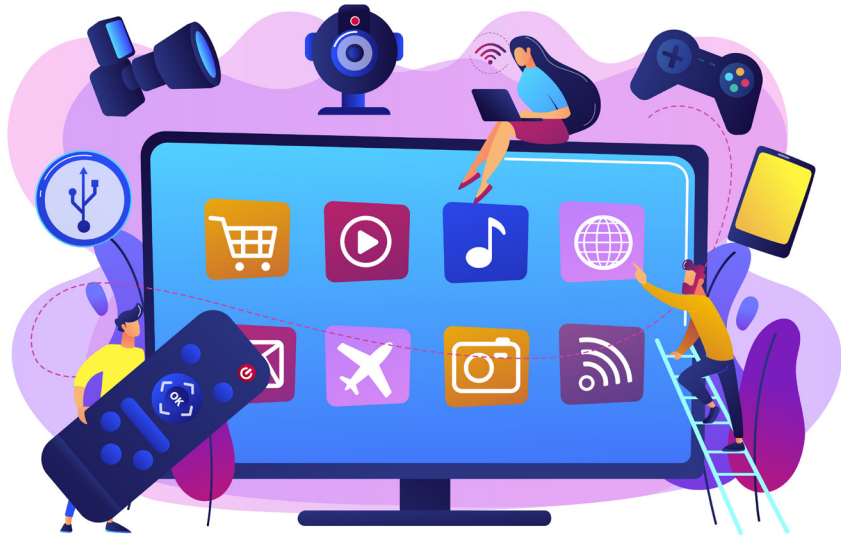
로우코드의 매력은 분명하다. 실제로 많은 조직에서 기하급수적으로 늘어나는 데이터 세트를 관리하기 위한 애플리케이션을 자체 제작하는 데 상당한 어려움을 겪고 있다. 이런 조직에 로우코드는 한 줄기 희망의 빛이며, 성가신 코딩 작업을 API, 애플릿 그리고 서비스에 떠넘기는 좋은 방법이다. 어떤 의미에선 마치 SF 영화와 같은 요소를 품고 있기도 하다. “블록을 가져다가 클릭만 하면 앱이 된다니!”

정말 이렇게만 된다면, 프로그래머가 무슨 필요가 있겠는가? 이론적으로 로우코드는 애플리케이션 개발 자체를 기술 전문가의 영역에서 가져와 비즈니스 애널리스트와 고객 지향적인 관리자의 통제 하에 둘 수 있는 수단이다. 애널리스트가 생성한 플로우 차트를 로우코드 명령어로 변환하고, 이를 다시 필요한 소프트웨어로 직접 변환할 수 있다. 이렇게 하면 개발자와 현업 간의 오해나 의사소통 문제도 줄고, 더 빠르고 간단하게 애플리케이션을 개발할 수 있다.

코딩을 제외한 나머지 개발의 비중

그러나 여기에는 매우 심각한 장벽이 하나 있다. 프로그래밍에는 기술과 경험, 그리고 특정한 마음가짐이 필요하다. 유능한 프로그래머는 끈기와 단련된 문제 해결 능력, 논리에 기반한 업무 전략, 체계적인 오류 제거, 세부사항에 대한 놀라운 주의력, 그리고 변수나 루틴 등 열 개가 넘는 요소를 한 번에 곡예하듯 다룰 수 있는 능력이 있다. 물론 능력에 비해 임금이 너무 적은 느낌이 드는 경우도 있지만, 그것은 프로그래밍이 그만큼 고도의 집중력과 뛰어난 역량을 필요로 하는 작업이라는 걸 이해하지 못하는 기업 탓이지 프로그래머의 잘못은 아니다. 즉, 프로그래밍은 말로는 쉬워 보여도 실제로 해보면 절대 그렇지 않은 작업이다.

이 작업에서 코딩 파트를 제거하거나 축소한다 해도 생각한 만큼 영향이 크지는 않을 것이다. 코딩은 결국 개발 아이디어를 구현하는 것이다. 하지만 코드의 상당 부분을 블록으로 대체



한다 해도 결국 아이디어를 ‘구현’하는 역량은 필요하며, 오히려 코딩할 때보다 더 고난이도의 작업이 될 것이다.

컴퓨팅의 역사에서 예를 찾을 수 있다. 고급 언어가 등장했다고 해서 어셈블리 프로그래머가 전부 실직자가 된 것도 아니고, 또 어셈블리 언어 자체에 대한 수요가 완전히 사라지지도 않았다. 오히려 기존 프로그래머가 새 언어에 맞춰 자신의 역량을 새롭게 사용했을 뿐이다. 고급 언어는 분명 어느 정도 진입 장벽을 낮춰주긴 했지만, 그것이 반드시 좋은 것이라고 볼 수도 없다. 프로그래머가 많다고 소프트웨어가 더 나아진다는 보장은 없기 때문이다. 플론의 법칙(Flon's Law)이 지적하듯 “나쁜 프로그램을 작성하는 데 가장 덜 어려운 언어는 지금도 앞으로도 존재하지 않을 것이다.”

로우코드 개발도 예외는 아니다. 오히려 프로그래머가 되기 위한 장벽이 더 낮아짐에 따라 소프트웨어 품질이 저하되지 않으면 다행이다. 비즈니스 애널리스트가 플로우 차트에 각종 아이টে를 드래그해 ‘앱 제작’ 버튼을 클릭한다 해도, 결국 복잡하고 지저분한 데이터 세트라는 현실을 마주하면, 쉽고 간편하고 깔끔한 애플리케이션 개발이라는 꿈은 난관에 부딪힐 수밖에 없다.

여전히 필요한 코딩이라는 과제

로우코드 개발 프로세스는 일반적으로 다음과 같이 설명할 수 있는데, 이것만으로도 많은 것을 시사한다.

1. 비주얼 IDE(Integrated Development Environment)를 이용해 프로세스를 구상한다.
2. 각종 앱, 서비스, 라이브러리 등을 호출해 실제 작업을 한다.
3. 필요하다면 여러 요소를 통합하기 위해 약간의 코딩을 가미한다.
4. 배치한다.

진짜 단순하고 기본적인 앱 제작을 제외하고서는, 아마도 3단계에서 막힐 확률이 높다. 이 단계에서는 로우코드 개발자 역시 프로그래머의 도움이 필요하기 때문이다. 처음에는 SQL 쿼리


몇 개만을 추가하는 것으로 시작할 지 몰라도, 조금만 애플리케이션이 복잡해지면 견잡을 수 없이 일이 커질 것이다.

좀 더 냉소적인 이들은 그래서 로우코드 이야기가 나올 때마다 “마케팅 부서에서 과연 얼마나 좋은 소프트웨어를 만들어 낼지 정말 기대된다”며 비꼬기도 한다. 그런가 하면 로우코드라는 개념 자체가 사실은 제대로 된 애플리케이션을 만드는 것이 얼마나 어렵고 힘든 일인지를 깨닫게 하기 위해 프로그래머 스스로가 만들어 냈다는 우스개소리까지 나온다. 이쯤되면 아주 오래 전 RAD(Rapid Application Development)가 떠오를 정도이다.

너무 한다고 생각할 수도 있지만, 이런 냉소적인 태도에는 다 이유가 있다. 기업은 오랜 경험을 통해 업무를 IT, 판매, 마케팅, 개발, PR 등 여러 부서로 나눠 처리한다. 그래야 일이 되기 때문이다. 부서간 커뮤니케이션이란 과제가 생기지만, 분업과 전문화를 통해 가장 만족스러운 결과를 얻었다. PR 담당자가 어느 날 갑자기 코딩을 할 수는 없고, 개발자가 어느 날 영업을 시작하는 것도 비현실적이다. 물론 자신의 전문 분야가 아닌 곳에서 일을 하는 사람이 없지는 않지만, 결과물은 상대적으로 비효율적이다. 예외는 있겠지만, 어디까지나 예외일 뿐이다. 비즈니스 애널리스트가 전문 코더보다 더 코딩을 잘 할 가능성은 극히 낮다.

