LEARNING MADE EASY

VeriTran Special Edition

# Low-Code

for dummies®
A Wiley Brand

Dispel the confusion around low-code

Learn the business drivers for low-code

Apply low-code in enterprise scenarios

Compliments of

VeriTran™

Jason Bloomberg

## About VeriTran

VeriTran is a global company focused on driving digital transformation through its Low-Code Platform, which streamlines the creation, development, and roll-out of digital business applications. VeriTran provides the technology that enterprises need in order to innovate their digital capabilities. VeriTran's Low-Code Platform integrates emerging technology into legacy systems, safely and securely improving deployment times and delivery costs. As a result, clients can execute in months what often takes years.

For more information visit **www.veritran.com**.

# Low-Code

VeriTran Special Edition

**by Jason Bloomberg**

for dummies®
A Wiley Brand

# Low-Code For Dummies®, VeriTran Special Edition

Published by
**John Wiley & Sons, Inc.**
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com



ISBN: 978-1-119-65449-0 (pbk); ISBN: 978-1-119-65451-3 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

## Publisher's Acknowledgments

Some of the people who helped bring this book to market include the following:

**Project Editor:** Carrie Burchfield-Leighton

**Editorial Manager:** Rev Mengle

**Acquisitions Editor:** Steve Hayes

**Business Development Representative:** Yemily Lopez

# Table of Contents

# Introduction

Marc Andreessen once said that software was eating the world, and today, that trend hasn't stopped. From our phones to our cars to every aspect of our businesses, software has become the lifeblood of our existence. Given the ubiquity and importance of this intangible commodity, it's remarkable — appalling even — that every piece of software, every application on this planet was written *by hand. One line at a time. By a human being.*

Not only is it folly to expect handmade creations to eat the world, but there's simply no way for people to keep up. There aren't enough software developers today to meet the incessant demand for code — and the problem is only going to get worse.

We need a way to build better software more quickly — to leverage the expertise of both technical and business-oriented people to create the applications that run our increasingly digital world.

Fortunately, we have an answer: platforms and tools that empower a wide range of individuals to put together applications with little to no hand-coding, resulting in higher quality software, faster.

That answer is *low-code*.

## About This Book

*Low-Code For Dummies*, VeriTran Special Edition, helps you understand how to build better applications faster. In this book, there's something here for professional developers, as well as people in any business role. Software will impact everyone's role — and empowering anyone in the organization to create software is essential to meeting the competitive and customer pressures of the digital era.

# Icons Used in This Book

I use special icons throughout this book to call attention to particular information. The following icons are the ones I use in this book.

REMEMBER

How can you forget anything in this book, seriously? Just in case, Remember icons represent a few things that are especially important to remember.

WARNING

This information gives you the heads up that there are risks ahead — and you should pay attention to avoid any potential pitfalls.

TECHNICAL STUFF

Sometimes I need to get geeky to explain something. The good news: There's not much technical stuff in this book, and you won't miss much by skipping it.

TIP

This entire book is jam-packed with useful advice, of course. I call special attention to the really juicy stuff with the Tip icon.

IN THIS CHAPTER

- Understanding the basics of low-code
- Comparing low-code and no-code
- Checking out low-code platforms
- Learning where to start

# Chapter 1
# Dispelling the Confusion around Low-Code

When you boil it down to the basics, all software consists of a sequence of ones and zeroes that instructs computers what to do. The act of creating software, therefore, is really nothing but stringing those ones and zeroes together. Of course, it's been a while since people have had to deal with those binary digits manually. For many decades now, creating software has meant writing programs by typing out line after line of instructions in one programming language or another. I call this laborious, error-prone approach to building applications *hand-coding*. Unlike most other products we use today, either as consumers or in business, software is still created by hand — by specialists who hand-code it one line at a time. You might think, "Well, there's got to be a better way." And there is.

In this chapter, I introduce you to what low-code is, the basics behind it, and help you understand the difference between low-code and no-code. I also describe the two types of low-code platforms. After that, you're well on your way in your low-code journey.

# Introducing Low-Code

So let's say that, instead of pounding out software code line-by-line, you had a tool that enabled you to build an application without writing any code. Instead, you might begin with a template, as shown in Figure 1-1. From there, you drag and drop bits and pieces of your app onto the template. Then you might double-click on this bit or that, and go through a wizard to tell the component what it's supposed to do. Voila! You just created your own software without writing a line of code. Welcome to the world of *low-code.*


**FIGURE 1-1:** Building an application from a template.

## Low-Code versus No-Code

There's an old saying in application development shops: *Coders love to code*. And to be sure, many of them do enjoy the practice of hand-coding. However, there is more to creating applications than pecking out lines of code.

Any software developer worth his salt these days is focused more on the value an application provides than the process for creating it. Will it help customers or employees? Will it add to the bottom line? Is it a part of a critically important digital transformation strategy?

Low-code can help. Even for the savviest of developers, low-code takes the grunt work off their plates, empowering them to provide greater value to their organizations and in the end, to their customers than hand-coding can — and more quickly as well.

If you've been poking around the Internet doing research on low-code, one of the first things you may notice is that sometimes people use the term *no-code*. Low-code and no-code are related terms, but in fact, they're quite different in meaning. Nevertheless, there's a bit of confusion about these terms, and in some cases, people use them interchangeably.

REMEMBER

Fundamentally, *low-code* platforms and tools simplify and streamline the work of professional developers, enabling them to deliver enterprise apps in a fraction of the time of hand-coding, often with higher quality. Meanwhile, *no-code* platforms and tools give non-technical businesspeople the ability to assemble simple business apps with minimal involvement from IT. These people are called *citizen developers.* These folks are generally non-technical line-of-business personnel.

However, these categorizations are shifting, as no-code platforms become increasingly sophisticated and low-code platforms become simpler to use.

One example that illustrates this convergence is shown in Figure 1-2. This screenshot illustrates assigning a theme to a mobile interface — a task that a citizen developer using a no-code tool could easily handle. But this example uses a low-code tool — one simple enough that there's really no way to tell the difference.

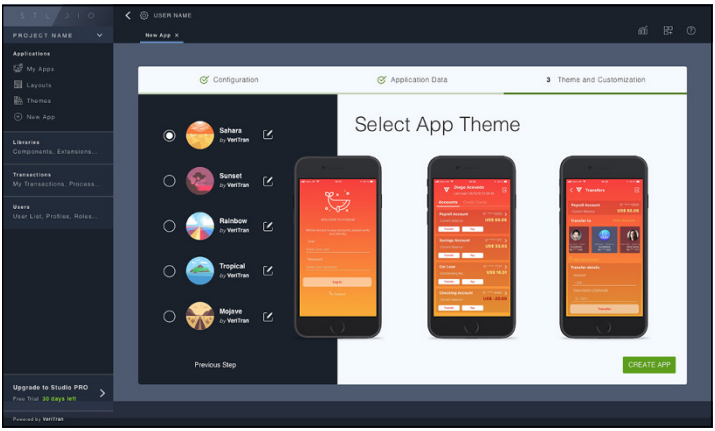


**FIGURE 1-2:** Building a simple mobile interface using a low-code tool.

You may think the *no-code* terminology means that you never have to write any code whatsoever when using such a tool. In contrast, the low-code language suggests that there may still be reasons to write a line or two of code here and there.

