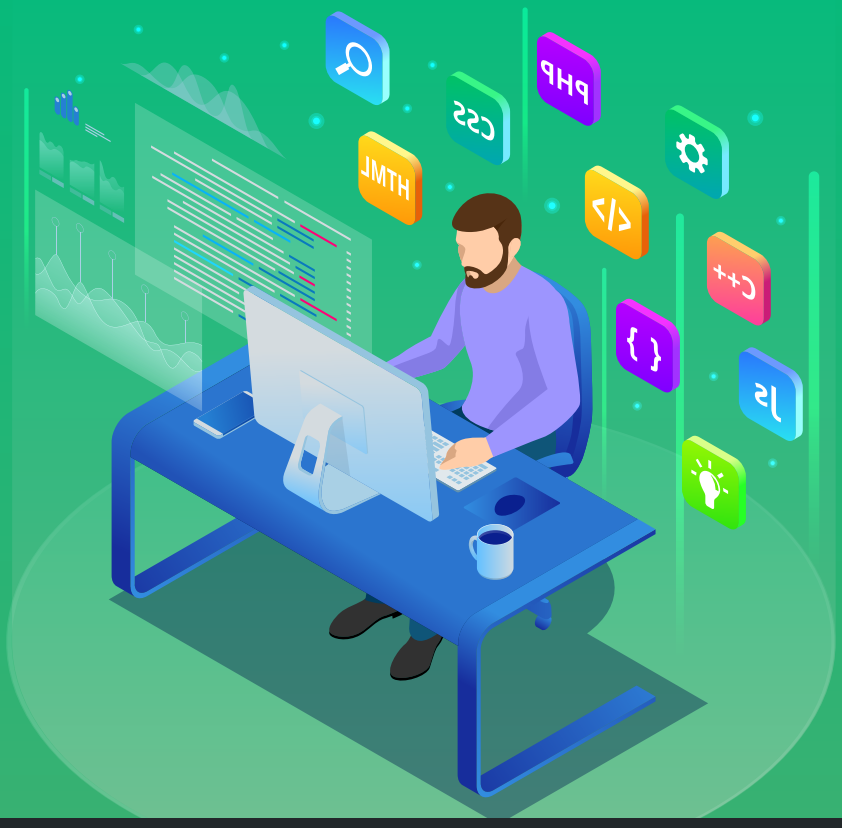
LANSA®

An Idera, Inc. Company

The Power of Writing Code in a Low-Code Solution



Introduction

When it comes to creating your business applications, there are hundreds of coding platforms and programming languages to choose from. These options range from very complex, traditional programming languages to low-code platforms where sometimes no traditional coding experience is needed. Low-code platforms range from not using a programming language at all to using a very detailed programming language to supplement the low-code functionality. A detailed guide for knowing the difference in the platforms and when to use which one can be found here. Traditional programming languages provide great power and flexibility but at the cost of complexity and time. Enterprises are looking for a faster way to get new applications to market, and that's where low-code development platforms come in.

Low-code platforms can be divided into two categories: model based and language based. Both platforms provide great benefits for rapid application development. Low-code's greatest strength is providing developers a quick way to start developing applications. It provides a quick and easy starting point which then can be further enhanced by additional platform options. Low-code provides the initial creation of an application and handles all of the initial "heavy setup" so the developer can focus on creating the best solution to the business need. After the initial application creation and quick starting point is where model- and language-based low-code platforms start to differentiate themselves. Both approaches use similar theories but vastly differ in how they allow the application to be completed. In this paper, we will discuss the importance of being able to write code in a low-code solution.

Having to Leave the Integrated Development Environment

Model-based low-code platforms often limit application creation to features available inside the low-code platform's integrated development environment, or IDE. If the platform's IDE does not offer a solution to the problem, a developer must leave the IDE and create a workaround in another programming language. Once that is done, then a connector must be created to link the workaround to the low-code solution. If a developer has to leave the platform's IDE to write code in another IDE or language, then the promise of faster application creation is diminished. In fact, this approach actually adds time and complexity to troubleshooting and maintaining applications since the entire application is not accessible in one environment. The most common reasons developers have to leave model-based low-code IDEs are integration and complex logic processes.