그래서 설령 로우코드 전략을 도입한다고 해도, 실질적인 개발 업무는 여전히 프로그래머에게 집중될 가능성이 크다. 사실 코딩을 줄이기 위한 노력은 수십 년 전 객체 지향 프로그래밍(Object-Oriented Programming)과 라이브러리에서 시작됐다. 오늘날도 모든 프로그래머가 새로운 애플리케이션을 만들 때 옛날 코드를 일부를 재사용한다. 로우코드의 빌딩 블록이 아무리 크다 해도, 여전히 기술과 재능, 그리고 경험을 가진 개발자가 이들을 적절히 조합해 주지 않는 이상 유용하고 효과적이며 확장성까지 갖춘 결과물이 나오기 어렵다.

따라서 기업은 로우코드를 개발 병목 현상의 해결책으로 신뢰해서는 안된다. 로우코드는 능숙한 코더를 도와 애플리케이션 개발의 일정 부분을 더 신속하고 깔끔하게 하는 데 도움을 주는 보조 역할만을 할 뿐, 실제 프로그래머를 대체할 수는 없다. 로우코드 이니셔티브 도입을 고민하는 기업, 특히 비 전문가에게 개발 자체를 일임하려던 기업은 녹록지 않은 현실에 부딪히고 말 것이다.

세상에는 참으로 형편 없고, 버그투성이에, 쓸모 없는 소프트웨어가 많다. 성공적인 기업이 라면 굳이 여기에 하나를 더 추가하게 될 뿐인 선택을 하지는 않을 것이다. 병목 현상이 소프트웨어를 자체 개발하는 속도가 느려서 발생하는 것이라면, 가장 근본적인 해결책은 유능한 개발자를 더 많이 고용하는 것이다. 

Practical
Guide

Suresh Sambandam | InfoWorld

‘약이 되거나 독이 되거나’...
‘로우코드’ 제대로 시작하기

모든 기업은 몇 가지 공통된 핵심 과제를 가지고 있고, 이런 과제를 처리하기 위해 통일된 소프트웨어 솔루션을 사용한다. 모든 기업에 공통된 만큼 소프트웨어 솔루션도 비슷한데, 널리 사용되는 데이터베이스나 CRM 시스템이 대표적인 예이다. 그러나 범용 솔루션으로 해결할 수 없는, 해당 기업에 특수한 종류의 문제는 여전히 남는다. 이때는 전용 솔루션이 필요하다. 구매 주문을 생각해 보라. 이를 다루지 않는 회사가 없지만 기업마다 요건이 달라 모든 기업의 모든 프로세스를 관리하는 단일 제품은 있을 수 없다.

이런 고질적 문제는 단일 제품보다는 플랫폼 솔루션을 필요로 한다. 기업의 특수한 문제를 기존 범용 솔루션이 해결해 주기를 기대하지 않고, 기업이 필요한 것을 자체적으로 구축해 해결하는 방법이다.

로우코드 개발이란 무엇인가

이런 해법 중 최근 새로이 주목받고 있는 것이 바로 로우코드(Low-Code) 플랫폼이다. 로우코드 플랫폼은 현업 직원이 IT 부서의 관여 없이 자체적으로 애플리케이션을 만들 수 있다. 프로그래밍 지식이 전혀 필요 없거나 약간 필요한 정도이다. GUI(Graphical User Interface), 드



래그 앤 드롭 모듈 등 다양한 사용자 친화적인 구조를 이용해 프로그래머가 아니라도 자신만의 필요에 맞는 애플리케이션을 스스로 개발할 수 있다. 이렇게 만든 애플리케이션은 현업 사용자가 처한, 표준 해법이 없는 특수한 문제를 해결해 준다.

정의만 놓고 보면, 로우코드 플랫폼은 기업의 현업 사용자가 수십 년 동안 기다려온 솔루션이다. 하지만 로우코드 접근법을 올바르게 이해하고 로우코드가 적절한 상황이 있는가 하면 적절하지 않은 상황도 있다는 것을 알아야 한다. 로우코드가 기업에 어떤 식으로 기여할 수 있는지 자세히 살펴보자.

로우코드 플랫폼이 가장 효과적인 분야

로우코드 플랫폼이 이상적인 해법인 경우는 다음과 같다.

- 현업 사용자가 자체적인 애플리케이션을 만들고자 한다.
- 문제를 해결할 통일된 해법이 없다.
- 애플리케이션이 정상적인 비즈니스 사용례이다.

추상적인 가이드라인이므로, 실제 비즈니스 관심사와 결정의 맥락에서 생각해 보는 것이 필요하다. 실제 비즈니스 운영에서 이런 추상적인 가이드라인이 어떻게 나타나는지 몇몇 사례를 살펴보자.

첫째, 판매에 대해 생각해보자. 기업이 판매 송장을 생성하는 방식은 매우 다양하다. 당연한 말이지만, 송장은 CRM 시스템에 연결되어야 한다. 그러나 문서화되어야 할 승인 절차를 거치는 것이 보통이다. 영업팀은 소프트웨어 중심의 판매 송장 관리를 생성하고 설계하기에 가장 적절한 위치에 있다. 이 문제를 영업팀보다 더 잘 이해하는 곳은 없기 때문이다. 송장을 생성하는 애플리케이션은 정교할 필요가 없기 때문에 이를 개발하는데 로우코드 플랫폼을 이용할 수 있다.

둘째, 마케팅을 생각해보자. 마케팅팀은 콘텐츠 승인의 세계에서 일한다. 하나의 콘텐츠가 현실화되기까지 다양한 관련부서의 승인이 필요하다. 이 프로세스가 수작업으로 처리된다면 지연과 실수가 발생하기 쉽다. 따라서 소프트웨어를 통한 자동화가 매우 유익하다. 하지만 대다수 마케팅팀은 수작업을 통해 승인을 얻는데 의존한다(보통 이메일). 업무 관리 시스템에서 이를 자동화하기가 쉽지 않기 때문이다. 또 마케팅팀의 프로세스는 기업마다 달라서 범용 솔루션을 이용하는 것도 쉽지 않다. 대신, 로우코드 플랫폼을 이용한다면 마케팅팀은 필요할 때마다 승인 워크플로우를 IT 부서의 개입 없이 구축하고 재구축할 수 있다.

마지막으로, HR 부서를 생각해보자. 어느 기업이든 놀라울 정도로 복잡하고 미션 크리티컬한 업무는 직원 채용이다. 모든 기업에 중요하지만, 표준화된 채용 방식을 만들기란 거의 불가능하다. 채용 요건, 교육, 서류 작업 프로세스가 기업마다 다르기 때문이다. 그러나 채용 및 여타 인력 관리는 종종 신속하게 진행해야 한다. HR팀은 IT 부서가 좀 더 긴급하고 중요한 작업을 끝마칠 때까지 몇 주나 기다릴 여유가 없다. 여기서 로우코드 솔루션은 HR 담당자가 자동화된 채용 프로세스를 스스로 갱신할 수 있으므로, 자신의 작업을 신속하고 효과적으로 완수하는 데 필

요한 톨을 엄격하게 제어할 수 있다.

예를 들어, HR 관리자는 모든 필요한 정보를 채용 후보자로부터 수집하는 양식을 생성하고, 앞으로 상호작용할 부서를 모두 포함하는 워크플로우를 생성하는 앱을 개발할 수 있다. 각 부서의 관계자가 관련 작업을 완수하면 데이터는 자동으로 다음 단계로 진행하고, 중간에 IT 부서가 개입할 필요가 없어 채용 과정을 보다 효율적으로 만든다.

로우코드 플랫폼이 유효하지 않은 경우

어떤 조직에서든 두 종류의 프로세스가 있다. 하나는 구조화된 프로세스이고, 다른 하나는 개방적인 프로세스이다. 구조화된 프로세스는 엄격하게 준수하는 것이 보통이고, 제반 업무의 대략 2/3를 차지한다. 휴가 관리, 출근, 조달 등 기업이라는 집단의 '생명 유지' 기능에 해당한다.

예를 들어, 잡지사는 기사 출고 방식에 대한 명확한 프로세스가 있다. 우선 기고자가 기사를 작성하고 편집자가 검토한다. 그 후 편집장이 기사를 확인한 후 이를 디자인팀에 보내도록 승인한다. 마지막으로 인쇄소에 전달되고 회계팀은 각종 결제 과정을 처리한다. 혼란을 피하려면 이 워크플로우는 매주, 나아가 매분기 일관성 있게 유지되어야 한다. 이들 프로세스는 구조와 목표가 명확하므로 로우코드 솔루션으로 훌륭하게 처리될 수 있다.