REMEMBER

Many of today's no-code tools do allow some coding, although it's rarely if ever necessary. And true to form, people can use most of today's low-code tools to build perfectly fine applications without writing a single line of code. But many of today's low-code platforms are also becoming so simple to use that you don't need to be a professional developer to build an app with one of them — although having some serious software engineering or architecture background can still be quite helpful.

TIP

If you consider yourself to be a citizen developer, then either no-code or low-code may be right for you, depending on the complexity of the applications you want to build, as well as the makeup of your team. Regardless of whether you use a no-code or low-code tool, the benefits to your organization are similar: You're able to build applications faster than if you hand-coded them — sometimes *much* faster — and the resulting apps will have higher quality. And all that human error that results from hand-coding? Gone.

Does this combination of speed and quality seem too good to be true? After all, don't enterprises need professional developers to hand-code *real* applications? It's true that earlier generations

of low-code platforms weren't up to the task of building fully-featured enterprise applications. Fortunately, these tools have evolved quickly.

REMEMBER

Today, everything has changed. Modern enterprise low-code platforms can tackle the toughest of modern enterprise applications — from complex, multicloud workflows to elaborate application modernization initiatives. Not every low-code platform on the market can do it all, but there are few applications any enterprise may want to create that are out of the realm of leading enterprise low-code technologies today.

## Distinguishing Low-Code Platforms

Up to this point, I lump *platforms* and *tools* when talking about low-code. In truth, the offerings in this market often combine tools for building applications as well as platforms for running them. But the primary distinction among such low-code offerings isn't whether they have platforms attached; it's how those platforms work.

Some low-code tools automatically generate applications that in essence constitute programming code — code that you may write by hand. Only the tool automates the writing process for you. In such situations, you can run that code anywhere you could run any other kind of code you may create by hand. The low-code offering may include such a platform out of the box, but if you'd rather use a different one, that option is usually available to you.

Other low-code tools don't actually generate running code at all. They essentially generate a *description* of how your application is supposed to behave and then the platform is able to take that description and execute it as though it was code — even though it isn't.

The first way that low-code platforms work is the *code generation* approach. Vendors of this approach provide a visual application development environment that simplifies the creation of the application, but after it's finished, the platform generates executable code — either editable source code, or in some cases, bytecode that runs on a Java Virtual Machine (JVM) or in a Microsoft Common Language Runtime (CLR) environment.

TIP

One advantage of the code generation approach is that the applications it creates can run independently of the platform. Code generation is necessary for creating native mobile apps, including apps that can run while disconnected from the Internet.

In the other corner of this boxing match are vendors that take a *model-driven execution platform* approach. In this case, the visual application creation environment generates an intermediate domain-specific representation of the application in question and then the platform interprets and executes it directly at runtime.

Model-driven execution platforms typically run in a public cloud, although a few vendors offer private cloud and on-premises versions of their platforms as well. As such, applications can run as soon as the developer assembles a few pieces of the app, which simplifies and speeds up development, testing, and interactions with stakeholders.

REMEMBER

Some model-driven execution platforms can't create standalone native mobile apps. Such platforms are generally well-suited for implementing mobile web apps that run inside browsers on the mobile device, and they often leverage a downloadable app on mobile devices that serves the same purpose.

Other low-code vendors (including VeriTran) provide a downloadable container runtime that allows mobile devices to run the apps, even when the device is disconnected from the Internet. This container runtime is able to access all the special features of the device (camera, accelerometer, and so on). In addition, the application creator can bundle the container runtime and the application into a single file that end-users can download from an app store.

The essential takeaway: Low-code platforms that are model-driven execution platforms require you to run your apps on their platform, which typically runs in the cloud (either the vendor's or your own), or on-premises in your own data center. The main advantage of this approach is that you can rely on the vendor to take care of the platform. If it needs a security patch, or if it requires some tweak to run at peak performance, the vendor takes care of it for you.

The downside is that you're forever locked into your vendor of choice, while the code generation approach allows you to take your application and give the vendor the boot — or perhaps continue to run it even if the vendor goes out of business.


TIP

For mobile applications, the code generation approach is the more straightforward of the two because it can generate native code for iOS (Apple) and Android devices. Some of the model-driven execution platform low-code vendors offer a downloadable helper application that acts as a device-resident execution platform for the application. It's possible to put this helper app in an app store, either as a separate download or bundled with the application itself.

In any case, making this tradeoff has its advantages. Model-driven execution platforms provide security and performance benefits because the vendor takes care of the underlying technology that runs in the cloud. These vendors also keep their tools updated, where updates automatically apply to all applications running on the platform.

## Taking Off from Here

If you've read this chapter or this book up to this point, you may be asking, "What's next?" or "Where do I go from here?" The best place to start, of course, is at the beginning, which involves downloading a low-code tool and just starting to build.

There are plenty of vendors and tools out there for you to choose from. Some are easier to use than others, and some are more expensive than others. Regardless, just start building.

VeriTran, the sponsor of this book, would naturally be ecstatic if you downloaded its platform. As one option, check out `www.veritran.com/know_our_platform` for more information.

IN THIS CHAPTER

- Needing faster, better apps
- Reaching your business goals with external and internal apps
- Getting value from low-code
- Reducing backlog with low-code
- Understanding the role low-code plays in automation
- Extending existing applications with low-code

# Chapter 2
# Understanding the Business Drivers for Low-Code

Today's professional software developers have many tools at their disposal that increase their productivity and efficiency while empowering them and their teams to generate better applications faster. Better apps faster, however, is just the starting point for low-code. In reality, these tools offer a number of advantages to organizations who are looking for a range of different value propositions from their software. In this chapter, I give you the main businesses drivers behind low-code.

## Building Better Apps Faster

To understand the business benefits of low-code, the starting point is all about speed and quality. Creating apps using a simple visual environment has the clear and obvious advantages of speed

as well as quality. Basically, far fewer things can go wrong when building an app with low-code as compared to hand-coding. I cover hand-coding more in Chapter 1.

There's no guarantee that you'll build a high-quality app simply because you're using low-code, of course. Where low-code really helps is in avoiding mistakes that can lead to errors, or worse, security vulnerabilities.

Given the power of low-code tools, building high-quality apps is easier than ever, even when starting from scratch, as opposed to using a template or building on one or more existing applications. Such custom-made apps do require a level of sophistication that low-code (as opposed to no-code) tools offer.

Figure 2-1 illustrates the process of building an app from scratch.

