

5 Architecture Considerations for Highly Innovative CIOs

Dmitri Tcherevik

WEBINAR TRANSCRIPT

Laura Lewis:

Welcome, everyone. I'm Laura Lewis, and I'm on the marketing team here at Progress, supporting our Kinvey and Health Cloud products. It's my pleasure to introduce today's webinar, "5 Architecture Considerations for Highly Innovative CIOs." Today, we have a highly regarded speaker with us. As CTO, Dmitri Tcherevik leads Progress's vision and technology strategy for cognitive applications across its product portfolio. As a core member of the Progress executive team, he not only executes upon Progress's technology road map, but leads future technology efforts, including incubation projects, technology mergers and acquisitions, and strategic alliances. Dmitri is an industry innovator with a proven track record of creating and evangelizing game-changing technology strategies. Please join me in welcoming Dmitri.

Dmitri Tcherevik:

Thank you, Laura. I would like to begin with adaptive applications. So we will be talking about the five architecture considerations for highly innovative CIOs, and adaptive applications is definitely one of them. And I think it is all about changing the way we were thinking about applications. We are entering the post-application era. And so we used to think of just, "Let's build application for this particular platform." Let's say it's an email application deployed on a desktop. And then people will just have to learn how to use it, and they'll just have to use this particular device. Well, now it's no longer about applications; it's all about experiences. I may get an email notification on my watch, I may read it on my phone, I may start composing the reply on my tablet, and then, finally, add some attachments and send it on my desktop. So there is no email application per se; there is an email experience. And it is all over the place. You see the same in the consumer space. I can start watching my movie on a phone, I can continue watching it on a computer, and then finally move on to a TV. So if you're thinking about developing a mobile application or doing some kind of visual experience for your customers, it's no longer about building one particular app.

(SLIDE)

And we should really stop thinking about applications. It's all about experiences. And if you think that way, then you realize that developing for multiple platforms is no longer an option; it is a requirement. So building a web application and be done with it is no longer sufficient. You need to build for the phone, you need to build for the TV, and for all the different platforms. And it's getting even more complex as we go forward.

(SLIDE)

My phone today -- and I have an iPhone 10 -- is apparently more powerful than a desktop PCs or some PCs or the laptop that I have in front of me. And that really opens up new opportunities. So it's not only a very powerful computer; it also has sensors. And it has more sensors than a human being: it has a microphone; it has a camera -- in fact, two cameras; it has a GPS; it has a bunch of radios. So it's a very powerful device that has multiple sensors. It can create a very precise picture of the environment around it. And it can then use the huge computing power to start modifying that reality. It's what we call augmented reality. And that is yet another interface that must be supported -- or you will have to be supporting in the future.

(SLIDE)

In addition to augmented reality, we also have something that's called virtual reality. And very often, when we talk about virtual reality today, we associate it with gaming. There have been a range of devices produced for gaming -- many interesting games -- but there is really no reason to limit it to gaming and to consumer space. In business, it can be applied equally well. When I was growing up as a kid, I was always fascinated by these flight control rooms with rows and rows of monitors and a huge monitor in front of everybody. And that, of course, has taken a lot of effort -- and very expensive -- to put together. So with virtual reality, there is really no reason to spend all that effort. You can build a virtual control room that can then be used by anybody, anywhere. So that, again, is yet another form factor, is yet another interface that we have to support as part of this end-to-end experience.

(SLIDE)

And then, of course, there is conversational. Speech processing is becoming better and better. Natural language processing is becoming better and better. So, that mechanics has been taken control of. Now it is all about understanding exactly what that human being is saying, and then composing an answer that sounds intelligent. It's all about being able to maintain conversation in the context. And for that, you, again, need certain types of tools and platforms. So many bright minds and everybody is working on this problem. How do we enable our developers -- somebody who may know just web development tools like JavaScript and HTML and CSS -- how do we enable them to build these kind of multi-touchpoint experiences for different platforms and form factors?