그러나 개방형 프로세스는 정의하기가 쉽지 않고, 목표가 항상 분명하지도 않다. 1회성 행사를 개최한다고 하자. 최종 결과가 어떨지는 대충 알고 있지만, 이런 행사를 항상 주관하는 것은 아니므로 계획 프로세스를 사전에 정의할 수 없다. 예컨대 오프사이트 미팅의 일정을 수립하는 것과 같은 비정형 프로세스는 훨씬 더 협업적인 경향이 강하고, 흔히 여러 이해관계자의 의견에 따라 유기적으로 진화한다.

이런 종류의 작업은 로우코드 플랫폼을 이용해 솔루션을 만들기 어렵다. 이런 워크플로우에는 페이스북의 워크플레이스, 슬랙, 여타 플랫폼이 더 적합하다. 로우코드 솔루션에 적합한 정형화된 구조가 없을 때 가장 효과적이기 때문이다. 물론, 이러한 비구조적 활동은 빈번히 조직에서 최고의 가치를 창출한다. 그래서 적절한 시간과 관심을 투입하는 것이 결정적이다. 반복적이고 엄격하지만 필수적인 프로세스는 내부 전문가가 만드는 로우코드 솔루션에 맡기고 나머지 인력은 모두 혁신을 주도하는 창의적인 문제 해결을 자유롭게 추구할 수 있다.

로우코드 플랫폼 시작하기

로우코드 플랫폼은 분명히 이를 구현하는 기업의 성공을 가속화할 수 있다. 그러나 생소한 접근법이어서 이를 조직에 정착시키기 위해서는 몇 가지 난관을 극복해야 한다. 특히 로우코드 개발 플랫폼의 대부분이 클라우드 기반 플랫폼이라는 점도 적지 않은 영향을 미친다. 로우코드 세계에 진입할 때 다음의 원칙을 기억하기 바란다.

1. SaaS를 수용한다

SaaS 시스템을 도입해 로우코드 접근법을 활용하려면, 먼저 전체 팀이 기존의 기업용 톨로부터 탈피할 의지가 있어야 한다. 여기서 '탈피'란 일반적인 SaaS, 특히 APaaS(Application Platform as a Service)로의 이동을 가리킨다. 어느 조직에서나 변화는 쉽지 않지만, 기술 환경


이 진화하는 흐름을 거부한다면 뒤처지고 말 것이다. 따라서 무엇보다, 조직이 온라인 플랫폼을 솔루션으로 수용하려는 의지가 있어야 한다. 저항에 부딪히면 핵심 관계자들을 설득해야 한다.

2. 보안에 대한 두려움을 잠재운다

SaaS 및 클라우드 서비스 업체는 보안이 경쟁력을 유지하는 데 있어 최우선 순위임을 너무나 잘 알고 있다. 침해가 하나라도 드러나면 업체의 생존이 위태로워진다. 따라서 대다수 SaaS 제품에 보안 기능이 포함되어 있는 것은 새삼스럽지 않다. EU의 GDPR 등 날로 엄격해지는 규제가 보안 측면에서 SaaS 제품을 온프레미스 솔루션보다 오히려 더 견고하게 만들고 있다. 따라서 보안이 로우코드 플랫폼의 도입에 장애가 되어서는 안된다. 서비스 업체에 보안 프랙티스를 문의하고, 보안 문제에 관해 개방적이고 솔직한 대화를 나누도록 한다. 기업은 보안을 우려하는 대신 핵심 영역에서 가치를 창출하는데 주력할 수 있다.

3. 전문가가 필요로 하는 툴을 제공하라

전문가를 최고의 툴로 무장시키는 것이 혁신의 시작이다. 이들이 사용하는 툴을 일일이 제한하는 것은 좋지 않다. 따라서 실제 사용자를 제한하는 정책보다는 권한을 부여하는 정책을 세우는 것이 좋다. IT 부서와 매일 기술을 사용하는 사람과 상의하지 않고 일을 추진한다면 재난으로 귀결될 것이다.

전문가가 자신의 전문 영역에 대해 재량을 가지고 있고, 업무를 강화할 수 있는 변화를 신속하고 고통 없이 이행할 수 있다면 조직에는 바로 혜택이 돌아온다. 로우코드 플랫폼을 채택하면 HR로부터 마케팅, IT에 이르는 모든 사람이 자신이 가장 잘 아는 분야에 몰입할 수 있고, 자신만의 독자적 기술과 재능을 발휘할 수 있다. 

● **Suresh Sambandam**은 SaaS 기반 엔터프라이즈급 워크플로우 및 비즈니스 프로세스 자동화 플랫폼 업체인 키스플로우(KissFlow)의 CEO이다.

현대적인 로우코드 개발 플랫폼의 4가지 특성

오늘날 로우코드 개발 플랫폼은 디지털 트랜스포메이션을 경험하는 모든 기업에서 중요한 역할을 할 수 있다. 오르락 내리락하는 요구사항과 새로운 시장 기회들이 지속적으로 나타나는 상황에서 민첩성을 추구하는 조직이라면 더 빠르게 혁신하고 타임투마켓(Time to Market)을 줄일 수 있는 방법을 찾아야 한다. 전통적 개발 사이클은 매우 복잡하고, 수 개월씩 걸리는 반면, 로우코드 개발 플랫폼은 기업 소프트웨어 개발을 훨씬 단순화해 몇 주, 심지어 몇 일로 배치 과정을 단축해 준다.

로우코드 개발 플랫폼은 새로운 개념은 아니다. 비주얼 베이직이나 파워빌더와 같은 고속 애플리케이션 개발 툴은 원래부터 있었고, 그 외에도 마이크로소프트 액세스와 같이 최소한의 코딩만으로 빠르고 효율적으로 애플리케이션을 제작할 수 있는 단순하고 쉬운 개발자 플랫폼은 수십 년 전부터 존재해왔다.

그러나 구성에 주로 신경을 썼던 과거의 로우코드 솔루션들은 기획, 디버깅, 테스트, 실행, 배치 등 소프트웨어 개발 라이프사이클의 다른 단계들을 매끄럽게 능률화하지 못했다. 그 결과 제대로 된 테스트를 거치지 않은 로우코드 소프트웨어를 출시하면 할수록 개발 프로세스는 더 복잡해지고, 시간도 더 많이 걸렸다. 사용하기 쉽다는 이유로 로우코드를 선택했던 비개발자들에게는 버그가 많은 프로젝트를 수정해야 하는 것이 여간 곤혹스러운 일이 아니었다.

다행히도 이제는 이런 문제로 고생할 필요가 없다. 오늘날의 로우코드 개발 플랫폼은 버전 컨트롤에서부터 성능 테스트, 변화 관리에 이르기까지 전체 소프트웨어 개발 사이클의 각 단계를 최적화했다. 오늘날 제대로 된 로우코드 플랫폼으로 인정받기 위해서는 소프트웨어 제작이 쉬워야 할 뿐 아니라 손쉽게 문제를 파악하고, 테스트를 진행하며, 안정적인 애플리케이션을 전달할 수 있어야 한다. 확장성 또한 용이해야 함은 물론이다.

그렇다고 로우코드 개발 플랫폼만 가지고 모든 소프트웨어 개발 요구를 충족시킬 수 있다는 말은 아니다. 로우코드 플랫폼은 비즈니스 프로세스 관리나 운영 관리 등의 특정 솔루션에 특화되었을 때 더 많은 생산성을 이끌어낸다. 덧붙여 오늘날의 로우코드 플랫폼은 전통적인 프로그래밍 언어 및 개발 환경과 마찰 없이 연동되도록 설계되어 전문적인 개발자들도 로우코드 디자인 툴을 연장할 수 있다.



로우코드 개발 플랫폼을 고려 중인 기업들을 위해, 플랫폼에 인적, 물적 자원을 투자하기 전에 알아둬야 할 네 가지 주요 특성들을 모아 보았다.

