گروه شرکتهای رایان نظم
RayanNazm
Companies Group

LOW - CODE vs. NO – CODE

محقق: مهدی فضلعلی
ترجمه: مهدی فیضلی

منبع : IBM

خرداد ۱۴۰۱

گروه شرکت های رایان نظم

مقدمه

Low-code و No-code دو راه‌حل جدید توسعه نرم‌افزار هستند. تقاضا برای خودکارسازی سطح بالا[1] و مدرن‌سازی فناوری اطلاعات افزایش‌یافته است، اما شرکت‌ها به دلیل محدودیت در دسترس بودن استعدادهای توسعه دهندگان، در تلاش برای همسو شدن با این روندها بوده‌اند. بسیاری از پروژه‌های فن‌آوری اطلاعات به دلیل کمبود منابع با مهارت‌های فنی تخصصی به پرونده "در حال انتظار" بازگردانده می‌شوند. در نتیجه، ناکارآمدی‌های عملیاتی همچنان وجود دارند و زمان بازاریابی که یک عامل حیاتی برای حفظ رقابت در کسب و کارهاست به خطر می‌افتد.

برای پرداختن به این چالش‌ها، راه‌حل‌های توسعه نرم‌افزار با Low-code و No-code به عنوان جایگزین‌های مناسب و قابل دوام برای فرآیند توسعه سنتی ظهور کرده اند.

تفاوت Low-code نسبت به No-code در چیست؟

Low-code چیست؟

Low-code یک رویکرد توسعه کاربردی سریع(RAD) [2] است که تولید کد خودکار را از طریق بلوک‌های سازنده بصری مانند رابط drag-and-drop [3] و pull-down [4] منو ممکن می‌سازد. این خودکارسازی به کاربران Low-code ، این امکان را می‌دهد که به جای برنامه نویسی بر روی استفاده از ابزار تمرکز کنند. Low-code یک زمینه میانی متعادل بین کد نویسی دستی و بدون کد نویسی است زیرا کاربران آن هنوز هم می‌توانند کد را بر روی کد ایجاد شده به صورت خودکار اضافه کنند.

نمونه‌هایی از برنامه‌های کاربردی که برای توسعه از **Low-code** استفاده کرده اند:

- پلت فرم‌های مدیریت فرآیند کسب‌وکار
- توسعه وب سایت و برنامه‌های کاربردی تلفن همراه
- ابزارهای میان دپارتمانی مانند نرم‌افزار مدیریت ارزیابی،
- متصل شونده‌های خارجی
- فن‌آوری‌های بعدی مبتنی بر ابر، مانند کتابخانه‌های یادگیری ماشین، خودکارسازی فرآیند رباتیک و نوسازی
- بازنویسی برنامه های قدیمی.

[1] Hyper automation
[2] Rapid application development
[3] کشیدن و رها کردن
[4] پایین کشیدن

No-code چیست؟

No-code نیز یک رویکرد RAD [۱] است و اغلب به عنوان زیر مجموعه‌ای از پلاگین ماژولار و رویکرد توسعه Low-code در نظر گرفته می‌شود. در حالی که برخی از قابلیت‌ها توسط توسعه‌دهندگان به شکل برنامه‌نویسی یا کدنویسی دستی انجام می‌شود. اما No-code ، رویکردی کاملا بدون استفاده از دست به کد شدن است و با وابستگی ۱۰۰ ٪ به ابزارهای بصری.

نمونه‌هایی از برنامه‌های کاربردی مناسب برای توسعه No-code شامل :

- برنامه‌های خدمات شخصی برای کاربران کسب‌وکار
- داشبوردها
- برنامه‌های تلفن همراه و وب
- پلت فرم‌های مدیریت محتوا
- سازندگان گذرگاه انتقال داده [۲]

No-code برای تولید سریع برنامه های مستقل، رابط های کاربری ساده [۳] و خودکارسازی های ساده [۴] است، و در ابزارهای برنامه‌ریزی تقویم، ابزارهای مدیریت تسهیلات و برنامه‌های گزارش BI با ستون‌ها و فیلترهای قابل تنظیم استفاده می‌شود.