(SLIDE)

And what we came up with is called NativeScript. So NativeScript is something that enables a web developer to build what looks like a web application but can then be dynamically compiled and run as a native application on any platform. So what we have here is an example of a virtual reality application. There is a NativeScript runtime, there is an application that was developed in CSS and HTML and JavaScript -- and JavaScript can be TypeScript or supported with Angular or View or another framework. And then there is a set of NativeScript plugin. So in this particular picture, we have a NativeScript plugin that is abstracting away differences between different AR frameworks: ARKit from Apple, AR Core from Google. And by doing that, with a single body of code, you can support these kind of AR experiences on different platforms. So let's play the video. (pauses; plays video)

(SLIDE)

And at this point, what I would like to do is run a short poll for everybody here on the webinar. What are the frontend platforms that you are targeting with your applications and with your cross-platform experiences?

(SLIDE)

All right, so we have web at 89 percent, mobile at 77 percent, voice at 23 percent -- quite good -- and wearable at 23 percent as well. So almost a quarter of us on this call today are thinking about supporting voice and variable. Very good. Quite impressive.

(SLIDE)

So when you start supporting all these new platforms, there is the next problem: how do you make it scale?

(SLIDE)

That problem is this picture here: the monolith. Today, we may be running this in the cloud, or many of us are still running this on-premise. And what we discover is that when you start to deliver a new experience through multiple channels, the load on your monolith grows by several orders of magnitude. And it's not really designed to handle that. And sometimes you can scale it by scaling it out, deploying it on multiple machines. And (inaudible) when you run out of machines [a new form?]. So migrating to the cloud has become the next imperative. And monolith is obviously something that needs to be deployed as a monolith. So the big difference between monolith and the modern architectures is that if you're making a change -- even a tiny change -- to your application, you have to redeploy the whole thing. And there are different ways for developing the monoliths. Over the past several decades, we have been using .NET, Enterprise Java, Ruby on Rails, and other technologies.

(SLIDE)

And here we have another little poll for us to see what platforms you have been using to develop your monoliths.

(SLIDE)

All right, we have Enterprise Java at 42 percent; we have .NET at 33 percent; and other at 17 percent. All right. Very good.

(SLIDE)

So, of course, the next step for us is to move this monolith -- just lift and move it to the cloud. And there are multiple issues that you have to deal with. And this is architecture consideration number two. So this is the structure of your monolith. Typically, it is an Enterprise Java or .NET application that's sitting on top of your database or system of record such as Salesforce. And you're using an interface like JDBC or ODBC to access your data. And then at the top there you have your application -- and that can be a web application or a mobile application or a voice or a wearable. Now you need to move it. And the first question you're confronted with is, how do I move my data? Because very often -- particularly if you work in financial services or health care -- moving your databases and systems of record is not an option. For compliance reasons and security reasons, they need to remain on-premise. And yet, you would like to move your application.

(SLIDE)

So some approaches that have been used is to poke holes in the firewall [by at least two IP addresses], jump through hoops to make that work. But that is risky, right? You never know what kind of security holes and risks you are creating by working with your firewalls in ways that it was not intended to.

(SLIDE)

And for this particular problem, we have what we think is a very elegant solution. It is called a hybrid data pipeline. Hybrid data pipeline consists of two components. One of them is deployed with your monolith. It is a library or a process that is running on the same machine and providing the same JDBC and ODBC interface that your monolith has been written to use. And on-premise, you deploy what we call a hybrid data pipeline agent. It can be running on a machine and in your data center and use the same JDBC/ODBC interface to go after databases and systems of record. And then the two are working together to ensure secure and scalable communication between your monolith and the agent. So now you can have any number of instances of your monolith running in the cloud, you can scale them dynamically based on demand, and all of those instances have secure access to data through HDP. So this has been taken care of, right? This first step.

(SLIDE)

And the next step is microservices. And the reason for microservices is really agility. When you deploy your applications and you support multi-channel experiences, what you discover is that your application on your backend has exercised a lot more in ways that you have not foreseen or intended. And that, of course,

uncovers all kinds of bugs. And your applications on your backend need to be updated on a much more frequent schedule. Very often, a monolith can only be updated once a year or once every few months. But when you're supporting multi-channel experiences, you need to do a continuous deployment -- update sometimes on a daily basis -- and, of course, the monolith was not designed to do that. And hence, the need for microservices. A microservice is a service that is running as an entity that can be deployed independently. And it often supports just a single step in a business process -- or a simple business process. When you are making changes in the process or when you're fixing bugs in that particular service, you can deploy it independently. And it's called a microservice for that reason. Obviously, it is not possible to just rewrite the monolith as a set of microservices in one day. It has to be a gradual process. And what we have here is the next architecture consideration: our recommendation for how you can [accomplish this?] with your application.