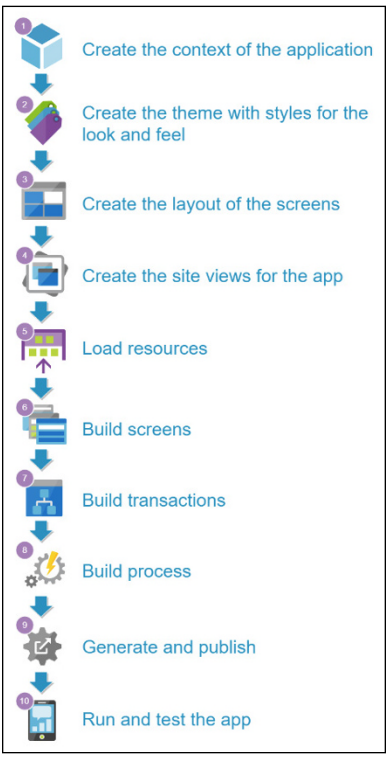


**FIGURE 2-1:** Building an application from scratch.

# Achieving Business Goals with Internal and External Apps

Is low-code right for you? The answer may depend on the type of application your organization wants to build. Many applications are for the internal use of employees or other people within your organization. Internal, business-to-employee (B2E) apps are generally tactical. A tactical application focuses on delivering short-term value within the context of the day-to-day operations of your business.

Here are a few app examples:

- Human Resources (HR) apps, such as timesheet apps, time-off request apps, and more
- IT-related applications, such as password reset apps or equipment service apps
- Data processing apps that present a form for people to fill in

  Such apps may be for anything from registering for an event to signing up for a special duty.
- A reporting or dashboard app that collates and presents useful information to managers and other personnel

## ALIGNING IT WITH BUSINESS GOALS

Banorte is a Mexican banking and financial services holding company with headquarters in Monterrey and Mexico City. It is one of the four largest commercial banks of Mexico by assets and loans and the largest retirement fund administrator. As a long-time customer of VeriTran, Banorte uses low-code to meet business objectives. In fact, José Francisco Martha González, Payments, Digital Banking, and IT Managing Director at Banorte, said, "An advantage of low-code has been aligning IT with the business objectives; it has helped us to get our teams closer and getting the experience we want (customer-business-IT) as fast as our consumers and market need it."

Apps that you want to put in the hand of your customers are externally-facing apps. These apps are generally more strategic because they help your organization achieve its business goals. These goals are likely making money, expanding your market, and keeping customers happy and coming back for more.

Here are some examples of external strategic apps:

- An ecommerce app that allows customers to view a selection of products, select one, and then purchase it online
- Customer service apps that enable customers to interact with contact center reps via a web interface
- A warranty processing app that enables customers to go through the process of returning an item for refund or repair
- A survey or other feedback app that solicits the opinions of customers or prospects

Apps that increase market share, for example, are likely to be strategic. If an application both drives revenue and also provides a competitive differentiator to the business, it's also likely to be strategic to the organization.


TIP

When apps are expensive and time-consuming to build, then the focus should be on building the most strategic ones. For many organizations that still hand-code apps, tactical apps often languish, as strategic apps are a higher priority.

As the app development team gets up to speed with low-code, it's now cost-effective to build a wide range of different types of apps, from the more strategic, mission-critical ones to the important, but more tactical ones.

## Streamlining the Work of Professional Developers

Given the wider range of people in the organization responsible for building apps, combined with the greater focus on the customer experience, it might be tempting to think that the role of the professional software developer has become less important in the era of low-code. However, the opposite is true: Professional developers become even more essential because their role centers on providing strategic value.

Historically, creating software consisted of understanding and documenting the business logic, combined with writing code. Low-code platforms have mostly automated the writing code part of this equation. The result? The role of the professional developer has shifted to a focus on the business logic — that part of software that provides value to the business while also frequently requiring the expertise of a professional developer.

In many cases, this business logic centers on how the organization executes its business processes. In other cases, such logic focuses on processing data or providing some kind of interactivity to customers or other people.

REMEMBER

Regardless of the purpose of the business logic, low-code empowers developers — both professional and citizen — to spend more time and effort focused on it, and less on what we may call the plumbing of a modern application. For the hand-coder, such plumbing consumes much of their time. These grunt-work activities include writing security code, integrating with third-party applications, and incorporating various software libraries that may or may not be a good fit for the application the hand-coder is writing.

## THE POWER OF LOW-CODE

Héctor Roldán, former CIO of Banco Bci and BancoEstado, and former Corporate Projects Director at Santander, and his team worked with VeriTran leveraging the VeriTran Low-Code Platform to launch the first mobile app of one of the largest banks in Latin America and the only public bank in Chile. The results were impressive, in terms of both speed of development and business results. With more than 3.5 million active customers and more than 9 million transactions processed daily, this project has been internationally recognized and awarded as a financial inclusion and digital transformation success case.

According to Roldán, "Low-code platforms will radically transform corporations' development processes, enabling them to transform their business models faster, more efficiently, and more securely."

Low-code largely or entirely takes all this plumbing off the hand-coders' plates, enabling them to focus on the more valuable business logic that in the end is also more fun to create.

## Clearing Out the Application Backlog

Given all the digital transformation going on in companies large and small, it's clear that software is becoming increasingly strategic and pervasive — which means they need more of it. A lot more.

The demand for new and updated applications and software infrastructure, as well as individual features and capabilities, is exploding. And yet, every such company is resource constrained, as professionals who are able to build and run such software are in short supply.

In the meantime, however, such enterprises have little choice but to add to their extensive software to-do lists. These backlogs, as the industry calls them, can get long and unmanageable — and business pressures combined with the ongoing shortage of skills only make this problem worse.

REMEMBER

Because low-code platforms focus on giving professional developers the ability to build applications more quickly while minimizing the need to hand-code, traditional development backlogs are better able to focus on strategic applications as well as tactical ones.

As a result, low-code improves both the productivity and morale of the developers as well as the customer-centricity of the apps themselves, while simultaneously easing the ability to add such features and tasks after apps go live. With improved productivity comes faster completion of applications in the backlog, and many firms are finding that low-code is essential to reducing the size of the backlog or eliminating it altogether.

## Automating Workflows

Throughout history, IT has always been forced to prioritize how it spends its money (in the biz, they call that *resource constrained*). After all, keeping the lights on has always taken so much of every dollar.

No more. As IT shifts from its historical cost-management role to its bet-the-company strategic position within digitally transforming enterprises, companies simply can't afford to prioritize some critical customer-facing efforts over others. And yet, managing costs will always be important. Don't fool yourself that it will ever be otherwise.

REMEMBER

The secret to balancing these strategic priorities is *automation*. Automation empowers people to be more productive, as the technology removes less valuable tasks from their plates. In fact, the strategic priorities of digital transformation make automation a must-have – and the companies that automate more effectively are the ones that will succeed in the long run.

It's no wonder, therefore, that most of today's low-code tools and platforms help organizations deliver some kind of automation. From traditional business processes to complex big data workflows, most enterprise apps take some sequence of tasks — some manual, some already automated — and join them into more complex applications.

