EPISODE 515

[INTRODUCTION]

[0:00:00.4] JM: Applications need to be ready to scale in response to high load events. With mobile applications, this can be even more important. People rely on mobile applications such as banking, ride sharing and GPS for their day-to-day jobs in life. During Black Friday, a popular ecommerce application could be bombarded by user requests. You might not be able to complete a request to buy an item at the Black Friday discount. If you attend the Super Bowl and then you try to catch an Uber after leaving the Super Bowl, all the other people around you might be summoning a car at the same time and the system might not scale appropriately.

In order to prepare infrastructure and mobile applications for high volume, mobile development teams often create end-to-end load tests. After recording incoming mobile traffic, that mobile traffic can be replicated and replayed to measure a backend's response to the mobile workload.

Paulo Costa and Rodrigo Coutinho are engineers at OutSystems, a company that makes a platform for building low-code mobile applications. In this episode, Paulo and Rodrigo discuss the process of performing end-to-end scalability testing for mobile applications backed by cloud infrastructure. We talked about the high-level process of architecting the load test and explored the tools used to implement it.

Full disclosure; OutSystems, where Paulo and Rodridgo work at, is a sponsor of Software Engineering Daily.

[INTERVIEW]

[0:01:40.3] JM: Paolo Costa and Rodrigo Coutinho work on engineering at OutSystems. Guys, welcome to Software Engineering Daily.

[0:01:46.7] RC: Thank you.

[0:01:47.4] PC: Thank you.

[0:01:47.9] RC: Nice to be here.

[0:01:48.6] JM: Yes. It's great to have you. You both work on a platform for building mobile applications and people use these pieces of technology to build mobile apps and this mobile need to work properly. Part of testing a mobile app is load testing. So we've done several shows about backend load testing, but mobile load testing is slightly different.

Paulo, why don't you explain why mobile load testing is different than just simple backend load testing?

[0:02:21.3] PC: Yeah. In fact, you're still testing the backend, but you're doing it using applications that are distributed through devices that each person has a single different device. It can scale to values that are not usual on web load testing. Also, the construction of the applications itself, the way it interacts with the backend, is slightly different, because it allows you a more one-to-one approach in terms of requests and answers. A mobile app usually makes — I'm going to the technical bit here, makes a POST supposed and retrieves a GET, and that's the core thing for mobile app to do to make a POST, a request, and get a simple answer right away.

Web application is slightly different, can have some differences on the architecture, and one request can trigger a lot of request on the backend and so it's easier for a slightly, more performant dedicated task to be done on a mobile app.

[0:03:28.8] JM: Okay. The main difference in load testing that I can think about is if you're load testing solely a backend system, then you could just have everything run in — Like if you have a cloud — If your application runs in the cloud, then you could do all of the testing just remotely in the cloud, but with mobile applications, you've got this added dimension of many, many users that are all separated by network connections with different latencies. Every additional user has their own smartphone and there're pros and cons to that in terms of the scalability of a system. So on the pros side of things, in a sense, you've got auto-scaling, because you have every user that has their own computer, their own client. In a sense, you've got auto-scaling, because the more users there are using a mobile application, the more access you have to their mobile

devices and you could theoretically have load be distributed to those mobile devices. But that's not really how applications are built these days. We've got a lot of logic on the server, and so the mobile applications don't necessarily — When you're scaling the amount of mobile applications, it's not like that's magically giving you a linear equivalent amount of server capacity.

Can you maybe explain how — As the number of users grow for a given applications. If we're talking about a real-world scenario, like a Black Friday, how does server load typically scale relative to the growth in the mobile load?

[0:05:08.8] PC: Yeah, that's a terrific question. First of all, you're absolutely right when you say there's a kind of an auto-scaling for the — Because you have the mobile apps, the mobile phones for each user growing immensely. This translates into an amount of load to the backend, that is sent to the backend that in a peak scenario, like the Black Friday one and is in a completely different order. It doesn't grow by two times or three times to the regular, because in our case it grew like a hundred times since the first test. So it's something in that scale.