Issues with Integration

In almost every enterprise application being created today, there is a need to integrate with at least one other system or specialized software package. Most model-based low-code platforms come with limited native connectivity for these special integration needs. Their cloud-based approach can make integrating with on-premises applications much more difficult, especially when the on-premises applications are older and not designed for cloud integration. If the low-code platform does not provide a built-in connector, then a special integration solution outside the IDE will need to be created. Customized integration is another example where developers must leave the low-code platform's IDE to create a solution in another programming language.

Complex Logic Processes

Model-based low-code is centered around creating workflows in a drag-and-drop graphical interface. The developer can create processes and logical decisions only if the low-code platform has an icon to do so. Anything the developer needs to accomplish that isn't available in the IDE cannot be completed with that particular platform alone. Imagine using a really complex coding solution that needs several different types of connections to several various external resources and then having to make some processing decisions based on what each resource returned, only to find that each resource returned data in a different format, which then needs to be reformatted into a common format. Trying to solve such a complex issue by dragging and dropping icons in a graphical interface would lead to a very convoluted solution, and you probably couldn't do it all without extensive coding outside the low-code's IDE. Professional developers love to code, and writing code to solve problems is how they prefer to handle complex situations. Model-based low-code platforms severely limit how developers can solve complex problems. Without a way to natively handle complex processes, model-based low-code platforms are not capable of handling large enterprise applications.

Locked-In to Low-Code, Locked-Out of Technology

Most low-code platforms run on specific web technologies and are hosted in a cloud environment. Creating applications on the latest release of the low-code platform probably offers the best performance with the latest technologies. But as web and mobile technologies change, the applications created a few years ago could be dated and run suboptimally. As new, major platform version releases occur, applications created with older versions may no longer be able to be upgraded. Also, since those applications were created with a specific platform, they can't be migrated to a different low-code platform. There is a very serious risk of creating applications that may need to be coded again using a different platform or modified to run on an upgraded version of the current low-code platform. A major consideration when selecting a low-code platform is the platform's ability to change with technology and how those changes will impact applications coded on older versions.

Traditional Coding

When it comes to full control over an application, traditional coding languages still allow the greatest flexibility and design control. Often times, especially in web design, more than one programming language must be used to complete just one application or website.

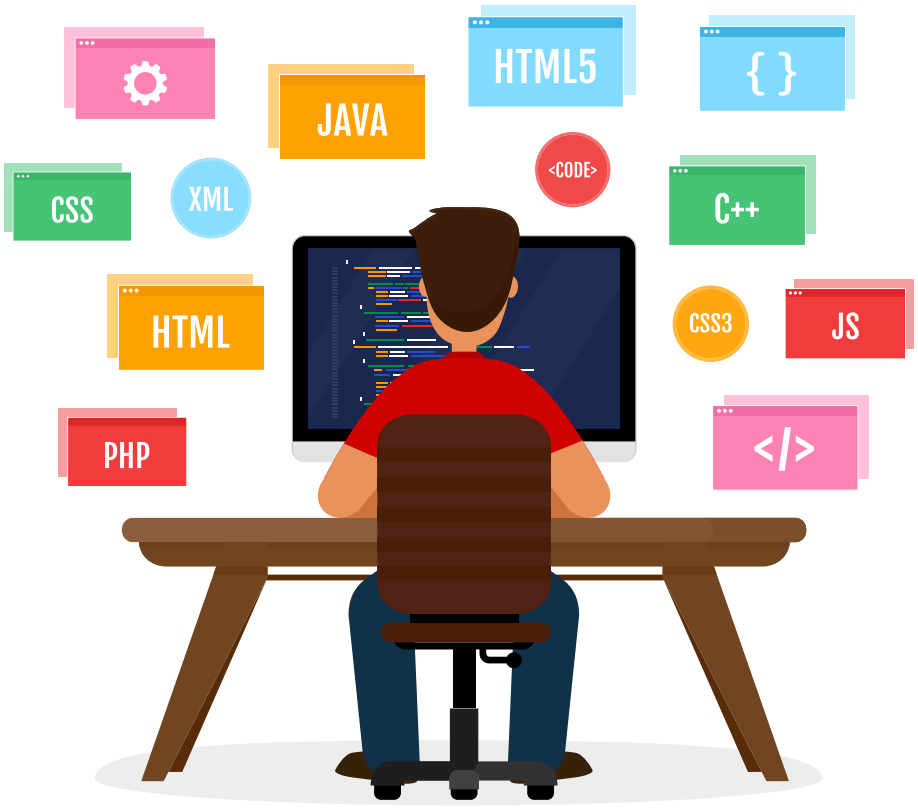
Following are languages used for a typical web application, with alternative languages in parentheses.