사전 구축된 구성 요소

로우코드 개발 플랫폼들은 개발자가 설치와 동시에 바로 사용할 수 있는 기능을 제공해야 한다. 이런 사전 탑재형 요소들은 마치 레고와도 같다. 각 블록마다 수행하는 역할이 다르며, 더 큰 퍼즐의 일부로서 다양한 애플리케이션에서 사용할 수 있다. 예컨대 사전 탑재된 UI 컴포넌트가 있을 경우 UI 오브젝트를 처음부터 코딩할 필요가 없어진다. 또한 이런 사전 탑재형 요소들은 필요에 따라 손쉽게 재설정해 업데이트할 수 있다.

로우코드 플랫폼을 평가할 때는 사전 탑재된 요소 및 기능 라이브러리를 주의 깊게 살펴봐야 한다. 예를 들어 플랫폼의 목적이 기업 계약 관리 시스템을 만드는 것이라면, 문서 관리, 비즈니스 프로세스 관리, 비즈니스 룰, 그리고 PDF 계약 문서 생성과 같은 기능들을 중점적으로 보는 편이 좋다.

코드 작성 그 이상의 플랫폼

앞서 설명해듯이 코드를 쓰는 작업 외에도 소프트웨어 개발에는 여러 가지 복잡하고 자원 집약적인 단계가 많다. 기획, 테스트, 배치 등은 모두 많은 시간이 걸리는 일이다. 과거의 RAD 툴과는 달리 현대의 로우코드 개발 플랫폼은 전체 소프트웨어 개발 라이프사이클을 단순화시켜야 했다. 즉 디버깅, 통합 테스트, 성능 분석, 그리고 배치와 같은 과정을 단순화해야 한다. 또한 고급 개발 툴과 마찬가지로, 로우코드 개발 플랫폼은 개발자에게 소프트웨어가 개발을 거쳐 테스트와 프로덕션 단계를 거치는 과정에서 이전 버전으로 복구할 수 있어야 한다.

비즈니스와 IT의 유동적 협업 지원

빠르게 변화하는 오늘날의 디지털 기업은 IT와 비즈니스 간 긴밀한 협력을 통해 새로운 디지털 고객 경험을 창출해 내고 동시에 비즈니스 운영을 자동화할 수 있어야만 한다. 전통적인


소프트웨어 개발 과정은 새로운 애플리케이션을 기획하고, UI 실물 모형을 제작하고, 제품 사양을 결정하고, 코드를 쓰고, 모든 요소를 테스트하고, 허가를 받고, 구현 및 확장에 이르기까지 수 개월의 시간이 필요했다. 사용자 교육에도 시간이 걸리는 것은 말할 것도 없다. 긴 개발 사이클은 더 많은 시간과 자원을 소모할 뿐만 아니라 새로운 디지털 이니셔티브와 관련된 위험성 역시 커진다.

현대적인 로우코드 개발 플랫폼은 기술 비전문가들도 사용하거나 이해하기 쉬운 시각적 개발 수단을 제공하기 때문에 비즈니스와 IT 간 커뮤니케이션을 더욱 원활하게 한다. 이를 통해 비즈니스 프로세스 관리, 의사 결정 모델 등 프로세스 로직과 연관 비즈니스 룰을 비즈니스 전반이 이해할 수 있는 표준 형식으로 정리할 수 있다. 로우코드 툴은 이러한 설계를 구현하는 역할을 하며, 이를 통해 요건 문서에서부터 코딩에 이르기까지 전통적 개발 툴에서 요구되던 길고 긴 '번역' 과정을 건너뛸 수 있다. 로우코드 개발자는 숨겨진 납기 시일을 맞출 수 있고, IT의 설계가 비즈니스의 요구를 충족시키는지 확인할 수 있다.

클라우드를 수용한 로우코드 플랫폼

비주얼 베이직, 파워빌더, 그리고 마이크로 액세스와 같은 솔루션은 확장성이 없는 것으로 유명하지만, 현대적인 로우코드 플랫폼은 클라우드에 그 뿌리를 두고 있는 아키텍처이다. 사실, 오늘날의 로우코드 플랫폼은 이미 정부 프로젝트나 금융기관에 대규모 미션 크리티컬한 솔루션을 전달하는 데 사용되고 있다. 뿐만 아니라 로우코드 플랫폼은 PCI, FedRAMP 컴플라이언스 등 사전 인증을 받은 클라우드 보안 인증서도 있다.

로우코드 애플리케이션 개발 플랫폼은 오래 전부터 있어왔다. 하지만 이전 세대 플랫폼들은 지나치게 구성에만 집중하고, 전체 소프트웨어 개발 라이프사이클을 지원하지 못했으며, 확장성이 떨어져 평가가 좋지 않았다. 그러나 오늘날의 로우코드 개발 플랫폼들은 사전 탑재된 요소들을 충분히 보유하고, 전체 개발 사이클에 걸쳐 복잡성을 제거했으며, IT와 비즈니스 간 협업을 용이하게 했고, 무엇보다 동적인 클라우드 아키텍처를 수용했다.

오늘날의 로우코드 개발 플랫폼은 소프트웨어 개발 과정을 가속화하고, 비즈니스 프로세스를 능률화하며, 비즈니스의 요구에 맞춰 놀라운 확장성을 보여줄 것이다. 

개발자가 로우코드 플랫폼을 다시 생각해야 하는 이유

새로운 애플리케이션에 대한 비즈니스 수요는 그 어느 때보다 크다. 동시에 더 많은 소프트웨어 프로젝트와 애플리케이션 출시를 지원하는 더 빠르고 혁신적인 방법을 찾으라는 IT에 대한 요구도 커지고 있다. 하지만 아무리 프로세스나 툴이 발전하더라도 그것은 말처럼 쉬운 문제가 아니다.

로우코드(Low-code) 개발, 노코드(No-code) 개발은 IT 부서와 개발자가 치를 떠는 개념이다. 하지만 꼭 그렇게 거부해야만 하는가? 이들 방법론을 무조건 나쁜 것으로 치부하기 전에 한 번쯤은 꼼꼼히 살펴보고, 혹 제대로 된 전문가의 지도가 있다면 실제로 효과적인 방법론이 될 수 있지는 않을까 생각해 볼 필요가 있다. 최소한 이런 방법론의 한계를 좀 더 구체적으로 파악한다면, 구체적인 분석과 근거에 기반해 반대할 수 있을 것이다.

비즈니스 책임자가 로우코드에 관심을 갖는 이유

개발팀은 다음과 같은 압박에 직면하고 있으며 이로 인해 비즈니스 리더는 IT 개발자를 보조하기 위해 새로운 접근 방식을 모색하고 있다.

- 3개 부서에서 생산 작업을 추적하기 위해 스프레드시트를 공유하고 있다. 스프레드시트의 데이터 검증은 제한적인 데다 오류가 발생해 작업 품질 문제가 발생하고 있는 상황이다. 스프레드시트를 없애고 부서의 생산성과 품질을 향상시키는 새로운 워크플로우 애플리케이션을 며칠 이내에 프로토타입으로 제작, 개발, 테스트 및 배포까지 하는 것이 가능한가?
- 한편 현장 운영팀은 고객을 방문할 때마다 여러 기업 시스템의 정보를 업데이트해 달라는 요청을 받는다. 과연 개발팀은 운영팀을 위해 시스템에 쉽고 빠르게 연결할 수 있는 단일 툴을 제공하는 모바일 애플리케이션을 얼마나 신속하게 만들 수 있는가?