What you're trying to test here is everything, and I mean everything from the moment the request exists the mobile device and reaches the frontend that is putting all the logic, is processing all the logic. So there're a lot of components in the way. You have, like I said, the network. Once it reaches our infrastructure, you have to test your load balances, you have to test the frontend capacity, you have to test the CPU ability to process all the requests, the ability to connect the database, the ability for the database to answer in good fashion. There are so many components to test that it's a different world from the web tests.

What we try to do here was we try to simulate, and we achieved it. We tested every component from the mobile app of the end user to the database of our backend. So we tested every single component in the middle.

[0:06:52.1] JM: And to do this end-to-end test, you chose a mobile banking application. I have several different mobile banking applications I've used overtime, and I think this is a pretty good prototypical application. Explain why that is? Why is a mobile banking application pretty good for — If you're trying to just generally load test your infrastructure, because, again, you have a

platform where people build whatever mobile applications they want. So you want — If you're going to do a load test of that systemic infrastructure, you want a mobile application that can go through all of the typical paths of different programs that people might build. So why did you choose a mobile banking app?

[0:07:36.7] PC: You said earlier — You mentioned the Black Friday scenario, and there's no better application to test than a banking application for that kind of scenario, because — Let's, for instance, take a social app. You have a social app. You don't use a social app more during Black Friday than the other days. Okay, you might check some extra post on a social network, but you don't have that kind of growth. You don't really stress your infrastructure in a particular day with those kind of apps.

On a banking app, you do. On a Black Friday, you spend a day going through your mobile app and check the credit card's balance, the aggregates of your several accounts. Like you said, you have several banking accounts in your mobile phone. So you probably are a very good use case for us. Once you reach a day like Black Friday, you're always checking your accounts, you're always checking the balance, you're always looking for, "Okay. Is this a good deal or not?" and the mobile apps do have a very, very high deep usage in those days.

Fortunately, we have a customer that was able to share with us numbers for that peak usage, and since we have that date and we could collaborate our assumptions with that data, we were able to, okay, establish a goal and go for it. The banking mobile app was the perfect scenario for us to test.

[0:09:04.2] JM: You've got load balancers, backend servers, databases. Give an explanation for the requirements of the test and how you expected the load to traffic through the system. Give an overview for the bottlenecks that you expected to see and how you expected your system to respond to the additional load.

[0:09:31.4] PC: That was kind of the question we faced when we started the test. How is this going to respond? Because our target was clearly to have the better user experience possible for the end user. We wanted to end user to be completely abstracted from having one other user on the network or 11,000 users or 20 million users like we tried to simulate using the same

application as it was. So the idea was to always provide the better user experience possible and, for that, well, the bottlenecks, we expected to fine — We imagined the CPUs would go crazy on the further test, on the latest tests that were really, really hard on the servers. We expected to have issues with the load balancers. We didn't have any experience regarding this kind of loads in round robin strategies or whatever strategies for load balancing. So we were expecting to face a lot of bottlenecks on the database side. We had no idea how will the database respond to this specific use case. So we tried to cover all those infrastructure bits in the test, and we did it. We actually took a lot of conclusions from that.

[0:10:59.4] JM: Did you expect the user experience to degrade at all during the test or did you expect the user experience to smoothly adjust to how your backend infrastructure was scaling in response to the load test?

[0:11:15.6] PC: Yeah, that's a good question, because empirically we would assume — Everyone would assume, "Oh! If I have lots and lots of users, the user experience would degrade overtime," but we also understood that with our experience regarding the usage, the high peaks with other applications, not on this level, but we know that, for instance, using a SQL database on cloud and on RDS, overtime things tend to adjust, because the maintenance plans are going to place, there are statistics copulated, stuff like that that usually translate into a slightly better user experience.

We were kind of trying to figure out how is this going to — is this really, the empirical side is going to be correct. Like will we have the load time, the response times degrading is our things going to be more academic and have the response time slightly improving overtime? There were two options on the table and the test gave a clear answer on that, but I'll keep you in suspense a little bit more about that.