Front end: HTML5, CSS, and Javascript (AngularJS or React)

Back end: Python (PHP, Ruby, Ruby on Rails, Perl)

Database: MySQL (MongoDB)

Also, the latest and greatest programming language changes often, many times within just a few years. This means programmers creating web applications in traditional languages must keep learning several new languages every few years. It can be overwhelming to keep up with the fast-paced changes in web languages when the hottest trends may not necessarily be the best for the business's needs.



Visual LANSA: The Solution to Low-Code's Biggest Drawbacks

Visual LANSA is a language-based low-code solution — in fact, one could actually make a case that it is the first enterprise-ready midcode solution. Visual LANSA combines the flexibility and control of traditional programming with the benefits and concepts of low-code principles. With Visual LANSA, you can stay inside the IDE to create a solution for all your complex processes and custom integration needs. A developer can learn one language and start creating enterprise applications at the speed of low-code.

Visual LANSA has its own programming language, which can be used inside the IDE. Adding even greater flexibility, the same programming language can be used for server-side and client-side applications and objects. In traditional coding, a developer may use PHP as the server-side language and a combination of Javascript and HTML/CSS for the client side. In model-based low-code, a developer might create the client-side portion in the low-code IDE and then have to code a separate connector depending on how complex the data retrieval is. With Visual LANSA, the entire application is coded in one IDE using one language. It really is one language and one platform for creating any application whether you're deploying to mobile, web, server, or local.

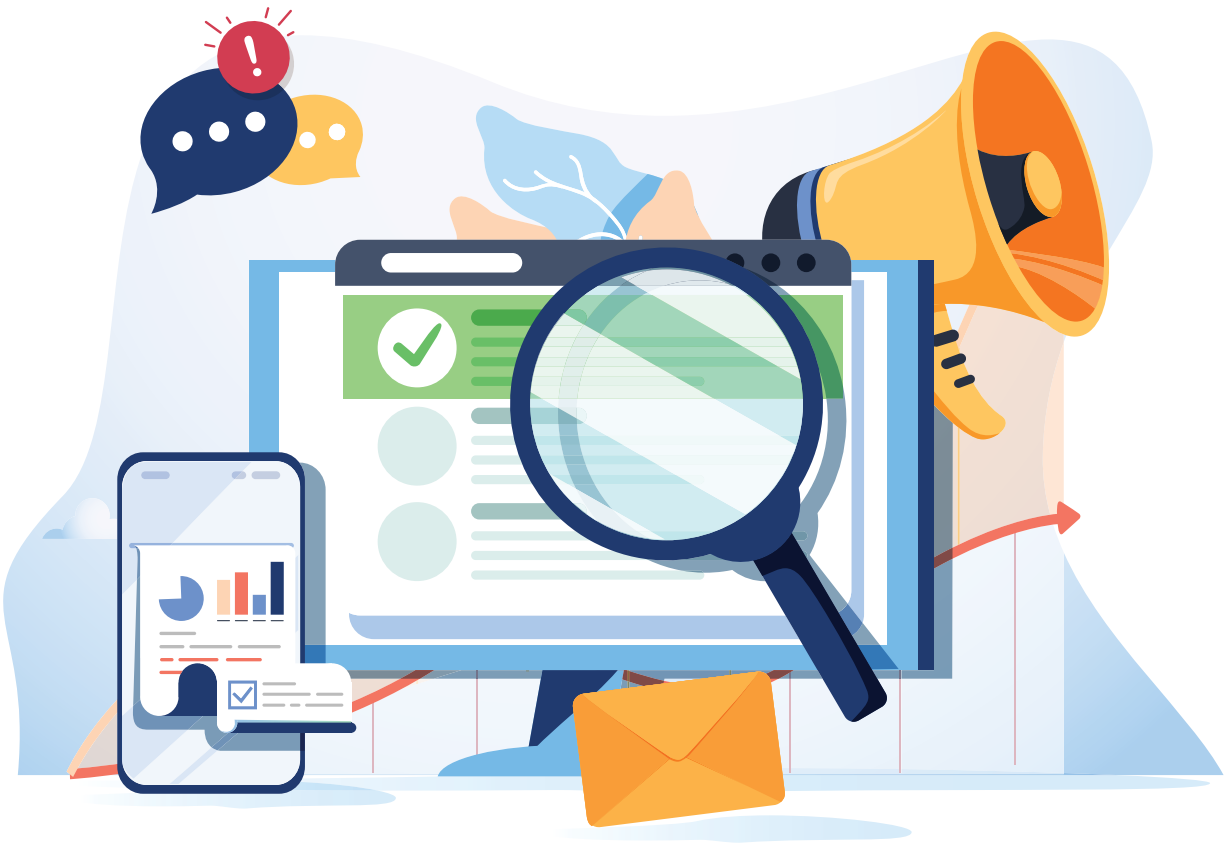
Visual LANSA increases the speed of the application-creation process over traditional coding by automatically creating lines of code simply by answering a few questions. The answers to those questions allow Visual LANSA to automatically create a server module or reusable part that then can be referenced by any Visual LANSA application. So not only do developers have to write less code, but they can create programs, functions, and modules that can be reused in other applications. The same process is followed for creating entire enterprise frameworks and web applications: answer a few questions and the structure and layout are completed for you. By using Visual LANSA, developers will write nearly 10x less lines of code compared to traditional coding but still have the ability to code complex solutions, which other low-code platforms simply cannot do.