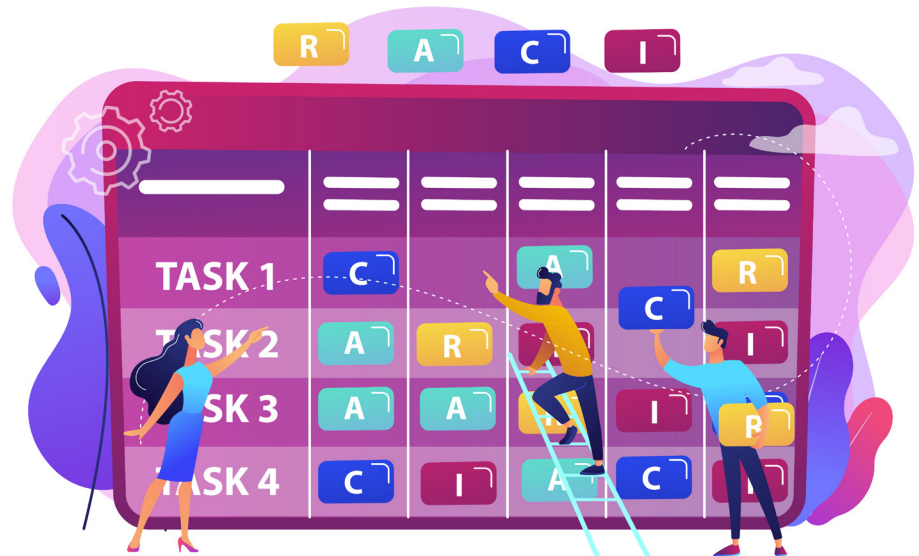
비즈니스 책임자는 운영을 디지털화하고, 고객에게 개인화된 경험을 제공하면 할수록 상당한 경쟁 우위를 확보할 수 있다는 것을 알고 있다. 이들은 애자일 개발 프랙티스를 지원하고, 애플리케이션을 조기에 출시하며, 사용자들로부터 배우고, 변화하는 수요와 기회에 맞춰 애플리케이션을 조정하고자 한다.

이 가운데서도 특히 CIO나 CFO는 애플리케이션 개발 및 지원 비용에 각별히 신경쓴다. 그렇기 때문에 지속적으로 지원이 필요한 레거시 플랫폼과 전용 애플리케이션의 수를 줄이고 싶어 한다. 이들은 유능한 소프트웨어 개발팀이 수익을 창출하거나 막대한 경쟁 우위를 제공하는 애플리케이션을 개발해 주기를 기대한다. 또한 운영 환경에 새롭게 구축된 애플리케이션의 확장성과 성능을 확인하고자 한다. 이를 위해 비즈니스 책임자는 소프트웨어 개발 생산성과 품질을 향상시킬 좀 더 민첩한 개발 프로세스를 찾고 있다.

IT 솔루션 업체는 애플리케이션 개발, 지원을 좀 더 용이하게 만들어 줄 플랫폼과 툴로 이런 기업의 요청에 응답했다. 예를 들어, IDE(Integrated Development Environment, 통합 개발 환경) 및 코드 에디터와 같은 툴을 사용하면 자바, 닷넷, PHP, 자바스크립트, 파이썬 및 기타 프로그래밍 언어로 커스터마이징 애플리케이션을 개발해야 하는 상황에서 생산성을 끌어올릴 수 있다.

한편, 애플리케이션 개발, 지원에 필요한 코드를 최소화하는 전략을 택하는 경우도 있다. 이들은 로우코드, 노코드, 또는 시민 개발 플랫폼이라고 부른다. 이들은 보통 클라우드 기반 기술들로, 애플리케이션 개발을 위한 툴을 제공하며, 프로덕션 사용례에서 애플리케이션을 구동하고, UI와 데이터, 그리고 워크플로우를 다른 기술과 통합하기도 한다.

로우코드 툴 공급업체 아웃시스템스(OutSystems)의 개발 지원 책임자 스테이시 르바인은 “비즈니스적 관점에서 로우코드 플랫폼이 가치있는 이유는 로우코드 개발을 할 경우 전통적인 코드에서 신경쓰던 미묘한 뉘앙스를 개의치 않고 복잡한 핵심 프로세스를 전달하는 데에만 온전히 집중할 수 있기 때문이다. 개발자는 애플리케이션을 더 빠르게 내놓을 수 있고, 이로 인해 비즈니스에 더 많은 가치를 제공할 수 있다”고 주장한다. 또 다른 로우코드 툴 업체 퀵 베이스(Quick Base)의 전략 및 제품 담당 수석 부대표 제이 제이미슨은 “오늘날 기업 환경에서는 개발자에 대한 수요는 크고 공급은 턱없이 부족한 상황이다 보니 개발자가 무척 귀한 몸이 되었다. 이제 소프트웨어가 지배하는 세상이 오면서, 기업이 개발 역량을 최대한 확보하는 것이 무척 중요해졌다. 로우코드 플랫폼은 이런 상황에 대한 나름의 해결책을 제시해



준다”고 강조했다.

이들 플랫폼 중에는 15년 전부터 존재해 왔던 것들도 있으며, 특히 요즘 들어서는 소프트웨어 애플리케이션 개발에 투자하는 기업이 늘어나면서 로우코드 플랫폼을 채택하는 곳도 늘고 있다.

로우코드와 노코드의 차이점

한편, 이들 플랫폼을 지칭하는 용어가 다소의 혼란을 야기하기도 한다. 로우코드 플랫폼이란 용어는 애플리케이션을 개발하기 위해 필요한 코딩이 있음을 의미하지만, IT 업체는 애플리케이션을 개발하기 위해 '저노동(Low Labor) 툴을 판매하고 있다. 저노동 툴이란 드래그 앤 드롭 인터페이스, 시각적 프로그래밍 모델, WYSIWYG 사용자 인터페이스 설계 툴 및 기타 패러다임의 형태로, 더 적은 시간에 가능한 높은 품질의 애플리케이션을 만들 수 있는 개발 환경을 의미한다.

로우코드 플랫폼은 대상 개발자도 다르다. 일부 코딩이 필요한 좀 더 정교하고 복잡한 로우코드 플랫폼은 애플리케이션 개발자를 대상으로 하며, 개발 과정을 좀 더 쉽고 생산적으로 만드는 것이 목적이다.

한편, 애플리케이션 개발에 코드가 거의, 또는 아예 필요없는 플랫폼도 있다. 이런 플랫폼을 사용하면 비즈니스 개발자가 자체적인 애플리케이션을 생성하고 지원할 수 있다. 이런 플랫폼은 개발자가 정교한 애플리케이션을 만드는 데에도 사용할 수 있다.

이들 플랫폼은 리서치 업체마다 부르는 이름도 다르다. 가트너는 이를 '서비스형 애플리케이션 플랫폼(APaaS, Application Platforms as a Service)'이라고 부르며, 포레스터는 이를 두 카테고리로 나누어 애플리케이션 개발 및 전달을 위한 로우코드와 비즈니스 개발자를 위한 로우코드로 구분했다.

로우코드 플랫폼 평가 시 핵심 질문

로우코드 플랫폼에서 가장 중요한 것은 비즈니스가 필요로 하는 종류의 애플리케이션을 괜찮은 품질의 사용자 경험과 함께 제공할 수 있는가다. 이것을 기존의 프로그래밍 방식보다 더 저렴한 비용, 시간 및 인력으로 달성할 수 있는지 여부다.

이들 플랫폼 가운데 상당수는 일반적인 개발 언어 및 관련 환경처럼 모든 종류의 애플리케이션 개발에 다 사용할 수 있는 것이 아니라 특정한 부류의 애플리케이션 개발에만 맞도록 최적화되어 있기도 하다. 이들은 개발자가 애플리케이션을 빠르고 쉽게 만들 수 있도록 일종의 가이드라인을 제공한다. 그렇다면 로우코드 개발 플랫폼을 검토할 때 어떤 사항을 중점적으로 봐야 할까? 다음은 로우코드 플랫폼들을 검토할 때 반드시 확인해야 할 사항이다.

- 해당 플랫폼이 어떤 종류의 애플리케이션 개발에 도움을 주는가.
- 플랫폼이 만족스러운 사용자 경험을 제공하는가, 아니면 결국에는 네이티브 코드로 다시 커스터마이징하는 과정을 거쳐야 하는가.
- 해당 플랫폼이 제공하는 통합이 자사가 개발하고자 하는 애플리케이션에 부족함이 없는가.

- 개발자 경험과 기술 기대치가 조직의 역량과 잘 부합하는가.
- 해당 플랫폼이 지원하는 호스팅 모델 및 규정 준수 표준이 자사의 규제 필요를 충족하는가.
- 이 플랫폼의 개발 프로세스 및 애플리케이션 수명 주기는 자사가 개발하고자 하는 애플리케이션의 최소 기준치를 충족하는가.
- 개발하려는 애플리케이션에 대해 비용 모델은 잘 부합하는가.