[0:12:23.2] JM: Okay. What was the spec for the amount of load that you were going to administer on the architecture and how are you planning to administer that load?

[0:12:34.6] PC: Yeah. We were given the application that was developed in-house for — If I'm not mistaken, it was our demo team that had the application, or at least one mobile app built, and we were given the app. Because we are a very agile company, we have a very, very

amazing way to debug applications. We used the method of capturing every action of the mobile app, and the way to do it is very simple, it's actually use the app, connect it to your computer and retrieve everything from the browser that it's connected through USB debugging. I captured every action that I made in the application and I was acting like an absolutely regular user. I opened the app, I went to check my account statements. I went to check the details from one account and I changed accounts. I went to check another detail. I made several actions like that, and that simulated the usage of a real mobile user.

So once I have that script, I was able to, "Okay. I know exactly what a user does, and I can start to plan to inject loads using this script in a very high volume."

[0:13:54.8] JM: So the simulation for the load test, did you have simulated frontend clients or did you record what happened on the backend when you had a single frontend client and then just blast the backend servers with the simulation that you recorded.

[0:14:11.6] PC: Exactly. I recorded what arrived at the backend. So I simulated using the app with real usage and everything that reaches the backend can be captured. So I recorded those actions, every input, and I was able to record the complete script for testing.

[0:14:33.0] JM: Does that test the network as well?

[0:14:35.0] PC: It doesn't test the network, but we did take it in account. First of all, it doesn't test the network, because we're connected through a USB, to the computer. So there was no network through their default, but there is an option and we used it on the tool that we used to simulate several metric degradation factors. One user is using 3G, the other one is using Helm and is able to use Wi-Fi with a lot of speed, a lot more speed. So there's different latencies involved. The test took that in consideration, yes.

[0:15:08.4] JM: What are the actual tools that you need to build a load test? Do you just cobble together different batch scripts or are there actual tools that you can use to do a load test?

[0:15:20.0] PC: There are tools. First tool you need to do a proper load test is use OutSystems. It's the starting point, and then you start looking around for tools that are able to, for instance,

record the script that we needed. For that, we simply use Google Chrome and we were able to record the script for later usage, then converted the script into a format for JMeter. JMeter, which is an open-source tool, it's community-driven, it's very, very powerful, very flexible, a lot of plug-ins. One of them is, like I said, the ability to simulate network conditions, and we converted the script to the JMeter format, and from there on we were set, absolutely set.

[0:16:09.9] JM: Okay. Can you explain what JMeter is in more detail?

[0:16:13.0] PC: JMeter is a tool that allows you to — It has the ability to also record on its own the usage of a mobile app into the backend. It can record, it can adjust the requests that are made, you can transform. For instance, when I make a log-in, I'm going to record the request that has my username and my password, but I don't want to inject a thousand times in a minute my username and my password, so I need to transform that into variables. JMeter is an extremely simple tool that allows you to do that. It allows you to run the tests with the predicted loads. It allows you to make ramp-up times. It allows you to generate a lot of statistics, very raw files of statistics that can be edited and processed later. It's a very complete one, and since it's completely open-source, I'm able to say go ahead and test it. It's a very good tool to help developers during their development time.

[0:17:13.4] JM: You're on AWS. What are the AWS services that you're using within your infrastructure? So we can start to get a better picture of what exactly is being load tested.

[0:17:26.0] PC: Yeah. Very basic services; the EC2 for creating machines, Windows servers, Window Server 2012 was used a base-operating system. We installed platforms serving those machines. Then we had the RDS service. We had SQL database.

[0:17:44.5] JM: Relational database. That's relational database service.

[0:17:45.9] PC: Yes. Relational database service. We used Microsoft SQL Server, and we had the ELBs, so the load balancers, and that's basically it. That's all you need to set up the proper infrastructure for having OutSystems working like a Swiss clock on your cloud.