(SLIDE)

So the first step is to introduce what we call an API gateway in the backend as a service. It is abstracting the API that is exposed by your monolith and used by the application. Now that you have this middle tier, you can also introduce cache in. And that cache can be used to immediately speed up your application. We have seen cases where an application would go after a system of record -- like SAP, for instance -- and sometimes, a request can take up to a few seconds. But by introducing this middle-tier cache, you can speed that up immensely, to a fraction of a second. So there is a huge benefit in doing just that. And, of course, you still have HDP to provide secure access to data on-premise.

(SLIDE)

But once you introduce this layer, the next step is that you can start introducing microservices little by little. And the process is called "strangling the monolith." So, let's say you're rewriting the service or you're fixing a bug somewhere. Instead of changing the monolith, you're introducing a new service. So let's say you want to create a new user onboarding process. So for that particular process, you may create a new microservice. And, by the way, you have a whole number of options now for implementing it. If your monolith is built as an Enterprise Java application, this particular microservice can be implemented as a JavaScript application and deployed on [Node.js?]. And then you deploy that new service side-by-side with the monolith. And you use the API gateway to route the requests transparently to the application. So as the time goes, you can see how, little by little, you can start replacing services that run inside the monolith with microservices that run outside of it. And eventually, you will strangle the monolith -- you will rewrite it entirely. It may take you a month; it may take you a year; but you will eventually get there. So that was another important architecture consideration: introduce the API redirection layer in the form of an API gateway or backend as a service or a system that combines the two, and then use the API gateway to transparently route requests between your monolith and the new services that you introduce. So as I mentioned, a microservice is used to support a single step in a business process or maybe a simple business process. But a microservice needs to be deployed and managed as an independent service. Very often, you allocate a virtual machine just for that service -- to manage it independently. The overhead of doing that may still be too high for certain operations.

(SLIDE)

And here is another architecture consideration: event-driven functions and serverless. So with this, you can take care of operations at high granularity -- smaller operations that are event-driven. So an example of this would be a new user sign-up. Something needs to happen every time when a user signs up to your application. That is an event that can be handled by a function.

(SLIDE)

So it is good if your BaaS and API gateway component supports cloud functions. A cloud function is something that is dynamically spun up -- launched -- in response to an event, performs the work, and then shuts down. So that also allows you to manage capacity in your cloud dynamically, in response to the workload. Another good example of using functions would be, for instance, content conversion. Very often, you allow your users to upload content into your cloud, and that content must be transformed before it can be presented in applications through multiple channels -- for multi-channel experience. So that conversion, again, can happen in that event-driven function. So you launch FFmpeg, you convert to a video, and you shut it down. Very fast, very efficient. So that is the architecture overall. And you see it is now very modern: it is using microservices; it enables continuous deployment; it is using functions to save you some costs; capacity can be dynamically provisioned; there is no need to pre-allocate capacity. So overall, this new approach should lead to a lot higher agility in your organization, ability to continuously deploy and update your applications, and at the same time reduce costs due to better [use of?] elastic capacity allocation in the cloud.

(SLIDE)

And this is how this whole picture can be fulfilled with the products that we offer at Progress. So there is a hybrid data pipeline. It is part of our DataDirect product. We offer Kinvey, our backend as a service. And Kinvey Backend as a Service has that middle-tier cache, and it also offers an API gateway as part of it. And then on the frontend, you have NativeScript, the framework that can be used to use one body of code, one language, one set of tools to support applications running natively on different platforms.

(SLIDE)

So at this point, I would like to do another little poll here and ask you, what are the cloud infrastructure providers that you use in your organization?

(SLIDE)

All right. Very good. We have Amazon at 47 percent, Microsoft at 32 percent, and then other at 16 percent. Very good. So this would be very consistent, I think, with our observations. And as I mentioned, with Kinvey, you have the benefit of being able to use any of these clouds. And we heard on multiple occasions that companies would often use different cloud providers for different workloads, so that kind of flexibility is very important.