로우코드 플랫폼이 더 적합한 롱테일

로우코드 업체 킨톤(Kintone) CEO 데이브 란다는 로우코드 및 노코드 플랫폼이 충분히 성숙해 이제는 상당히 정교한 수준의 애플리케이션 제작에 사용될 수 있다고 말했다. 한편 또 다른 업체 카스피오(Caspio) CEO 프랭크 자마니는 “이들 플랫폼을 사용하는 것이 적합한 애플리케이션이 따로 있다”며, “모든 기업은 롱테일 프로세스를 가지고 있다. 대부분의 기업에서는 주력 운영 기능이 가장 많은 주목을 받지만, 롱테일 문제를 해결하지 않고는 그 어떤 조직도 혁신할 수 없다”고 설명했다. 모든 부서에 존재하는 롱테일 프로세스의 대표적인 예는 다음과 같다.

- 마케팅의 경우 기업 웹사이트에 게재되는 콘텐츠에 대한 편집 과정을 추적하는 하나의 방법으로 로우코드 플랫폼이 사용될 수 있다.
- 인사과라면 새로운 직원이나 계약 업체를 채용하기 위한 프로세스 간소화에 사용할 수 있다.
- 재무부의 경우 부서별 예산을 제안, 검토 및 조정하는 연례 워크플로우가 될 수 있을 것이다.
- 운영팀에서는 주요 위기 상황이나 문제에 대응하도록 설계된 특수 애플리케이션 제작에 이 같은 플랫폼을 사용한다.
- IT 부서에서 부서 예산을 관리하고 IT 자산 추적 및 혁신 파이프라인을 관리할 수 있다.
- 로우코드 플랫폼은 또한 공급망 측면과 파트너와의 워크플로우, 그리고 공급업체 관리를 위한 툴 등을 관리할 때에도 사용된다.

개발자에게 적군인가 아군인가

이러한 롱테일 사용례들의 공통점은 문제 해결을 위해 앱을 개발해야 한다는 사실을 설득하기 어렵다는 것이다. 그 결과 일부 사용례들은 충분한 지원을 받지 못하기도 한다.

- 이메일, 스프레드시트 등 기타 오피스 툴로 임시 방편 워크플로우를 만들어 문제를 해결하려는 경우
- IT의 통제권에서 벗어난, 외부 구입 기술을 사용해 문제를 해결하는 경우. 이는 비효율과 리스크를 증가시키는 일종의 새도우 IT가 된다.
- 가장 최악의 경우는 해결책이 없는 상태에서 그 어떤 정보나 협업 툴도 제공하지 않고 ‘알아서 해결하라’는 식으로 직원들에게 떠밀어 두는 것이다.

이런 바람직하지 못한 결과를 피하려면, 개발자가 앞장서서 로우코드 솔루션을 사용하도록 주도해야 한다. 특히 좀 더 복잡하고 정교한 애플리케이션 개발에 로우코드 기술을 사용할 수 있다면, 기존의 개발 방식으로는 비용이나 시간 측면에서 도저히 감당할 수 없었던 비즈니스



수요를 충족할 수 있다. 로우코드 플랫폼이 제공하는 가이드라인이 애플리케이션 품질을 향상시키고, 사용자 경험을 개선하며, 네이티브 개발 언어를 사용해 만든 애플리케이션보다 안전한 호스팅 플랫폼을 제공해 줄 것이다.

또한, 개발자는 유지 및 지원 가능한 애플리케이션 개발에 있어 비즈니스 개발자의 노력을 지원할 수도 있다. 로우코드 및 노코드 플랫폼이 있으면 비즈니스 개발자가 직접 필요한 애플리케이션을 개발할 수 있지만, 그럼에도 여전히 UI 디자인이나 데이터 아키텍처, 명명규칙(naming conventions), 테스트와 같은 작업에서는 개발자의 도움이 필요하다. **ITWORLD**

개발자가 로우코드 플랫폼을 믿지 말아야 할 때

애플리케이션 개발 세계는 끊임없이 변화한다. 애널리스트들이 애플리케이션 개발 툴과 플랫폼의 여러 범주와 정의를 자주 수정하는 것만 봐도 알 수 있다. 빠른 발전 속도와 그에 따른 변화를 이끄는 동력은 데스크톱과 웹, 모바일, 웨어러블 등을 아우르는 고객 중심의 옴니채널 앱을 신속하게 제공하기 위한 단일 플랫폼과 툴세트에 대한 수요다.

개방형 표준을 기반으로 하는 적절한 로우코드 솔루션은 “더 많은 앱을, 더 빠르게, 모든 곳에서 실행”이 화두인 시대에 큰 가치를 제공한다. 물론 로우코드 솔루션이라고 다 똑같지는 않다. 솔루션을 평가할 때 주의해서 살펴야 할 세 가지 경고 신호는 다음과 같다.

로우코드 경보 1. 블랙박스

로우코드는 “내부를 볼 수 없는 블랙박스”라는 오명을 갖고 있다. 개발자들은 자신이 통제할 수 없는 어떤 것에서 미션 크리티컬 서비스를 실행하기를 꺼린다는 점을 생각하면, 왜 그런 오명이 생겼는지 이해할 수 있다. 빠르게 변화하는 환경에서 생산성 증대를 위한 답은 극단적인 노코드 블랙박스가 아니라 개방형 표준을 기반으로 하고 소스가 완전히 공개된 “오픈박스”인 로우코드 솔루션이다. 본질적으로 로우코드는 코드를 사용하는 사람들에게서 가치가 나오는 툴이다. 즉, 노코드 비즈니스 사용자가 아니라 전문적인 개발자가 필요하다.

전문 개발자를 위한 로우코드의 핵심은 한 번 써서 여러 플랫폼에 걸쳐 실행하고, 이 과정에서 사용자 경험에 대한 완전한 통제력을 유지하는 것이다.

로우코드 경보 2. 모놀리식 아키텍처

앱 개발은 이미 복잡하다. 여러 채널에 걸친 무제한 확장성을 달성해야 하고 컨테이너와 마이크로서비스를 사용한 클라우드 기반 개발로의 빠른 전환에도 대처해야 한다. 개발팀은 이러한 기대치를 충족하는 동시에 사용자 경험에 대한 초점도 계속 유지해야 한다. 하지만 개발자들은 로우코드 솔루션 아키텍처에 회의적이다. 애플리케이션 개발에 적합하지 않은 구시대의 모놀리식 기술이라고 생각한다. 이렇게 생각하는 이유는 다음과 같다.

— 모놀리식 로우코드 아키텍처는 처음에는 개발하고 배포하기 쉽지만, 장기적으로 보면 “밀착 결합” 되는 경향이 있고, 이로 인해 확장과 유지가 어려워진다. 어느 한 프로그램 구성 요소를

업데이트해야 하는 경우 보통 애플리케이션의 많은 부분을 다시 작성해야 한다.

- 모놀리식 로우코드 아키텍처는 솔루션에 내장되어 잘 보이지 않는 종속성을 가진 경우가 많으므로 이해하기가 어려울 수 있다. 구성 요소를 개별적으로 개발, 테스트, 배포, 확장하는 기능이 없는 로우코드 솔루션이 많고, 이 경우 커다란 단일체로 개발과 테스트, 프로덕션 작업을 관리해야 하는 부담이 생긴다.

로우코드 경보 3. 전용 툴세트

로우코드 솔루션에 포함된 전용 툴을 보유할 때의 이점은 툴과 플랫폼의 정렬이다. 두 가지의 출처가 한 곳이기 때문이다. 정렬은 이론적으로는 개발자, 궁극적으로는 비즈니스에 이익이 된다. 그러나 전용 툴의 현실적인 문제도 있다. 가파른 학습 곡선과 업체 종속성 문제, 미비하거나 아예 없는 코드 샘플 생태계와 자습서, 해당 업체 전용 커뮤니티 등이 대표적인 예다. 플랫폼 업체가 표준 언어 사용을 장려하면서 여전히 독자 개발 프로세스 내에 사용자를 가두어 두는 경우도 있다.