[0:18:04.5] JM: Right. Why isn't there an off-the-shelf way to do these kinds of end-to-end load test — I guess, JMeter is kind of an off-the-shelf tool, but it still sounds like you had to do significant amount of engineering to test — This was a serious project and it just seems like the kind of thing that, I'm surprised, there's not an off-the-shelf way to do mobile load testing.

[0:18:28.5] PC: Yes. I guess it's because of the complexity. You have different requirements for each application you build. You have a different approach for every test, because currently there's a lot of discussion in the dev ops community about the definition of load tests, stress tests, performance tests. I guess that each company and each team developing applications has their own vision about what I need to test and how I need to test and what's my expected outcome. It's very hard to translate this into a single application or a service from any company to create a service that pleases all.

Then comes the analysis part, and the analysis part is very, very subjective to what your customer is expecting. So it's hard to create something that is not standard among development, and developers have different visions for it, and how can you create something that is standard for that. There are companies that actually try to solve this by providing load testing as a service, and one of the companies I used a lot of references from them to understand how to create the script, how to optimize the script for high usage. I saw that those are the guys that are way in front of the others, but they are still not able to provide an end-to-end solution, and there's another thing that we need to consider.

You're not testing the device itself, and there's a lot of confusion about — There's a gray area about that, because customers usually expect us to — When they say, "Oh, yeah. I want to load test application. I want to understand how it will perform on the mobile device." We won't test — In this kind of test, we are not trying to test the mobile device performance, because there are hundreds or thousands of devices out there. It depends on the usage that the owner gives to their mobile phones and it's impossible to test end-to-end that. You can test the mobile devices. There are services for that. You can test the networks. You can test databases. You can test a lot of things, but they are not glued together in one single service.

We actually built a service for that in OutSystems, and we do have mobile load testing services in OutSystems, but it's always something that has to be discussed with a customer to

understand exactly what they are trying to do and we give them our vision of the thing. It's basically an agreement between OutSystems and customers on what we expect — What the customers are expecting to be tested and what can we give them as an outcome.

[0:21:15.8] JM: Right. That makes sense. If I'm some mobile developer developing my own iOS app from scratch, there's really nothing that I can take off-the-shelf, because the ways that mobile applications are built when you talk about from top to bottom are just so heterogeneous that it doesn't really make sense for there to be something that is like a service or a company that is built entirely around that mobile load testing.

[0:21:42.3] PC: Yeah, exactly.

[0:21:44.7] JM: When you were doing these load tests, you were putting a Black Friday level of traffic load on your infrastructure. Did you spin-up some reserve infrastructure, some test infrastructure in order to absorb that load or did you just use for your production infrastructure?

[0:22:03.5] RC: We created a new one. We created a new one. This is a serious project, like we said before, and we wanted to make no mistakes on it. We wanted to simulate that, "Okay. We are a customers and we're trying to build an application that will serve a Black Friday scenario for end users." So we have our own infrastructure for this, and we couldn't risk putting this into our production infrastructure and modifying the results of the test, because we were already using that infrastructure, a dedicated one for this test. So we spun-up a new one entirely for this.

[0:22:39.2] JM: So once you had it spun-up, describe what happened when you ran the load tester. Maybe you could a little bit more about numerically what exactly the amount — We can say, "Okay. This is simulating — We want to simulate Black Friday traffic across a mobile banking application." What is that look like in terms of numbers?

[0:22:58.6] PC: The Black Friday scenario was extrapolated the numbers that we had from a very large American bank regarding the user-base and we had the data from a European, very large bank also, and we had the data for the most or the greatest number of requests ever recorded in their infrastructure. We had their user-base, the European bank user-base and the largest number of request ever recorded to reach the backend servers at the same second.

So we just extrapolated and we reached the conclusion that for a 20 million user-base, we could simulate something like 11,247 requests to be precise. We round it up to 11,250 requests per second. This is not 11,000 users running scripts to simulate load. It's per second reaching the backend. Those are the numbers that hit our servers. It's a big thing.