At this level, a number of forces are in play. An enterprise low-code platform can provide the overall framework for generating such complex logic, typically without requiring hand-coding. The architectural effort will then support such logic while dealing with non-functional requirements like scalability and resilience.

Just how much of this enterprise context for such applications a low-code platform can deliver depends on the particular platform in place. Some low-code vendors focus more on such enterprise scenarios than others — a classic example of the right tool for the job.

## Extending and Updating Existing Applications

The reason people use low-code is to build applications — but not always *new* ones. Sometimes you have an existing application that's partly serviceable but needs some freshening up. The good news is that in many cases low-code can help. Because low-code tools can be great for building slick new user interfaces, they can often replace an out-of-date interface with a new one, leaving the existing business logic in place.

TIP

For example, many organizations use low-code to build mobile front-ends for existing applications. Older data entry apps, in particular, are rarely mobile-friendly. Creating a new app with low-code can make the data entry simpler and friendlier on various devices, and in some cases, can also access special functions of the device itself.

The typical example of this kind of mobile extension is an insurance claims app. In the old days, the adjuster took notes and photographs manually and then went back to the office and laboriously entered and scanned the information into the application. Today, the claims app runs on a phone or tablet and accesses the device's camera, automatically entering that fender bender info directly into the underlying corporate claims processing application.

## BEWARE LEGACY INTEGRATION GOTCHAS

TECHNICAL STUFF

Typically, connecting the new app or component to the old leverages the older one's application programming interfaces (APIs). Today's low-code tools come with a number of connectors to common enterprise apps, both on-premises and in the cloud, that leverage those APIs.

In other less common situations, such a connector may not be available, often because you're trying to connect the new app to an older application that your organization custom built perhaps years in the past. In this situation, a professional developer typically has to code the connection to the API manually — one of the most common uses of hand-coding with low-code tools.

Of course, some particularly ancient legacy apps don't have APIs at all. In other cases, the business logic itself is in need of updating, perhaps in conjunction with the interface. When this eventuality happens, it sometimes makes sense to build a modular addition to the older app that essentially runs as a new application, only it replaces some of the functionality of the older one.

One familiar example: Many organizations have ancient HR systems that handle all elements of the hiring process. But your organization may want to have a new, social media-savvy recruitment tool that replaces the clunky recruitment module of the older HR app. Low-code can be the perfect way to accomplish this feat.

IN THIS CHAPTER

- Deciding between low-code and no-code
- Understanding who should build which applications with low-code
- Dealing with political issues like pushback
- Working with the right tools
- Knowing where to start

# Chapter 3
# Weighing Your Options before Building Applications

While professional software developers are often the target users for many low-code platforms and tools, other people in the organization may also be able to build applications with low-code. However, not everybody is comfortable with this responsibility, no matter how simple the tool is to use.

Who in your organization should use low-code, in fact, often depends on the sorts of applications your organization needs. Just as individuals specialize, so too do low-code platforms — and such specializations often align with each other.

There is more to consider, however, than matching up the right people to the right tool. In many cases, political and cultural considerations also impact an organization's ability to get the most out of their low-code investment. In this chapter, you gain some of the context you need to be successful with your low-code efforts.

# Choosing Low-Code or No-Code


TIP

For any organization looking to accelerate and simplify how it builds applications, the first decision is whether to use a low-code or no-code platform. I cover low-code versus no-code in more detail in Chapter 1, so flip back there if you want to know more about the differences. Here, I want you to take a hard look at the software projects on your agendas and decide whether those projects require the engineering skills of professional developers.

If the complexity and sophistication of the required applications require a level of coding beyond what a low-code tool can offer (suggesting that professional developers are best suited to the task), then a low-code platform is the right choice.

No-code tools, in contrast, are the best fit when the business division in question has a reason not to involve professional developers. Perhaps those developers are too busy because resource constraints limit their abilities to meet every need of the business. In other situations, the apps in question require less technical expertise, instead leveraging the greater subject matter expertise that the business people are more likely to have.


REMEMBER

These considerations are merely broad generalizations. There are certainly plenty of professional developers who find that no-code tools are up to the tasks set out for them. Similarly, many of today's low-code tools are simple enough for citizen developers to use for most tasks — and they can always call in the pros for those situations that require them.

# Looking at Different Application Builders

Just calling someone a professional or citizen developer doesn't tell the whole story. In reality, there are several types of each.

## Meeting the pros

For the purposes of this discussion, I break down the pros into three camps:

- **Front-end developers** who usually work on user interfaces (including web pages) and are typically adept at languages like HTML, CSS, and JavaScript

In some cases, a front-end developer is more of a designer than a programmer, while others have serious programming skills. In either case, these front-end folks are well-suited for both low-code and no-code tools because they have the expertise to design a well-thought-out user interface. Of course, a designer with few technical skills may be perfectly comfortable with such tools as well — especially no-code.

- **Back-end developers** who work on integration, databases, and other parts of the IT infrastructure

  An organization is less likely to call on the back-end developer to use a low-code tool to build applications themselves, but these individuals can still be an essential part of the low-code team.

  In many cases, integrating with other apps is core to the application, as are security and regulatory compliance considerations. The back-end developer can handle these tasks, as can the full stack developer. In other situations, a significant part of the application itself is on the back-end (in other words, on the server), for example, when an application supports multiple people working on mobile devices.

- **Full stack developers** who work on tasks that require their unique combination of skills

  These people are rare, and your organization may be lucky to have one or two. The low-code team is unlikely to get all that much help from this area, because it will be too busy with other tasks.

## Meeting the citizens

Citizen application builders are typically analysts of one type or another (although their titles can vary widely). There are also several different types of citizen developers:

- **Process analysts:** The process analysts understand the business processes in one or more divisions or functional areas, and are best suited for building process-centric applications.
- **Data analysts:** Other citizen developers are data analysts and are comfortable working with a low-code tool for building data-centric applications such as business dashboards or analytics programs.

- **Business analysts:** By far the greatest number of citizen developers are business analysts, who have a broad understanding of what the business is trying to accomplish, and can help build applications accordingly.

REMEMBER

For all of these different types of citizen developers, one question always comes to the surface: Just how technical does a citizen developer have to be to become a productive member of a low-code application construction team? My answer is . . . as technical as an Excel power user. Just as most business people have some expertise in Excel while a few have deeper knowledge of the tool, so too with citizen developers. Anyone can drag and drop visual elements onto a canvas — but it takes people who know their way around Excel formulas and the spreadsheet's other more advanced functionality to get the most out of either a low-code or no-code tool.

## Being Successful with Low-Code Regardless of Your Role

You may think of application development as having an easy end and a hard end — the easy end being the front-end, user-facing part where the visible portion of business applications reside, and the hard end being the back-end, where modernization and integration challenges tend to complicate application deployment and slow everything down. Over the years, low-code tools have focused primarily on the easy front-end. Building user interfaces following drag-and-drop principles is relatively straightforward, so many vendors have tossed their hats into the low-code ring, offering various user interface builders to citizen and professional developers alike.