(SLIDE)

Good. And what I would like to do next is talk about cognitive application. What is the cognitive dimension to it?

(SLIDE)

And this is becoming more and more important. And it is because of the abundance computing power and plentiful data. The amount of data is doubling every 12 months out there in the world. And now we have the computing power in the form of cloud computing and GPU computing that enables us to handle that data. And this is creating interesting new use cases. Speech recognition, natural language processing, predictive maintenance, facial recognition, biometric authentication -- all of those depend on cognitive computing in the form of data science, statistical learning, machine learning, neural networking, and other approaches.

(SLIDE)

And what's common about all of them is that it's not about algorithms per se -- it's about data. It's about looking at the data, understanding the data, cleaning the data, massaging the data to match your algorithms, and then selecting the right algorithms to process that data. So algorithms are all open-source -- you can go and download the most sophisticated tools out there. But to really apply them in the right way to the data your organization has, you take a certain kind of people, called data scientists. And they are extremely popular, extremely well paid, and extremely difficult to find. So with a new approach called meta-learning and machine learning automation, this problem can be somewhat addressed. With meta-learning and machine learning automation, features can be automatically extracted from the data set. You can also take and extract what we call meta-features, something that describes the data inside the data set at the whole. And then based on those meta-features, you can then automatically select an algorithm or a set of algorithms that are best designed to process that data in order to achieve the results that you need. You can then automatically create several models, run those models, compare their performance to each other, and select the one that does the job in the best possible way. And that is the model that we will be deploying. So that whole automation process can be performed by certain kind of systems that support this new approach called meta-learning.

(SLIDE)

And that is exactly the approach that we offer with a product called DataRPM. With DataRPM, not only do you get the models that you can use to do predictive maintenance or cognitive anomaly detection, you also get very sophisticated tools that raise the productivity of the data scientists. So with the help of those tools, you don't need as many of them. You can achieve the same results within a shorter period of time.

(SLIDE)

All right. So at this point, I would like to thank everybody. And you can find more information at this link here.

(SLIDE)

Laura Lewis:

All right. Thank you, Dmitri. We're going to open up this discussion now and answer any questions that have come in. Our first question today: what is the difference between an API gateway and a backend as a service?

Dmitri Tcherevik:

So an API gateway is just a router. It is something that you can use to publish your API. And there are several ways to describe that API to the applications. And then you also describe routes -- you know, this particular request needs to be handled by this particular microservice. And the API gateway is responsible for performing that routing. A backend as a service is much more than that. It is also a collection of common services that may be required by your application, such as authentication, for example, single sign-on, assistance, automatic data synchronization between clients and the cloud, and other services. And a backend as a service often includes an API gateway as one of the components.

Laura Lewis:

Staying on the topic of backend as a service, can a BaaS replace a microservice's architecture?

Dmitri Tcherevik:

Yes. A BaaS -- backend as a service -- does not replace a microservice's architecture, it is part of it. So a backend as a service offers a collection of microservices that we believe the majority of the applications will need -- as I mentioned, assistance, integration with backend systems, authentication, user management, user onboarding, push notifications, and others. These are all implemented as microservices. What you also have is an option of adding your own microservices to the set. So a backend as a service becomes a platform for deploying microservices. And with something like Kinvey, you can be developing your microservices in JavaScript and then deploying them transparently through the infrastructure. They will be run as microservices on the infrastructure. You can also be building cloud functions in JavaScript. And, of course, you can be doing top-to-bottom development testing and debugging with NativeScript, which is also based on JavaScript. So yeah, backend as a service is an integral part of your microservices architecture.

Laura Lewis:

What about when we break the monolith to microservices? What tool sets or [governing?] processes are recommended to orchestrate NativeScript? Is it a number of microservices now -- maybe something might get out of control if there's not proper governance?