틈새 스킬을 배우고 통합하기 위해 필요한 비용과 시간이라는 문제 외에 숙련된 개발자는 업체 전용 툴로의 전환에 저항할 가능성이 높다는 문제도 있다. 회사가 전용 툴로 전환하면 소속 개발자들은 주류 개발 커뮤니티에서 멀어지는데, 이는 경력 관리에 결코 도움이 되지 않기 때문이다. 또한 전용 코드는 예제를 찾고 문제에 대처하는 데 참고할 자원이 적고 그만큼 디버깅하기도 어렵다.

로우코드의 성공 공식


전용 프로세스, 전용 플랫폼, 전용 코드에 따르는 경직성과 위험을 보면 전문 개발자들이 전용 툴을 도입하거나 추천하기를 꺼리는 이유를 명확히 알 수 있다. 이상적인 솔루션은 로우코드 개발의 용이함을 AWS나 애저, 구글 클라우드 플랫폼 등이 제공하는 클라우드 기능을 활용하는 옵션까지 포함해서 개발자들이 선호하는 표준 툴과 플랫폼으로 가져오는 것이다.



서버리스 클라우드 기반 플랫폼에서 개발하면 플랫폼에서 마이크로서비스와 함수를 관리하고 자동으로 확장할 수 있다. 이를 Node.js 백엔드, 그리고 웹과 모바일을 위한 자바스크립트 기반 프론트엔드와 결합하면 풀스택 옵션을 갖출 수 있다. 이 경우 다음을 포함한 여러 이점을 얻게 된다.

- 자바스크립트를 기반으로 구축된 서버리스와 로우코드 플랫폼을 결합하면 기존 개발자 스킬을 사용해서 소비자 등급의 옴니채널 경험에 대한 요구를 충족할 수 있다.
- 여러 개발자, 나아가 여러 팀이 각각 독립적으로, 다른 팀이 수행한 변경 사항과 충돌하는 경우 없이 자신이 담당하는 애플리케이션 부분에서 작업할 수 있다.
- 앱 코드를 다시 쓰거나 다시 배포하지 않고 업데이트할 수 있다.
- 코드를 여러 앱에 재사용할 수 있으며, 기능이 격리되므로 유지 관리하기도 더 쉽다.

전체적인 결과는 개발자가 앱 기능과 사용자 경험에 집중할 수 있는 자유를 얻고, 나머지는 플랫폼이 관리한다는 것이다. 관건은 풍부한 툴과 라이브러리, 학습 자료 생태계를 갖춘, 광범위하게 사용되며 표준화된 언어(예를 들어 자바스크립트)를 기반으로 하는 로우코드 플랫폼을 구하는 것이다. 자바스크립트는 애플리케이션의 프론트엔드와 백엔드, 양쪽에서 공통 언어를 사용하는 기능을 통해 프론트엔드 개발자가 백 엔드에 자신의 지식과 기술을 전수할 수 있는 추가적인 혜택도 제공한다.

유연한 개방형 툴과 플랫폼, 언어(자바스크립트)는 개발자와 개발자를 채용하는 회사에 모두 이익이 된다. 로우코드 솔루션을 구매할 때는 이 사실을 항상 염두에 두어야 한다. 

Case Study

“속도가 경쟁력” 로우코드 개발의 혁신 가속화 사례

디지탈 혁신을 진행하는 많은 기업이 신규 애플리케이션 및 서비스 제공 속도를 높이는 방법을 모색함에 따라 로우코드(Low Code) 개발 툴이 점차 중요한 IT 요소로 부상하고 있다.

로우코드 개발 플랫폼을 이용하면 전통적인 프로그래밍 방식을 이용하지 않고도 애플리케이션을 작성할 수 있다. GUI(Graphical User Interface)를 이용한 각 요소의 구성 작업만으로 소프트웨어를 개발하는 것이다. 이 접근법의 주된 이점 중 하나는 필요한 직접 코딩의 양을 줄여 비즈니스 애플리케이션 제공 속도를 크게 향상시킬 수 있다는 점이다. 또한 좀더 많은 직원들이 개발 프로세스에 참여할 수 있다.

시장조사 기관인 R&M(Research and Markets)은 글로벌 로우코드 개발 플랫폼 시장의 성장률이 2022년까지 연간 44%에 달할 것으로 전망했다. 2017년의 43억 2,000만 달러에서 2022년에는 272억 3,000만 달러 규모로 성장한다는 설명이다. R&M은 웹 및 모바일 애플리케이션 개발 편의성과 함께 소프트웨어 자동화 및 더욱 혁신적인 애플리케이션에 대한 수요 증가로 모든 규모의 기업들이 로우코드 개발 플랫폼을 도입하고 있다고 진단했다.

포레스터 리서치 또한 전통적인 방법보다 소프트웨어 개발 속도를 10배나 높일 수 있는 플랫폼이 디지털 비즈니스 혁신을 지원하는 데 있어서 핵심 전략으로 등장하고 있다고 분석했다. 개발자와 비 개발자 모두 로우코드 개발 플랫폼을 이용할 수 있으며 특히 약간의 교육을 통해서도 시작할 수 있다는 점이 장점이다. 포레스터 리서치의 부사장 겸 수석 애널리스트 존 라이머는 로우코드 플랫폼이 주로 3가지 방식으로 디지털 비즈니스를 발전시키는 데 도움이 된다고 전했다. 비즈니스 요건을 충족하기 위해 필요한 시간을 크게 단축할 수 있고 ‘새도우 IT’의 힘을 긍정적으로 활용하며 운영 프로세스 자동화에 필수적인 역할을 담당한다는 것이다.

애자일 환경 구축하기

ED(Equity Derivative) 청산 조직인 OCC(Options Clearing Corp)는 산업 규제 증가 등 급격하게 변화하는 금융 시장 환경에 대응하여 2015년 애플(Appian)의 로우코드 플랫폼을 사용하기 시작했다. OCC의 수석 프로세스 혁신 부사장 데니스 크나비안은 “수동 프로세스 및 관리 절차에 대한 통제력을 높임으로써 위험을 완화하고자 했다. 이로 인해 희귀한 코딩 기술 또는

일련의 하드웨어 인수 및 유지보수에 의존하지 않는 더욱 민첩한 기술에 기초한 새로운 접근방식이 필요하게 되었다”라고 말했다.

애플의 소프트웨어는 OCC에 신규 제품 및 서비스 개발, 어음교환 구성 기업의 온보딩, 모델 라이프사이클 관리, 규제 기록, 자재 합 등의 여러 내부 프로세스 관리를 쉽게 해주는 통합 지점을 제공한다. 크나비안은 “더욱 민첩한 플랫폼을 활용할 수 있게 됐다. 새롭거나 변화하는 비즈니스 수요에 신속하게 대응하면서 극대화된 보안을 유지하는 데 도움이 된다”라고 말했다.

특히 애플 플랫폼 내에서 구성 요소의 드래그 & 드롭 기능과 재사용성은 개발 프로세스를 간소화하는 데 도움이 된다. 현재 OCC의 개발 활동은 IT 내의 ETS(Enterprise Technology Solutions)팀이 관리하고 있다. 그러나 직접 사용자 지정을 가능하게 하는 혁신적인 활동 모니터링 애플리케이션은 현업 사용자들이 직접 개발했다.

크나비안은 또 “애플 내에서 직접 문서 및 증거 수집을 중앙에 집중함으로써 감사 프로세스를 간소화했다”고 밝혔다. 이전에는 감사 요청에 대응할 때 팀들이 시간을 들여 여러 이메일 및 스프레드시트에 포함된 배경 정보를 수집해야 했다. 하지만 애플의 애플리케이션을 이용하면 “감사자들이 시스템에 직접 액세스하여 관련된 모든 정보를 더욱 효율적인 방식으로 검증할 수 있다”라는 설명이다.