خودکارسازی با Low-code و No-code

یک پلتفرم برنامه کاربردی Low-code (LCAP) [۵] نیز پلتفرم توسعه Low-code (LCDP) [۶] نامیده می‌شود که شامل یک محیط توسعه یکپارچه (IDE) [۷] با ویژگی‌های داخلی مانند API ها، قالب‌های کد، ماژول‌های پلاگین قابل‌استفاده مجدد و اتصالات گرافیکی برای خودکار کردن درصد قابل‌توجهی از فرآیند توسعه برنامه است. پلتفرم های Low-code معمولا به عنوان راه‌حل‌های پلت فرم مبتنی بر ابر [۸] به عنوان یک سرویس (PaaS) [۹] در دسترس هستند. یک پلتفرم Low-code بر روی اصل کاهش پیچیدگی با استفاده از ابزارها و تکنیک‌های بصری مانند مدلسازی فرآیندکار می‌کند، که در آن کاربران از ابزارهای بصری برای تعریف جریان‌های کاری، قوانین کسب‌وکار، رابط‌های کاربر و مانند آن استفاده می‌کنند. در پشت صحنه، گردش کار کامل به طور خودکار به کد تبدیل می‌شود. پلتفرم های Low-code عمدتا توسط توسعه دهندگان حرفه‌ای برای خودکار کردن جنبه‌های عمومی کد نویسی به منظور دسترسی به امکان انجام توسعه های پیچیده تر مورد استفاده قرار می‌گیرد.

---

[۱] Rapid application development
[۲] Data pipeline builders
[۳] Straightforward UIs
[۴] Simple Automation
[۵] Low-code Application Platform
[۶] Low-code development Platform
[۷] Integrated development environment
[۸] Cloud-based
[۹] Platform-as-a-Service

نمونه‌هایی از چنین پلتفرم‌های خودکارسازی عبارتند از :

- پلتفرم‌های کاربردی Low-code [۱]
- مجموعه‌های مدیریت فرآیند کسب‌وکار هوشمند [۲]
- پلتفرم‌های توسعه کاربر نهایی [۳]
- و سایر ابزارهای از این دست.

پلتفرم توسعه ای No-code (NCDP)[۴] ، و یا پلتفرم توسعه خودکارسازی کاربر نهایی (CADP) [۵] نیز نامیده می‌شود. تمامی کدها از طریق رابط‌های drag-and-drop و یا point-and-click تولید می‌شوند.

پلتفرم های توسعه ای No-code توسط توسعه دهندگان حرفه‌ای و کاربران نهایی استفاده می‌شود (کاربران غیر فنی یا غیر توسعه دهندگان با مهارت‌های کد نویسی محدود یا بدون مهارت).

Low-code و No-code: شباهت‌ها و مزایا

هر دو مشابه هم هستند به این دلیل که هدف آن‌ها خلاصه و ساده کردن جنبه‌های پیچیده کد نویسی با استفاده از رابط‌های بصری و الگوهای از پیش پیکربندی شده می‌باشد. هر دو پلت فرم‌های توسعه به عنوان راه‌حل‌های PaaS در دسترس هستند و یک طراحی مبتنی بر گردش کار را برای تعریف پیشرفت منطقی داده‌ها اتخاذ می‌کنند. مزایای بسیاری را به دلیل رویکرد مشترکشان به اشتراک می‌گذارند:

- **دموکراتیکه کردن تکنولوژی**: راه‌حل‌های Low-code و No-code با هدف توانمندسازی انواع مختلف کاربران ساخته می‌شوند. این امر وابستگی به متخصصان و تکنولوژیست های گران قیمت را کاهش می‌دهد.
- **توانمندسازهای بهره‌وری**: No-code / Low-code سرعت توسعه را افزایش می‌دهد، بک لاگ ها را حذف میکند، زمان بندی پروژه را از ماه ها به روزها کاهش می دهد و عرضه سریع تر محصول را تسهیل می کند.
- **بازخورد سریع مشتری با ریسک پایین تر**: قبل از سرمایه‌گذاری منابع قابل‌توجه در یک پروژه، Low-code / No-code به توسعه دهندگان اجازه می‌دهد تا با نشان دادن نمونه‌های اولیه ساخت آسان، از مشتریان بازخورد بگیرند. این مساله در زمانبندی پروژه تصمیم برای شروع / عدم شروع را تغییر می‌دهد، ریسک و هزینه را به حداقل می‌رساند.

---

[۱] Low-code application platforms
[۲] Intelligent business process management suites
[۳] Citizen development platforms
[۴] No-code development Platform
[۵] Citizen Automation development platforms

- **ساخت بیشتر، خرید کمتر**: در حالی که محصولات تجاری - آماده (COTS)[1] می‌توانند گران باشند و یک رویکرد یک سایز متناسب برای همه[2] داشته باشند، No-code / Low-code سفارشی سازی داخلی را تحریک می‌کنند، و در دوراهی بین ساخت یا خرید، جهت را به سمت "ساخت" تغییر می‌دهند.
- **ثبات در معماری (معماری سازگار)**: برای ماژول های مقطعی مانند ثبت وقایع و حسابرسی، یک پلتفرم No-code / Low-code متمرکز، طراحی و سازگاری کد را تضمین می‌کند. این یکنواختی در حالی مفید است که برای برنامه‌های ایراد یابی و رفع خطا [3] نیز مفید هستند، زیرا توسعه دهندگان می‌توانند به جای درک چارچوب‌ها، زمان خود را صرف حل مشکلات کنند.
- **مقرون‌به‌صرفه بودن**: No-code / Low-code به دلیل تیم‌های کوچک‌تر، منابع کم‌تر، هزینه‌های زیرساخت کم‌تر و هزینه‌های نگهداری کم‌تر، نسبت به توسعه دستی(سنتی) مقرون‌به‌صرفه است. همچنین منجر به ROI [4](بازگشت سرمایه) بهتر با انتشار سریعتر و چابک می‌شود.
- **هم‌کاری بین کسب‌وکار و IT**: تیم‌های کسب‌وکار و توسعه به طور سنتی یک رابطه فشار - کشش[5] را به اشتراک گذاشته‌اند. با این حال، با مشارکت بیشتر کاربران کسب‌وکار در توسعه از طریق حرکت به سمت No-code / Low-code، تعادل و درک بهتری بین دو دنیای به ظاهر متفاوت وجود دارد.

Low-code چگونه از No-code متفاوت است؟

علی‌رغم تفاوت‌ها و ویژگی ظریف بین راه‌حل‌های این دو، همپوشانی زیادی بین این دو رویکرد وجود دارد (که با موقعیت های گیج‌کننده فروشندگان تشدید می‌شود) . با این حال، تفاوت‌های مهمی وجود دارند که باید در نظر گرفته شوند:

۱) کاربران هدف

Low-code توسعه دهندگان حرفه‌ای را هدف قرارداده تا از تکرار کد پایه اجتناب کنند و نیز برای جنبه های پیچیده تر فضای بهتری ایجاد نمایند و درگیر توسعه ای شوند که منجر به نوآوری و غنای مجموعه ویژگی ها می شود. با خودکار کردن جنبه‌های استاندارد کد نویسی و اتخاذ یک رویکرد syntax-agnostic ، توسعه دهنده را قادر می‌سازد که دوباره استعداد هایش را گسترش و توسعه دهد.