Dmitri Tcherevik:

What you often see is that your business processes is what's guiding the number of microservices. Typically, it is one microservice per process, or maybe one microservice per important step in the process. So that, I think, puts a certain physical limit on the number of microservices. And you may also have a few of them for supporting services like user management, but typically, that is the limit. And another limiting factor is the number of teams working on your microservices. Previously, with your waterfall monolith development, you would have teams structured by function -- so you would have a frontend team, a team of designers, a team of testers, a team of developers, a DevOps team. With microservices, what you will notice is that eventually, you will want to reorganize around microservices, because very often, (inaudible) functional team is responsible for a particular microservice, like user onboarding. And so the number of people in the organization will be another limiting factor for the number of microservices that you can deploy and manage.

Laura Lewis:

How is the frontend of the monolith application exposed via the cloud?

Dmitri Tcherevik:

It depends on, I guess, the area that it is coming from, but fairly recently ones like Enterprise Java monoliths and .NET monoliths -- they would be built in accordance with the service-oriented architecture, so they would actually follow this kind of SOA structure, and then different services would be implemented as modules in the monolith, and they would have their own endpoints and API. And that API would be either SOAP or REST -- but something that you could then call from a web application or mobile application. So a monolith has an API, and the different API calls are mapped to different modules in that monolith. And that API can then be invoked from the apps. And this whole transition to microservices is all about taking those modules and making them into separate microservices that you can then manage and deploy and update independently.

Laura Lewis:

Second part of that question: how is the backend as a service connected with the monolith system? What changes are needed on the monolith application to support this?

Dmitri Tcherevik:

Thank you. Yeah, very often, no changes, I would say, in the monolith. So a monolith, as we discussed, usually has an API of some sort, and then your API gateway would be routing requests to that API. Now, if your monolith does not have an API -- let's say it was implemented as JavaServer Pages or ASPs -- Active Server Pages -- in that case, there is a task of creating that API. But that, very often, can be done in a rather straightforward fashion using tools provided by the Java community or the .NET platform.

Laura Lewis:

I just have to say, we have a lot of really good questions coming in, everyone, so keep them coming. Switching gears a little bit, I like some of these product questions that I see coming in. So why should we choose Kinvey as a backend as a service?

Dmitri Tcherevik:

So, first of all, Kinvey and NativeScript, they offer a complete solution, and they are very well integrated together. So if you are thinking about supporting multi-channel experiences, then I would definitely recommend starting with NativeScript, because all your web developers suddenly become mobile developers and variable developers. And then from there, NativeScript is pre-integrated with Kinvey for authentication. So if you need to do single sign-on with an enterprise authentication provider like Active Directory, that module already exists. There's just a couple of lines in NativeScript, and Kinvey will do all the magic on the backend. And then Kinvey is just a complete collection of services, so if you need to develop a super-sophisticated app, it will give you 98 percent of the services that you need to implement that app. It is backend as a service, right? So your backend is fully [baked?]. And in fact, this is what we see in many teams switching to Kinvey, is that pre-Kinvey, you have a 50-50 split between frontend developers and backend developers; post-Kinvey, you have maybe 80-20 -- 80 percent working on the frontend and only 20 percent implementing Kinvey services and functions. So it is just a very robust platform that is getting you 80 percent there.

Laura Lewis:

From a quality assurance perspective, how do you automate the testing of applications built using NativeScript? Can a testing framework be built using NativeScript?

Dmitri Tcherevik:

Yes. So from the testing perspective, a NativeScript application is a native application for all means and purposes. So all the testing tools that are used to test applications on Android and iOS and web are fully applicable here. And there are wonderful tools both online -- something that you can use as a service, test your application on multiple devices -- and there are also tools that you can download and use. In fact, Test Studio from Progress can also be used to test native applications built in NativeScript.

Laura Lewis:

What about its security encryption? Is there anything between NativeScript, SDK, and an API gateway?

Dmitri Tcherevik:

Yes. If you are using the standard SSL protocol, all the communication is secure and encrypted. And then in addition to that confidentiality through encryption, you can also add authentication and authorization that [does?] accomplish the standard protocol, such as OAuth. And so both NativeScript and Kinvey support that out of the box.

Laura Lewis:

So within the DataRPM product, how does it make our data scientists more extensible? What is the ROI that can be expected? Is it something like one data scientist now equals three?