REMEMBER

But the modern context for application development goes well beyond the front-end. True, digital efforts require a solid user experience, but customers always require high-performance applications that connect them to enterprise functionality. For this reason, the application back-end is every bit as important as the user interface.

The keys to low-code success are a combination of multiple factors. Organizations need effective cooperation among lines of business, the application development team, and the rest of the

IT organization. Companies must place low-code efforts into the context of existing applications, both to integrate with them and, when appropriate, make modernizing them less risky.

## Dealing with Pushback

Given all the advantages of low-code for so many people in different roles in the organization, you might think that the path to adopting low-code is a bed of roses. Unfortunately, a variety of political and cultural roadblocks impede that path toward petal-covered bliss.

You can't really blame the individuals involved. Everybody acts out of self-interest, after all. In the corporate environment, self-interest is supposed to align with the priorities of the organization, but many times it doesn't.

WARNING

In practice, people are unlikely to support a decision that puts their jobs, their standing among their peers, or their likelihood of advancement at risk, even when such a decision would be in the best interests of their company. In many IT shops, this cover your assets (CYA) way of thinking impedes the decision to move to low-code. Here are a few examples:

- Developers may question whether low-code will put them out of a job.
- Ops personnel may wonder if low-code is going to be difficult to manage in production.
- Security and compliance staff often have concerns about how low-code will impact their areas.

*Asking* such questions is rational, and each of these professionals would be that much less professional for not doing so. However, jumping to the conclusion that low-code will adversely impact the individual is rarely based in fact.

TIP

In reality, train yourself in new technologies that may brighten your future career options because they help people focus on building applications that provide business value. Low-code can refresh a developer's focus on the customer because it reduces the need to focus on the plumbing aspects of the job.

# Making the Right Choice of Tooling

Every bit as important as the consideration of the type of low-code platform are the characteristics of the applications an organization is likely to build. Many applications are in fact workflows or automated processes. For these applications, a subcategory of the low-code market, *Digital Process Automation* (DPA), is likely the preferred choice. DPA tools are low-code tools that focus on building process-centric applications.

However, there are other types of applications, including mobile apps, form-based apps, and data analysis apps. While a few low-code platforms specialize in all of these types, most of them focus on one or another.

Then there's the question of how much hand-coding a low-code application requires. Sometimes, the role hand-coding plays in the context of a low-code platform is essential to making the right decision about which platform to use. For example, some organizations require custom user interface controls or widgets that they must hand-code. See Chapter 1 for more on hand-coding.

In other situations, some of the business logic in the application is complex and custom enough that hand-coding is the simplest approach to implementing it. In such situations, it's important for the low-code platform to support hand-coding.


Many low-code platforms support JavaScript coding for this purpose, but other languages are supported on occasion. Sometimes, Excel-like formulas are sufficient to meet the business need. Support for such formulas is common across many DPA, low-code, and no-code products on the market today.

Other low-code platforms support some sort of rules engine, allowing natural language 'if this, then do that' type statements. Such rules aren't appropriate for many situations, but in other cases, the rules approach is sufficient for defining the business logic of an application, without the need for any hand-coding.

Every DPA, low-code, and no-code platform follows some kind of visual metaphor for building applications. Some are drag-and-drop, while others favor wizards. Some products even

follow a spreadsheet metaphor for building complex, multi-user applications.

In Figure 3-1, you see an illustration of how to build a basic workflow by using a low-code tool. DPA tools that focus specifically on workflows and processes typically have a more sophisticated workflow building interface, but Figure 3-1 shows a typical one for a low-code tool.

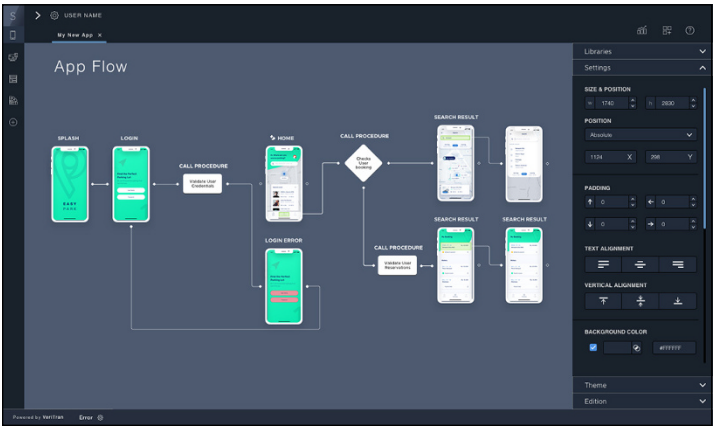


**FIGURE 3-1:** Building a simple workflow.

Finally, some low-code products have low (or no)-code front-ends with back-ends that require hand-coding, or vice versa. For an organization who wants citizen developers to hammer out the user-facing app but have professional developers handle the server-side logic, no-code on the front-end can make sense. In other situations, the back-end server-side application behavior is all low-code, but the platform enables front-end developers to hand-code user interfaces.

REMEMBER

Instead of the misalignment between business and IT that has been the bane of every enterprise for so many years, this new paradigm also requires *collaboration* — collaboration within IT to be sure, but more importantly, across the organization, spanning IT and line-of-business functions. Without collaboration, business agility is difficult to achieve.

# Knowing When You Really Need to Write Code

A low-code tool's application construction interface can be both simple and powerful, but sometimes you just need to write some code by hand. Even the most mature low-code tools require some hand-coding now and then. When to hand-code, and how the tool supports that hand-coding, varies from tool to tool, and depends on why you need to hand-code. Here are a few cases in which you'd want to write your own code:

- **You require some kind of custom behavior on the user interface.** Low-code tools generally come with a large number of widgets, such as pull-down menus, calendars, checkboxes, and the like, but sometimes you want to create your own. Hand-coding is probably the best way to build such widgets.
- **You want to integrate with other applications.** Most low-code platforms come with pre-built connectors that you can drag and drop into position with only a bit of mapping from one data field to another. Or perhaps the app you want to connect to doesn't have any connectors, either because it's old, obscure, or because your organization created it in-house. In such situations, hand-coding may be the only way to connect up the app.
- **Rules or formulas are overly complex.** If you've ever created a formula in Microsoft Excel, you'll recognize this situation. Most formulas are simple and straightforward, but sometimes they can be quite complex. So too with low-code tools. In fact, many of them support formulas that look a lot like Excel's. When those formulas get too messy, creating them amounts to hand-coding.

REMEMBER

In many cases, low-code tools support hand-coding directly in the application builder interface. Simply double-click on an object, and there may be a window you can type some JavaScript (or other language) into. In other cases, vendors give you a fully-featured software development kit (SDK). SDKs are essentially toolkits for professional developers that give them everything they need to extend the applications they can build in the low-code tool via hand-coding.