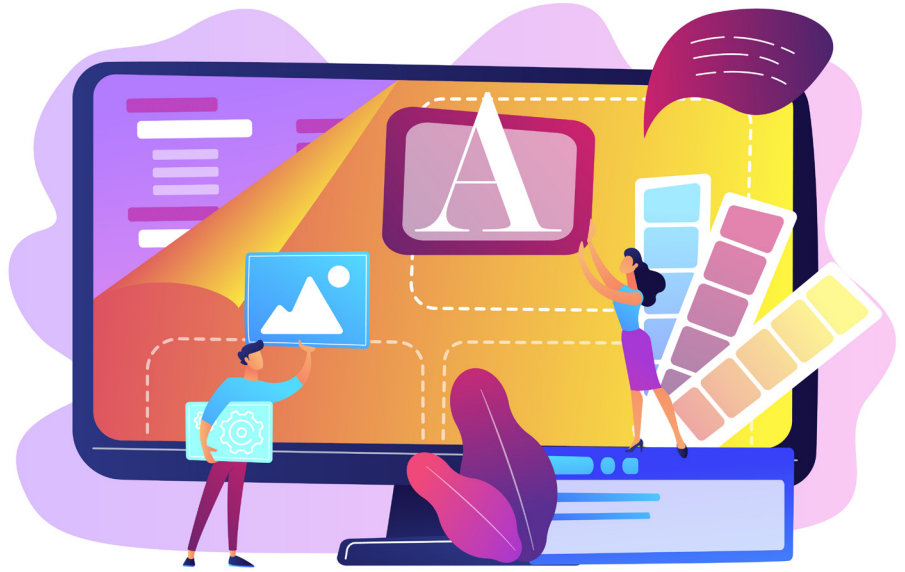
로우코드를 통한 OCC 성공의 핵심 중 하나는 IT 및 비즈니스 운영 구성원들 사이의 강력한 협업이다. 크나비안은 “전략적 디지털 혁신 계획을 실행하기 위해 긴밀히 협력하는 ETS 및 PI(Process Innovation)팀원들의 노력으로 자동화가 달성됐다”라며, IT도 OCC의 기술 인프라 및 전반적인 전략 내에서 실행 가능성을 확보하기 위해 톨 선택 프로세스에 능동적으로 참여한다고 전했다.

OCC는 디지털 혁신을 계속하면서 톨 사용을 증가시킬 계획이다. 현재 해당 플랫폼에서 14개의 프로세스를 자동화했으며, 프로세스를 자동화하고 간소화하는 새로운 방법을 지속적으로 찾고 있다. 크나비안은 “우리는 미래 역량을 위해 사용할 데이터를 능동적으로 구축 및 구성하여 다른 내부 시스템과 통합하면서 효율적인 프로세스를 생성하고 위험을 낮추기 위해 RPA(Robotic Process Automation) 등의 다른 기술을 활용하는 방법을 고려하고 있다”라고 말했다.

앱 공급 가속화

로우코드 톨은 특히 IT 인력과 예산이 제한적인 소기업에 요긴하다. 보조 의료 수당 지불용 소프트웨어 업체인 레드페이(RediPay)가 그렇다. 이 기업은 최소한의 인력과 예산으로 기업용 애플리케이션을 구축할 수 있는 가장 효율적인 경로를 찾아 2017년 1월 아웃시스템즈(OutSystems)의 로우코드 플랫폼을 도입했다. 아웃시스템즈의 소프트웨어를 선택, 시험, 배치하는 작업은 IT, 운영, 재무 사이의 협력 활동을 통해 이뤄졌다.

레드페이는 내부적으로 아웃시스템즈 플랫폼을 이용해 내부 고객 서비스, 영업, 온보딩 애플리케이션을 구축했다. 가장 최근에는 소비자용 모바일 앱, 제공자 웹 앱, 내부/관리자 웹 앱을 완성했다. 현재 레드페이는 해당 플랫폼을 활용하여 매우 노동 집약적이고 비용 소모적인 품질



확보, 시험, 배치를 약간의 시간과 비용으로 달성할 수 있다.

레디페이 CEO 마크 굴리는 “물론 결과는 산업 및 사용례별로 다를 것이다. 하지만 레디페이
의 경우 로우코드 개발이 상당한 생산성 증가로 이어졌다”라고 말했다. 또 “이제 새로운 소프트
웨어 솔루션, 모바일, 웹 등이 필요하면 (로우코드를 통해) 구축할 수 있다”라며, “머지않아 로우
코드 플랫폼이 컨테이너, 인공지능, 머신러닝 등의 개념을 적용하는 가장 효율적인 프레임워크
가 될 것이다”라고 덧붙였다.

굴리에 따르면 현 시점에서 로우코드 플랫폼은 여전히 플랫폼의 잠재적 가능성을 찾고 점
진적으로 활용할 수 있는 교육 받은 개발자가 필요하다. 하지만 로우코드 기술이 발전하면서
비 개발자가 이런 플랫폼을 활용해 영업 및 마케팅 툴과 제품 모형을 개발할 수 있을 것으로 기
대한다.

셰어포인트 솔루션 제공을 쉽게

LMI(Liberty Mutual Insurance)의 자기자본투자 펀드를 관리 사업부 LMI(Liberty Mutual
Investments)는 2010년 여러 비즈니스 프로세스를 지원하는 마이크로소프트 셰어포인트 애
플리케이션을 개발하기 위해 닌텍스 워크플로우(Nintex Workflow)의 로우코드 플랫폼을 도
입했다.

LMI의 수석 기술 엔지니어 스티브 로버트는 “현재 셰어포인트 내에 존재하는 여러 승인 프로
세스를 구축하기 위해 닌텍스를 이용하고 있다. HTML, 자바(Java), C++, 사용 가능한 기타 언어
를 이용해 완전한 기능을 갖춘 웹 애플리케이션을 개발하는 대신에 닌텍스 워크플로우는 내부
비즈니스 파트너의 일상 수요 중 일부를 충족하는 셰어포인트 애플리케이션을 개발할 수 있도
록 간소화된 그래픽 로우코드 솔루션을 제공한다”라고 설명했다. 로버트는 2011년부터 해당
플랫폼을 통해 비즈니스 솔루션 개발에 요구되는 코딩 분량이 감소했다고 전했다.

이렇게 개발된 비즈니스 솔루션 중 하나가 직원 합류 및 퇴사에 대해 단계별 접근방식을 취할

수 있는 사용자 온보딩 및 오프보딩 시스템이다. 이는 사용자, 관리자, IT 인력이 새로운 사용자 계정의 여러 승인, 시의적절한 알림, 액티브 디렉터리 프로비저닝을 가능하게 한다. 로버트는 “단순하고 반복 가능한 워크플로를 생성하여 배치할 수 있는 여러 캘린더에서 사용자 타임 오프 요청을 관리하면서 부서 내의 여러 팀에 솔루션을 제공하고 관리팀에 각 인력을 처리할 수 있는 기능 및 가시적인 수단을 제공할 수 있었다”라고 설명했다.

다양한 개발 환경 지원

물류 서비스 업체인 큐리어 로지스틱스(Courier Logistics)는 2014년부터 방문자 관리 애플리케이션을 위해 조호(Zoho)의 로우코드 툴을 사용하기 시작했다. 현재 큐리어는 이 소프트웨어를 CRM, 창고 관리, 직원 관리, 기타 운영 등의 내부 운영에 사용하고 있다. 큐리어는 다양한 부서가 있고 각각 자체적으로 개발을 처리하곤 했다. 즉 해당 플랫폼을 사용하기 전에는 개발 프로세스를 만족스러운 수준으로 자동화할 수 있는 현실적인 방법이 없었다.

큐리어의 커머셜 관리 책임자 레이첼 매카트니는 “시간이 소요되는 수동 과업을 포함하여 비즈니스 부문 내에서 여러 문제(와 부정확성)가 발생했다. 재무적으로 볼 때 비용을 잘못된 방향으로 투자하고 있었다”라고 말했다. 큐리어는 다양한 개발 애플리케이션을 검토했다. 특히 직원들이 많은 개발 지식이 필요 없는 단순한 형태를 개발할 수 있기를 희망했다. 매카트니는 “부서마다 저마다의 방식으로 작업하고 싶어할 수 있기 때문이다. 로우 코딩을 통해 가능하다고 판단했다”라고 말했다.

창고 관리, CRM, 기타 애플리케이션을 위한 조호 플랫폼 내에서의 프로그램 개발을 통해 큐리어 로지스틱스는 큰 비용을 절감하고 있다. 매카트니는 “비용 절감 외에도 고객 서비스 부서는 전화를 통해 소통하려 시도하는 대신에 운전자와 전자적으로 통신할 수 있게 되면서 약 90%의 추가적인 유입 고객 전화를 처리할 수 있게 되었다”라고 말했다. 