از سوی دیگر، هدف رویکرد **No-code**، کاربران کسب و کاری است که دامنه ی دانش وسیعی دارند و ممکن است اندکی آگاه به فن‌آوری باشند اما توانایی کد نویسی (نوشتن دستی کد) را ندارند. همچنین برای تیم‌های ترکیبی با کاربران کسب‌وکار و توسعه دهندگان نرم‌افزار یا صاحبان کسب‌وکار کوچک و تیم‌های غیر IT، مانند منابع انسانی، مالی و حقوقی خوب است.

---

[1] Commercial-off-the-shelf
[2] One-Size-Fits-All
[3] Debugging applications
[4] Return on Investment
[5] Push-Pull

۲) موارد استفاده

رویکرد **No-code** خودش را به خوبی در اختیار برنامه‌های کاربردی فرانت[۱] قرار می‌دهد که می‌توانند به سرعت با رابط drag-and-drop طراحی شوند. کاندیدهای خوب برنامه‌های کاربردی رابط کاربری هستند که داده‌ها را از منابع استخراج ،گزارش، تجزیه و تحلیل، وارد و خارج می‌کنند.

همچنین، **No-code** برای جایگزین کردن کارهای یکنواخت اداری مانند گزارش‌های مبتنی بر اکسل که توسط تیم‌های کسب‌وکار استفاده می‌شوند، ایده‌آل است. چنین پروژه‌هایی به راحتی توسط IT اولویت‌بندی نمی‌شوند، اما می‌توانند برای تیم‌های کسب‌وکار به عنوان یک ناجی باشند. همچنین برای نرم افزار های داخلی که بار عملکردهای گسترده ای را بر دوش نمی کشند و همچنین برنامه های تجاری در مقیاس کوچک با بودجه توسعه کم مناسب است.

Low-code ، با یک کتابخانه مولفه جامع، می‌تواند به کاربردهایی با منطق تجاری سنگین تعمیم داده شود و در سطح سازمان درجه‌بندی و رتبه بندی شود. همچنین، برای ادغام با سایر برنامه‌ها و API های خارجی، به چندین منبع داده متصل شده و سیستم‌هایی با محافظ های امنیتی که به نظارت های IT نیاز دارند ایجاد کنید، Low-code جایگزین بهتری نسبت به No-code است.

۳) سرعت (تندی)

Low-code به آموزش و زمان بیشتری برای نصب، توسعه و استقرار نیاز دارد زیرا فرصت‌های بیشتری برای سفارشی سازی فراهم می‌کند. اما هنوز هم بسیار سریعتر از توسعه سنتی است.

No-code، با قابلیت پیکربندی بالا با همه ی پلاگین ها، زمان کمتری برای ساخت در مقایسه با **Low-code** نیاز دارد. زمان تست نیز کاهش می‌یابد زیرا پایین ترین نرخ ریسک که معمولاً با کد نویسی دستی ایجاد برای خطاهای احتمالی وجود خواهد داشت. در اینجا، همه چیز برای اطمینان از این است که تنظیمات و جریان داده‌ها به درستی تنظیم شده باشد.

۴) سیستم های باز[۲] در مقابل سیستم های بسته [۳]

Low-code یک سیستم باز است که به کاربران اجازه می‌دهد تا عملکرد خود را از طریق کد گسترش دهند. این به معنای انعطاف‌پذیری بیشتر و قابلیت استفاده مجدد است. به عنوان مثال، کاربران می‌توانند پلاگین های سفارشی و ارتباطات منبع داده را ایجاد کنند تا موارد استفاده خود را تطبیق دهند و بعدا از آن‌ها استفاده کنند. اما شایان‌ذکر است که به روزرسانی ها و بخش‌های جدیدتر LCAP[۴] باید با کدِ معرفی‌شده دستی آزمایش شوند.