Yes, infrastructure obviously was not correctly set up at first. It would be basically a longshot to do that and get the ideal result. Even if we had enough servers to answers to those requests, we were perfectly comfortable with the idea that we would have tune our systems, we'd have to tune something in our platform configuration to achieve optimal results that was expected.

I usually tell to customers doing load tests that if you run a load test and at first it gives you the number you're expecting. You're doing an under appreciation of what you can achieve, so because you can always optimize something. You can always get a better result. For that, we started not with this number, this 11,000 request per second. We started with a number lower. When we ran a lot of tests to understand how servers were responding, how many transactions per second we were closing on the IIS level accordingly or in correlation with the number of request we were putting on the severs. We did a lot of math in between to understand, "Okay. How can we scale this to the numbers that we want?"

One thing for structure itself, we started with the N4X large machines on the EC2 service from AWS, which means it's 4 CPU machine. We were absolutely sure that RAM was not going to be the problem here, because of the experience we had already with mobile apps, but the CPUs, yes, would be stressed, and we were expecting to grow the CPUs into an 8-CPU machine. We never really needed it. It was something that we also tried to contain, because customers usually have infrastructures like running on their VMWare's or even if they have their own cloud on Amazon, they usually don't like to go for 8 CPUs per machine because it stresses out the host, or in case of AWS, it spikes the costs.

So keeping it in a 4-CPU machine was the goal, and for that to be achieved, we did have to make some adjustments on settings of our platform. We did introduce, which was something very, very [inaudible 0:26:35.9] to see in this company the way we adjust, that we made a change to a supported mode in IIS and we actually recommend now to customers to change a

parameter that was for years and years it was defined like that in OutSystems, like carved in stone, and then we said, “Okay. Now, it’s better to change this.” We get performance. We had a lot of learned lessons through the way.

The infrastructures grew and there was something else that needs to be addressed. The JMeter tool, it’s incredibly stressful for the hardware when you do test like we did, because in every step of our test, we made sure that the answer that we were getting back was not just an HTTP 200 code which says, “Yeah, the service is up.” We actually made sure that it contains the message or part of the message that we know or we knew was the correct one.

For instance, if I’m retrieving the account balance for account number 300, I’m expecting to have $300 there, and we actually tested, we asserted that every request we made gave back the message, recorded that message on a JMeter orchestrator. It was an immense amount of data and the network traffic was absolutely out of this world, and we were expecting to have issues there. We did have issues with the machines running JMeter. Not so much the machines running the platform server from OutSystems, and we had to scale up those machines for JMeter. We ended up within this astronomical number of machines running JMeter injectors and 8 CPUs and 32 gigs of RAM per machine. We were really, really doing a stress test for the entire environment, and when we ended up we were able to say, “Okay. We can do this with 15 frontend servers only and an 8-CPU database. That was enough. Once we fine-tuned everything, that was enough, and we were able to say, “Even if I want to double this, I know exactly how many frontends and how much do I need to scale vertically in my database.”

So we were so sure of our math that from a certain point onwards, it was incredibly simple to scale up. We just made — It was a rule of three — I’m not sure if “rule of three” is the correct translation to English, but I know that if I have X-requests, I have Y-frontends. So if I make X + 2, I need X + 2 frontends. From a certain point onwards, it was incredibly simple to scale up.

To answer directly to your question. Yes, infrastructure had some issues. We needed to make changes to our configurations. We needed to increase — For instance, we needed to increase the number of connections that we were allowing to reach the database at the same time for the frontend, because the default number is 100th in the configuration for OutSystems, and in this test that number was too low. We needed to run a configuration tool again in double that

amount. Actually, I think it was triple that amount in the end, something like that. Yeah, it was a long path that's very fun one, and especially when we started seeing the results and — It was a great, great outcome.

[0:30:23.5] JM: If I heard you correctly, you said that there was more scalability bottlenecking of JMeter, which was recording and monitoring the traffic itself. Then there was more issues scaling the measurement device than there were issues scaling the infrastructure itself.