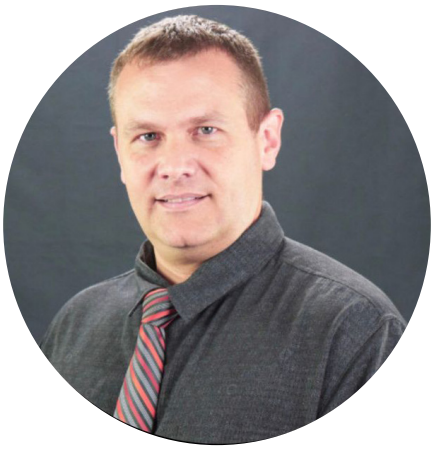
As web technology changes and new, more efficient programming languages come into use, Visual LANSA applications will remain unaffected. Visual LANSA separates the coding language from the underlying technology that runs the applications. So Visual LANSA can easily keep up with web technology without ever impacting the developers. The code for how a Visual LANSA application will run on the web can change, but the underlying code that executes the processes and logic will remain unaffected. This design model ensures that Visual LANSA is future proof and eases any fears that Visual LANSA applications will be outdated years down the road.

Visual LANSA can be a cloud-based solution if you so desire, but Visual LANSA can also be installed on-premises, whether it is on a Windows server or an IBM i. In fact, Visual LANSA is the only low-code solution that can be installed locally on an IBM i. With the flexibility to install Visual LANSA in house and run applications locally, you never have to worry about losing cloud service should a platform fold or stop being supported. Visual LANSA ensures that your applications will run well into the future without worry.

Summary

Being able to write code in a low-code solution is one of the most powerful application-creation processes available to developers today. Having the benefits of low-code's quick application creation process along with the benefits of traditional programming's ability to solve complex integrations and problems means developers can be more effective and efficient. Visual LANSA is the only low-code solution that allows developers to write code inside the IDE. This allows Visual LANSA developers to focus on solving business needs as opposed to trying to figure out workarounds for the platform's limitations. Model-based low-code can be beneficial when simple applications are needed, but when building enterprise applications, language-based low-code is the most efficient and effective method of application creation.





Anthony Graham

Product Manager at LANSA's Product Center (LPC)

About the Author

As Product Marketing Manager, Tony draws on his experience as a former Information Systems Manager where he was responsible for developing enterprise applications. His broad range of experience includes leading a digital transformation initiative that included rewriting a custom legacy ERP system using the Visual LANSA low-code platform. Tony also specializes in software integration and has created custom data collection software for the manufacturing industry

LANSA's mission is to make advanced software simple. We do this by taking care of the underlying and constantly changing technologies, enabling software developers to focus on the business problems that need solving and to rapidly produce high quality software.

When businesses can effectively capitalize on IT to innovate and differentiate, they gain a competitive edge. LANSA helps to make it happen.

LANSA®
An Idera, Inc. Company

THE AMERICAS
HQ Chicago, USA
Tel +1 630 874 7000
Email info@lansa.com

EUROPE
HQ London, UK
Tel +44 1727 790300
Email info@lansa.co.uk

ASIA PACIFIC
HQ Sydney, Australia
Tel +61 2 8907 0200
Email info@lansa.com.au