Dmitri Tcherevik:

I don't think you can make that kind of precise measurement, because it really depends on the problem domain that you are working in. But what you can expect is that your data scientists will definitely be a lot more productive, and that you will be able to automate repeatable tasks. They will be more productive because DataRPM, out of the box, offers the widest selection of algorithms for data transformation, data preparation, and feature extraction. It offers a very broad selection of machine learning algorithms. And in addition to that, DataRPM offers a workflow engine that can be used to build recipes and repeatable practices and then string them into processes. So if you have new data coming in, let's say, on a weekly basis, you can run your workflow processes -- batch processes -- on a weekly basis to produce and train models, and then push those models into production. So DataRPM is doing to machine learning what some of the other tools are doing to DevOps -- this kind of automation.

Laura Lewis:

And do you have any architecture considerations for offline support mobile applications?

Dmitri Tcherevik:

Oh, yes. Absolutely. So offline is extremely important for mobile, because you may be intermittently connected -- your network connection can be poor. And you get that out of the box with NativeScript and Kinvey. Kinvey is the case for different platforms. They support transparent synchronization of data between a cache -- offline cache -- that is resident on the device and your data repository in the cloud. So you get that support automatically, out of the box.

Laura Lewis:

Could you say a few more details about the Hybrid Data Pipeline that would allow financial services companies to consider it -- something about maybe security?

Dmitri Tcherevik:

Absolutely. So as I mentioned, with Hybrid Data Pipeline, you get automatic encryption, authentication, access control -- so the communication between the Hybrid Data Pipeline agent that is running in your data center and the Hybrid Data Pipeline engine running in the cloud, that communication is fully secure. And it just happens for you out of the box, right? There is no reason and there is no need to [whitelist?] a new IP addresses to poke any holes in the firewall. Our Hybrid Data Pipeline delivers that guarantee to you.

Laura Lewis:

On that same topic, do you also offer or recommend any end-to-end monitoring tools to ensure the overall system is operating efficiently?

Dmitri Tcherevik:

Well, application management -- it's a broad discipline, something that can be used to monitor your application. Kinvey has a very sophisticated application monitoring console that can give you a lot of insight into the health of your application and the usage information of your application. Tons of statistics are collected, not only from the microservices running this part of the (inaudible) cloud, but also from the SDKs instances running on the client side as part of your applications. So, yeah, Kinvey offers you a very sophisticated console. And it can obviously also be extended with any other tools that you may be using to manage the infrastructure overall in your company.

Laura Lewis:

All right. Still a few more in here. You guys have been putting in a lot. Thank you so much for the engagement. I'm glad you're finding a lot of value out of this. So from Mark, we have: what is the strategy for separating out monolith to microservices?

Dmitri Tcherevik:

Well, the strategy, as I said, is called "strangling the monolith." Once you introduce an API gateway or a BaaS and you can route-request transparently, then you would start -- every time you need to rewrite a module, every time you find yourself fixing lots of bugs in the module in your monolith or every time when maybe a business process changes and you need to change it significantly in that module inside your monolith, that is a signal to you that you need to rewrite it now as a microservice. And so you pick the best tool -- the most modern tool -- the tool that your developers are most familiar with, and you develop it as a microservice. A microservice is something that can be deployed and managed independently, so you may set up a tool set for that and a team to manage that, and just deploy it.

Laura Lewis:

Second half of that question: and what benefits would be accomplished moving more code to microservices?

Dmitri Tcherevik:

Well, the biggest benefit is agility. So now with the monolith, it is called a monolith because every time you fix a bug, you have to wait for the next update cycle. And the larger the team supporting the monolith, the bigger the distance between updates. It can be up to a year. But with microservices, you can implement what's called continuous deployment. So as soon as a bug is submitted to GitHub or another source code control management system, the change can be automatically picked up and pushed into production after some developer testing and release testing. So I would say agility is the biggest advantage. But then you also get the ability to scale your application in a lot more efficient way. With monoliths, very often, you have to pre-allocate capacity for peak loads. So if you know that the biggest load, let's say, is on holidays, you have to have 15 machines in your [form?] to handle that load. With a monolith, very often, you have to pre-allocate around those 15 machines. With microservices, very often, you can dynamically spin them up, and you can also scale a particular service without affecting other services. So, for example, user onboarding service can be scaled independently, and for that, you may need three instances instead of 15.