---

۱ Front-end apps
۲ Open systems
۳ Closed systems
۴ پلتفرم‌های کاربردی با کد نویسی کم

No-code، سیستم بسته تری است که تنها از طریق مجموعه ویژگی‌های قالبی[1]، قابل تعمیم و گسترش است. این به معنای موارد استفاده محدود و دسترسی به پلاگین ها وتکرار هاست، اما اطمینان از سازگاری رو به عقب آسان‌تر است زیرا هیچ کد نوشتاری دستی وجود ندارد که بتواند نسخه‌های آینده NCDP [2] را خراب کند.

۵) سایه ریسک IT

در حالی که این موضوع هم برای پلتفرم‌های No-code و هم برای پلتفرم‌های با Low-code نگران‌کننده است، ریسک سایه IT برای No-code بالاتر است،چراکه نیاز به مداخله کم یا تقریبا بدون مداخله تیم‌های IT دارد. این امر می‌تواند منجر به یک زیرساخت موازی شود که به طور دقیق تحت نظارت نیست و منجر به آسیب‌پذیری‌های امنیتی و فنی می‌شود.

با این حال، این واقعیت که Low-code هنوز زیر مجموعه تیم‌های IT است، می‌تواند به تضمین نظارت و کنترل بهتر کمک کند.

۶) محدوده معماری

امتیازات Low-code بیش از No-code در پشتیبانی از مقیاس پذیری و سازگاری بین پلتفرمی است. اضافه کردن پلاگین ها و کدنویسی سفارشی، امکان پیاده سازی گسترده تری را فراهم می‌کند و کار با چندین پلتفرم را فراهم می آورد.

No-code توسعه پذیری کمتر و پتانسیل محدودی در اتصال به سیستم های قدیمی یا ادغام با سایر پلتفرم ها دارد بنابراین، مجموعه محدودی از موارد استفاده را مورد توجه قرار می دهد و توانایی کاهش مقیاس را دارد.

[1] Templated feature sets
[2] پلتفرم‌های توسعه ای بدون کد نویسی

چه موقع از Low-code در مقابل No-code استفاده کنید؟

هم Low-code و هم No-code، نقاط قوت منحصر به فرد خود را دارند. شباهت‌هایی نیز بین این دو وجود دارد که تصمیم گیری را سخت میکند. بهترین راه پیش رو ارزیابی نیازهای فعلی و انتخاب براساس آن است.

در اینجا چند سوال برای تعیین نیازهای کاربر آمده است:

- اهداف استفاده از نرم‌افزار Low-code یا No-code چیست؟
- کاربران چه کسانی هستند؟ تخصص برنامه‌نویسی آن‌ها چیست؟
- دامنه و مقیاس این مشکل چیست که باید حل شود؟
- آیا ساخت، نیاز به یکپارچه سازی های سفارشی با نرم افزار های بیرونی و داخلی دارد؟
- زمان مورد نیاز ساخت چقدر است؟
- کاربران چقدر می خواهند روی کد ها کنترل داشته باشند؟
- آیا برنامه باید با داده های محرمانه سروکار داشته باشد یا ملاحظات امنیتی را در دستور لحاظ کند؟

دو سوال کلیدی در اینجا عبارتند از:

1) **چه برنامه‌ای برای آن وجود دارد؟**
2) **و چه کسی آن را خواهد ساخت؟**

در حالی که هر دوی این سوالات مهم هستند، بهتر است از رویکرد هدف محور نسبت به رویکرد کاربر محور استفاده شود. اگر موارد استفاده پیچیده باشند، به ادغام با دیگر برنامه‌های در محل یا برنامه‌های ابری نیاز داشته باشند، نیاز به مواجهه با مشتری یا نیازهای حیاتی کسب‌وکار داشته باشند یا نیاز به استقرار در سراسر شرکت داشته باشند، **Low-code** ارجح تراست.

در این مورد، حتی اگر کاربران تخصص لازم را در زبان‌های برنامه‌نویسی نداشته باشند، مشارکت با تیم‌های IT یا برنامه‌های آموزشی می‌تواند چالش‌ها را حل کند.

منبع:

https://www.ibm.com/cloud/blog/low-code-vs-no-code