IN THIS CHAPTER

- Learning enterprise low-code
- Understanding the modern enterprise application
- Dealing with enterprise-centric issues
- Placing low-code into the context of legacy modernization
- Understanding the role low-code plays in hybrid IT, the cloud, and DevOps

# Chapter 4
# Getting Enterprisey

Today, modern enterprise low-code platforms can tackle the toughest of modern enterprise applications — from complex, multicloud workflows to elaborate application modernization initiatives. Enterprise low-code must also become an integral part of any enterprise DevOps effort — both as a set of automation tools for accelerating development, integration, and deployment, and as a framework for greater collaboration as enterprises leverage modern applications as they compete in the digital era.

In this chapter, you gain an appreciation for the challenges and complexities of building applications with low-code in large enterprises.

## Comparing Lightweight Low-Code to Enterprise Low-Code

Within the low-code space, the market is broadly dividing into two camps: low-code and *enterprise* low-code.

REMEMBER

The notion of an enterprise-class low-code platform contrasts with what you may call the lightweight alternative. After all, one of the primary benefits of no-code platforms is that they're lightweight, easy to deploy and use, and have a shallow learning curve. Citizen developers can get up to speed in a matter of hours. Many of these platforms even allow such application builders to implement simple integrations using drag-and-drop tools.

Many low-code platforms are also lightweight, although not necessarily as simple to use as their no-code counterparts. The products without the enterprise label focus on supporting the construction of apps for midsize firms and departmental needs in large organizations.

But what all these lightweight offerings generally lack are enterprise-class features such as advanced integration, regulatory compliance, and enterprise-centric security controls. Enterprise low-code must focus on the software needs of large organizations beyond simple departmental applications.

REMEMBER

True enterprise low-code players must also support the broader architectural context of complex, enterprise applications that may span multiple teams working on multiple parts of the application at once.

On the one hand, these enterprise-class capabilities require more technical expertise than citizen developers are likely to possess (although there are exceptions, of course). But on the other hand, such capabilities also require active collaboration with the IT organization.

TIP

Even the simplest no-code platform requires some support from IT. But in the case of enterprise-class features, the IT organization has an active, ongoing role in supporting the low-code efforts — even when building applications requires no coding at all.

## Understanding the Modern Enterprise Application

One of the most important uses of enterprise low-code is to build complex, modern enterprise applications. What, then, do these applications look like? Do they look like the massive, monolithic applications of old?

Not on your life. Today's enterprise applications are fully modular and hybrid, running in cloud and on-premises environments as appropriate for the tasks at hand. Modularity — a core software best practice since the 1990s — now means building containerized applications with microservices following cloud-native best practices.

Even hand-coders realize that in this modern application landscape, it's impractical to build much of anything from scratch. Every modern application consists of significant quantities of repurposed code — typically open source modules and frameworks that make up much of the code of any app today.

This modular, reuse-centric context for modern application creation increases the practicality and usability of enterprise low-code platforms relative to hand-coding. In fact, the whole drag-and-drop metaphor at the heart of how most low-code tools work depends on this modularity. Without such platforms, much of the developers' time consists of writing various bits and pieces of code that connect, orchestrate, and scale existing components and applications — a process also called *plumbing.*

Low-code takes this plumbing work off the developers' plates, allowing them to spend more time engineering solutions that meet business needs quickly, securely, and with ongoing input from stakeholders as appropriate for the application in question.

## Dealing with Security and Regulatory Compliance

Both security and regulatory compliance differ from other enterprise requirements by virtue of the fact that they're both focused on risk mitigation. As a result, enterprises would simply rather not spend any money at all on them. They only do so because they have to.

While regulations set a clear bar that every organization must aspire to, security has no such threshold. In fact, it's impossible to drive security risk to zero — which is simply another way of saying that perfect security would be infinitely expensive.

WARNING

Both regulatory compliance as well as security are comprehensive business concerns that extend well past IT — and therefore, well past application development. Any organization that separates application development security and compliance challenges from this broader business context opens itself up to additional risk, rather than mitigating it.

Any low-code initiative must participate within this business context. As security and compliance priorities trickle down from the C-suite to the IT organization, IT must institute and reinforce cybersecurity (security within IT) and compliance procedures and constraints for low-code as well as the rest of the application development effort.

TIP

The challenge that the IT organization faces is how to ensure it meets these requirements without sacrificing the speed and agility benefits that low-code brings to the organization. If you're considering low-code platforms for enterprise use, select one that supports the IT organization's mandate to provide secure and compliant software as an integral part of the function of the low-code platform.

Many of the enterprise low-code platforms on the market are up to speed on such security and compliance requirements. In fact, building secure, compliant applications is actually *easier* on such a platform as compared to hand-coding because the low-code platform essentially enforces risk mitigation best practices.

For example, a common hand-coding mistake allows hackers to enter malicious text in form fields, which the application will execute without anyone knowing. Hand-coders must then apply the best practices for blocking such inputs, and implement those practices by hand. In contrast, the low-code platform automatically removes malicious data from such fields so the application builder doesn't have to worry about taking care of such issues manually. For less experienced developers (including citizen developers), the ability to remove dangerous data is absolutely essential. Even for more experienced, professional developers, automatic security and compliance controls reduce headaches as well as the possibility of a mistakenly introduced vulnerability.

REMEMBER

Any enterprise-grade platform must support the organization's security context following low-code principles. For example, it should be straightforward for developers to integrate an app with its existing identity management technology and then to configure user roles and permissions for that app via simple, low-code principles.

## Using Low-Code to Integrate with Existing Apps

In the enterprise context, no application is an island unto itself. While early low-code apps tend to be stand-alone, it doesn't take long until business units want to integrate their apps with other applications in the enterprise and in the cloud. Without the proper low-code integration capabilities built in, such integration presents a difficult set of challenges, especially for more complicated requirements.

TECHNICAL STUFF

Today's low-code platforms all generally support application programming interfaces (APIs), both creating them as part of the low-code application building process, as well as consuming them from third-party endpoints, including on-premises applications, databases, and SaaS apps such as Salesforce and ServiceNow.

As low-code deployments become more complex, organizations may focus on building automated, multi-step data orchestrations. They soon find that there is more to the integration challenge than simply connecting APIs.

In such situations, the application back-end itself requires some kind of hands-on development, either to expose it with APIs, or in the more general case, to modernize it to conform to broader architectural requirements like the horizontal scalability and elasticity we expect from the cloud.

The low-code platform's focus on separate applications then falls short of the needs of the organization. It may be straightforward to build the front-end of a multi-tier application using low-code, but shifting the back-end, say, to a modern microservices environment would still fall to hand-coding.

TECHNICAL STUFF

Delivering back-end capabilities as modular microservices is typically part of this evolution of low-code platforms, especially when vendors expose microservices as APIs and include them in their platforms' drag-and-drop application assembly environment.

Depending on the underlying, existing application, some measure of hand-coding is often inevitable. Low-code platforms must provide sufficiently powerful coding capabilities to handle such outliers, and once developers have created such integrations, the platform should support the ability to place them in a catalog for future use and configuration.

## Facing Modernization Strategies with Low-Code

Modern digital priorities are increasing the importance of legacy modernization by placing a greater negative value on the pain and expense that legacy assets are causing the organization. Gone are the days where enterprises can put up with such painful legacy. Today's IT requires an appropriate modernization strategy that takes into account this economic argument, while also better understanding the appropriate choices for any particular legacy asset.