Laura Lewis:

How do you best determine the granularity of microservices so that it's manageable and yet small enough and distinct, with no overlaps?

Dmitri Tcherevik:

So I would say granularity is probably best determined by the size of the team that is required to support it, and that is three to five developers. So if you have something that is big enough or small enough to be managed by three to five developers, that's a good candidate for the microservice. That's one way of segmenting it. Another way of segmenting it would be to map it to your business process, as I said. So if there is a particular business process -- like email nurturing, let's say, or user onboarding, or video transformation -- that's a good candidate for a microservice.

Laura Lewis:

And with an increase in the use of microservices, is there a heightened risk of communication failures between different services that could cause issues with applications?

Dmitri Tcherevik:

Well, whenever you introduce network communication, there is always a risk of failure. But I think it is a risk that we very well know how to manage through duplication and redundancy. And if you take a traditional multi-tiered monolith, that same risk is there, right? Because you need to communicate across tiers and you need to communicate across machines in a cluster. So I would say this risk is not new. It's just a matter of structuring your application in a different way.

Laura Lewis:

Does Kinvey have any integrations with leading CI/CD tools?

Dmitri Tcherevik:

Yes. Everything that you can do with Kinvey is available through a command line that is infinitely scriptable, and those scripts can be run as part of a CI/CD tool pipeline or workflow -- any tool, basically.

Laura Lewis:

And how does NativeScript compare with Microsoft Xamarin, which -- all have the same objective?

Dmitri Tcherevik:

Yes, they have the same objective, but Microsoft Xamarin is targeting the community of .NET developers. So you need to be a .NET developer and you need to know C Sharp. With NativeScript, we are targeting the army of web developers. With NativeScript, the only thing you need to know is JavaScript, CSS, and HTML.

Laura Lewis:

A few more in here. How do you overcome the cultural resistance of a conservative, predominantly on-prem, and private cloud IT department?

Dmitri Tcherevik:

Well, I think it's all about benefits, right? It's about articulating benefits. They must be under a lot of pressure to fix bugs and get those fixes to production quickly, and with microservices, that goal can be achieved and automated. So it's a matter of addressing their top concern, which is reliability and agility and resilience.

Laura Lewis:

What security are you using to enforce backend access?

Dmitri Tcherevik:

Well, that's a very broad question, and so it's -- as I mentioned, encryption -- everything must be encrypted with SSL and HTTPS. Then, obviously, you need to have very good authentication. And authentication needs to be done using your established and existing authentication providers, like Active Directory. And there are connectors and integrators for that. And then it is also access control. And access control can be performed at multiple levels. At the API level, only certain apps can be given permission to access certain API. At the level of your content assets, only certain users belonging to a certain role can have access to certain documents. So this is what's called policy-based security, and there are ways of defining that policy in

different places -- in Kinvey and also in outside providers.

Laura Lewis:

Thank you, everyone, for joining our webinar today. These were all great questions, and I want to thank everyone for submitting them. I'd also like to thank Dmitri for joining us today to share his knowledge. We hope you enjoyed the event, and look forward to having you on future Progress webinars. Be sure to check out our previous presentations and follow us on Twitter, LinkedIn, and Facebook. Thanks again, and have a great rest of your day.

END OF AUDIO



VIEW WEBINAR

About Progress

Progress (NASDAQ: PRGS) is a global leader in application development, empowering enterprises to build mission-critical business applications to succeed in an evolving business environment. With offerings spanning web, mobile and data for on-premise and cloud environments, Progress powers businesses worldwide, promoting success one application at a time.


Learn about Progress at www.progress.com or 1-781-280-4000.


Worldwide Headquarters


Progress, 14 Oak Park, Bedford, MA 01730 USA

Tel: +1 781 280-4000 Fax: +1 781 280-4095

On the Web at: www.progress.com

Find us on  facebook.com/progresssw

 twitter.com/progresssw

 youtube.com/progresssw

For regional international office locations and contact information, please go to

www.progress.com/worldwide

Progress is a registered trademark of Progress Software Corporation and/or one of its subsidiaries or affiliates in the U.S. and/or other countries. Any other trademarks contained herein are the property of their respective owners.

© 2018 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

Rev 2018/04 | RITM0016998