WARNING

What about those enterprises who divert their attention and resources elsewhere, choosing instead to leave their legacy alone? Many will find that such a strategic error will make them less competitive, and some will even find themselves in the impossible situation where it's simply too late for the organization to recover.

Furthermore, most if not all modern enterprise applications either leverage or replace existing software, or both. In fact, this challenge isn't simply about integration — although integration is difficult enough as it is — it's also about *modernization*.

The larger the enterprise, the more likely it is to be burdened with a diverse portfolio of legacy applications. Furthermore, modernizing such legacy systems while simultaneously replacing the remainder of aging application portfolios with new, flexible applications has long been a priority for such organizations — but a goal that has largely been out of reach.

Low-code platforms can make this goal a reality. Low-code development teams modernize a monolithic legacy application by targeting specific functionality changes that end-customers require and replacing that functionality with modular software.

The modern approach to modular software consists of microservices. It's relatively straightforward to implement microservices in a low-code manner with modern enterprise low-code platforms even in conjunction with existing legacy assets.

Enterprise low-code platforms that enable organizations to tackle complex, enterprise apps within the context of the modern enterprise IT landscape are no longer simply a wish-list item. Most enterprises are actually achieving some measure of success with such platforms today.

This story repeats itself across multiple verticals, from banking and insurance to manufacturing to retail and ecommerce. An increasingly wide swath of the enterprise legacy application landscape is now a target for low-code initiatives, in addition to the custom-built digital efforts that have long been the sweet spot of such platforms.

## Dealing with Shadow IT

REMEMBER

Shadow IT refers to how business divisions often bypass the IT organization in order to get the software functionality they require, typically by using simple application creation tools that don't require the help of a technical person. Such apps end up running off the radar of IT, leading to all sorts of problems.

The causes of shadow IT are well-known: The IT organization is too slow and process-bound to respond quickly enough to the needs of lines of business, so business users go around IT and buy — or build — their own apps and other technology.

WARNING

The problems of shadow IT, unfortunately, are also quite familiar: Without proper oversight and coordination, security vulnerabilities and compliance violations can proliferate, with no one at the helm to address such issues. Furthermore, these citizen-built apps can be redundant, obsolete, or otherwise low quality.

For modern no-code platforms, this shadow IT pitfall looms large. Certainly, some of the more mature no-code platforms tackle the shadow IT issue head on, providing lightweight ways of ensuring that apps on these platforms are adequately secure, compliant, and address ongoing business needs. Far more common, however, are less mature no-code tools that don't adequately deal with these pitfalls. These are common because no-code is a rapidly emerging market, and numerous startups are frantically rolling out their wares, putting shadow IT considerations on the back burner.

Traditionally, avoiding shadow IT depends on the establishment and enforcement of formal IT processes. However, in many cases, these processes date from the last century and are agonizingly slow and bureaucracy-laden.

From the perspective of line-of-business executives, many of the applications they require don't lend themselves to such a time-consuming process. Perhaps the need is too urgent, or in other cases, the underlying security or integration complexity doesn't warrant following traditional application development procedures. Such business/IT misalignment has typically led to shadow IT.


TIP

Enterprise low-code platforms can resolve this dilemma, giving lines of business an increased ability to call for new applications without requiring IT to follow every step of now-obsolete software life cycle processes. In addition, these platforms support the reuse of components that IT has built previously, while leaving those components in IT's control.

This approach, however, isn't the shadow IT of old. The "enterprise" part of the enterprise low-code story focuses on security, compliance, and integration — those areas where IT must still step up to the plate to ensure that such core non-functional requirements are still being met.

The end result is a new way of organizing the relationship between lines of business and IT, where IT empowers the business to take greater control over increasingly flexible application delivery, while still maintaining the necessary focus on security, compliance, and integration that everyone agrees are every bit as important as they always have been.

## Putting Low-Code, the Cloud, and Hybrid IT into Context

There's an old joke about legacy technology: Legacy technology is anything that works. The underlying truth to the joke, of course, is that existing technology is only in operation because it still meets a need, regardless of its age. In fact, "legacy" doesn't even necessarily mean old, and there's more to understanding the legacy modernization challenge than simply whether some existing piece of technology is still working.

REMEMBER

In reality, legacy refers more to the amount of technical debt a particular piece of technology has; by *technical debt,* I mean how expensive and difficult it would be to resolve any of the issues the subject technology suffers from that keep it from meeting current needs. The legacy modernization challenge has always boiled down to an economic argument of how much pain does the cost of maintaining an out-of-date system cause the organization versus the all-in cost of modernizing that legacy. This includes all the indirect costs of making the transition from old to new, including downtime, retraining, customer resistance to change, and so on.

At least, that was the modernization challenge enterprises faced until a few years ago. Today, *hybrid IT* has thrown a monkey wrench into these traditional economic arguments, largely because of the rise of cloud computing.

REMEMBER

Hybrid IT is a workload-centric management approach that seeks to abstract the choice of deployment environment across multiple public clouds, private clouds, and on-premises and cloud-based virtualized environments, as well as traditional on-premises systems, including legacy assets. Hybrid IT represents the overarching paradigm for modern IT operations. However, in spite of the inclusion of legacy under hybrid IT's umbrella, I don't mean for hybrid IT to *perpetuate* legacy. Rather, hybrid IT gives enterprises the means to *modernize* legacy.

Given this range of choices, modernizing a monolithic legacy app is no longer a monolithic task in its own right. However, such modernization typically requires the creation of new application capabilities, which brings up the essential role enterprise low-code platforms play within this modern perspective on legacy modernization.

One of the reasons why IT managers in the past have recoiled from modernization tasks is because of the cost, time, and risk inherent in hand-coding replacement functionality. Low-code changes this equation, lowering both the time and risk of application creation.

WARNING

Migrating applications to the cloud is one part of this move to hybrid IT. However, rarely is it possible to take a legacy application, stick it in the cloud, and get all the benefits of the cloud as a result. Instead, it is typically necessary to modernize such an application, if only to enable it to leverage the scalability benefits of the cloud. In many cases, low-code can reduce the time and effort necessary to accomplish such modernization.

As enterprises get up to speed with hybrid IT, being able to run applications in various clouds as well as on-premises environments becomes a priority. Once again, low-code can facilitate and streamline the work necessary to make applications sufficiently flexible.

Modernization efforts are perhaps the most advanced use of low-code that any organization is undertaking today. It requires the expertise of professional developers, architects, and other IT personnel who are particularly adept at dealing with hybrid IT scenarios. Even among such professionals, however, enterprise low-code platforms are becoming one of the important tools for achieving the ongoing value of hybrid IT.

## Lending a Hand to DevOps

The broader context of enterprise IT has been shifting to an entirely new hybrid IT model, centered on the cloud while incorporating on-premises assets, and so too has the organizational context for software in the enterprise. This organizational trend is *DevOps*.

REMEMBER

DevOps is an automation-driven model for collaboration across the IT organization, including development, quality assurance, operations, and security — as well as an increasing level of collaboration with people in customer-facing roles that represent the business.

Enterprise low-code, coupled with architecture and DevOps best practices, can finally empower even the most conservative of companies to better meet the ever-changing needs and desires of their customers — as long as such organizations can scale up their DevOps efforts without slowing them down.

Low-code can help with this scaling challenge. With a low-code platform, IT organizations can establish and enforce an application delivery *cadence* — a rhythm of application releases that maintains consistency across the IT environment.

The result is a modern update to traditional IT portfolio management, where a governance team can manage a large number of applications across many parts of the business, efficiently allocating resources while managing risk. However, unlike traditional approaches to portfolio management that slow everything down with paperwork-laden bureaucratic processes, the modern approach is lightweight, iterative, and customer-focused. Low-code platforms are essential to achieving these goals of such modern software-driven environments.

IN THIS CHAPTER

- Understanding VeriTran's low-code platform and tool
- Learning about VeriTran's differentiators in the low-code marketplace

# Chapter 5
# Ten Ways VeriTran Can Help You Be Successful with Low-Code

VeriTran is a low-code vendor that offers a model-driven execution platform, as well as a low-code application creation tool called VeriTran Studio. It offers different subscription levels as well as managed solutions for organizations that need help building their applications.

In this chapter, I introduce you to VeriTran and how it can help you on your path to success with low-code.

## Visual Development

VeriTran Studio takes a drag-and-drop, what-you-see-is-what-you-get approach to building app interfaces. This visual building tool enables application builders (either professional or citizen developers) to create, integrate, and extend their apps without the need for hand-coding.

## FASTER TIME TO MARKET

A good example of VeriTran's fast delivery is its work with Banorte. VeriTran helps Banorte in the development of its banking and payments mobile app. In fact, the first phase of work on the project only took eight weeks to complete. "VeriTran allowed us to reduce our time to market, making digital bank services available to our consumers in a monthly delivery plan. Working with the VeriTran Low-Code Platform has helped us to deal with the fast and changing digital environment we live in," states José Francisco Martha González, Payments, Digital Banking and IT Managing Director, Banorte.

# Faster Delivery

VeriTran accelerates application delivery via templates that are essentially pre-built apps for application builders to use as-is or tweak to meet their needs. VeriTran also offers pre-built functional components such as push notifications or biometrics that ease the assembly of sophisticated apps without coding. VeriTran also supports the testing of its apps in real devices with real end-users.

# Business Rules

VeriTran enables application creators to set up business rules, either within the front-end (user interface) or back-end (server functionality). These rules can represent business logic that drives application behavior. Alternatively, application creators can use rules to profile and segment users demographically in order to provide different groups of users with different functionality, either for testing or personalization purposes.

## Collaborative Teamwork

In some cases, a solitary individual builds an application, but in large organizations in particular, application creation is typically a team sport. VeriTran supports multiple people working on the same project at the same time. Sometimes such team members have particular roles and specialties, such as mobile interface design, process logic, or data entry forms. In other cases, everyone pitches in on whatever task needs to be done. VeriTran supports both modalities.

## Omnichannel Deployment

Omnichannel recognizes that from the customer's perspective, various marketing and communications channels (online, mobile, email, in-store, kiosk, call center, and so on) represent a single, coordinated interaction channel. With VeriTran, application builders can build an app once, and the VeriTran platform automatically displays it across multiple devices and form factors with a single coordinated user experience. The builders are able to reuse transactions, components, and other elements across channels seamlessly.

## Dynamic and Multilevel Security Models

VeriTran's authentication and authorization approach goes well beyond basic role-based access control in several ways. The platform offers multifactor authentication with built-in biometrics support. In addition, VeriTran requires security credentials according to risk level, where an app can require more stringent authentication if a user attempted to access a more secure area. VeriTran can even issue out of band authentication and sign transactions for additional security.

## LOW-CODE FOR SECURITY

BanBajío is a Mexican banking and financial services institution based in the city of León. BanBajío offers all the products and services of multiple banking with integral banking solutions for individuals and corporations. BanBajío has more than 800 thousand clients, 4,500 employees, and a multi-regional presence with 306 branches. The company is the eighth bank in Mexico and a leading bank in credit products dedicated to the development of the Small and Medium Business sector.

Roberto Hernández de Hita, Chief Transformation Officer (CTO) at BanBajío, one of VeriTran's oldest clients, states, "One of the main benefits of VeriTran Low-Code Platform has to do with its robust security. Our mobile app, developed with its technology in just eight weeks, is a channel in which we have had zero fraud incidents in the nine years we have been working together."

# Multiple Industries

VeriTran focuses on healthcare, banking and payments, insurance, and the public sector in particular, but its platform is suitable in most applications in any industry. It is especially adept at dealing with the particular requirements of heavily regulated industries.

# Xpress Plug – VeriTran's Software Development Kit (SDK)

In most cases, low-code app builders either create a new app or update an existing app within the low-code tool. In some situations, however, they want to use the low-code tool to add new functionality to a traditionally hand-coded app. For those cases, VeriTran provides Xpress Plug, which is a software development kit (SDK) that application builders can combine with new functionality they create in the low-code tool in order to update a hand-coded app without the need for additional hand-coding.

## VeriTran Marketplace

Even though low-code tools like VeriTran's are ideally suited for creating new, custom-built applications, leveraging pre-built templates and components can accelerate the development process even further. To make such components easily accessible, VeriTran provides a marketplace where application builders can subscribe for elements for their applications. The marketplace also gives VeriTran partners a place that they can offer their wares — including such templates and components and their services as well.

## Managed Solutions

Among the services that VeriTran and its partners offer are managed services: professional services for platform implementation and solution creation. VeriTran offers these managed services both for cloud-based deployments as well as on-premises deployments, where VeriTran's customers are running the entire platform in-house.

# Build better applications faster with low-code

Low-code platforms and tools make quick work out of building applications, with little or no hand-coding required. Not only does low-code save time and money, it also improves the quality of the resulting apps. Another advantage of low-code: it opens up the application-building role to more people in the organization than professional developers, giving the business the ability to take greater control over their custom applications.

## Inside...

- Distinctions among low-code platforms
- Streamlining the work of developers
- Building new applications
- Updating and extending existing applications
- Dealing with organizational pushback
- Modernization strategies for low-code
- Dealing with shadow IT


**Jason Bloomberg** is president of analyst firm Intellyx. He's a leading IT industry analyst, author, speaker, and globally recognized expert on multiple disruptive trends in enterprise technology. He's the author or coauthor of four books, including *The Agile Architecture Revolution* (Wiley, 2013).

**Go to Dummies.com™**
**for videos, step-by-step photos, how-to articles, or to shop!**

**ISBN: 978-1-119-65449-0**
**Not For Resale**


for dummies®
A Wiley Brand

Also available as an e-book

# WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.