[0:30:45.5] PC: Yes. Yes, I know it sounds awkward, but it was exactly that. There's a simple explanation. We have an orchestrator for JMeter, and then we had slaves. Each slave was injecting an X-amount of request into the frontends. It was reaching our load balancer, which then did its magic. The thing is JMeter injector not just set the request to the machine, got the response back. It created a file, a huge file with every single possible data from the request. Not just the answer, but the latency, the response time, the HTTP status, everything, everything you can imagine, and then it's sent back to that data through the orchestrator and it waited for the orchestrator to say, "That's okay. We're good. You can delete this file and proceed to the next one." It was a lot going on on the JMeter side.

The network, we had to change the instance back from M4 to R-something, R4 I guess, on EC2 for the JMeter injectors, because that kind of instance has a larger bandwidth limit. We weren't reaching the bottleneck on the network side of things on JMeter. We have to make sure of that. On the platform server, things were much, much calmer.

[0:32:09.5] JM: Okay. That's funny. What were some of the takeaways from the load test?

[0:32:15.5] PC: The first thing is we changed — Like I said before, we changed the default IIS way of work in the pipeline, the way the pipeline works in IIS. We changed from classic to integrated mode. It was actually something that Microsoft was already recommending, but the performance gains on web and for lower loads were not enough for us to justify the effort to make that change and things were working mighty fine, so why change it?

But then this comes up and we understand, "Okay. There are performance gains to do with this." So we changed the integrated modes in IIS. We changed two integrated mode in IIS. We also

got several lessons that, like I mentioned before, ended up in the creation with the creation of a service for our customers. We are able to now provide a service for load testing to our customers. Yes, it's still in agreement with our customers. We do a kickoff meeting. We talk a lot about the methodology with them. We have the agreement that, "Okay. This is good for us. This is the outcome that we expect." There's a lot of negotiation going on on this test, but it was only possible and we know — It became possible to deliver this kind of service with confidence that we are bringing value to customers only because we were able to create such a stressful test here. We covered so many bases that we are not perfectly comfortable with any case that customers bring to us.

[0:33:56.8] JM: So now that you have this load testing system built, are you just using it regularly? Did you put it in the CICD systems? I mean, I assume if you put all these work into building a load testing system, you might as well reuse it on a regular basis.

[0:34:12.2] PC: Yes. Internally, yes. We do have some pipelines that is load tests for our builds, and we did create a webinar giving hints to our customers on how to integrate this in your development pipeline. If you're developing frames, if you're creating a new version every day, why not test it overnight and compare it with the previous version? You can use something open-source like Jenkins. It's free.

So we have APIs that are enough for our customers to build pipelines in their Jenkins systems. So yes, not only we use it internally. We recommend our customers to do it, and we have documentation for that. That's one of the big takeaways from this test too.

[0:35:03.9] JM: Very cool. I'd like to zoom out and talk a little bit about the rest of the stuff that you guys are building at OutSystems. OutSystems, full disclosure, is a sponsor of Software Engineering Daily, which is much appreciated, but part of that sponsorship is me really digging in to the company and understanding if this sponsorship was a good fit, and I found that it's a pretty interesting environment, because what you guys are building are these low-code — You're building a low-code application platform. Basically, what it is, is if there are people within a company who are — They're reasonably technical, but they're not software engineers. You give them this robust set of tools to build applications. So it's sort of like — When I was much younger, I used to play with these games that you could make movies with, for example. You

don't need to know much about animation, but it makes it very easy for you to make movies, and I always found it very fun and empowering and I think that's kind of what you guys do for mobile applications. You make it really easy for people who are not well-seasoned mobile developers to build these low-code applications, which is actually really important, because the amount of CRUD apps that could be useful to a large organization vastly outnumbers the number of software engineers at a company. It makes sense to have all these domain-specific mobile applications, for example, warehouse workers, or data entry people or — You name it. So maybe you could talk a little bit about why is there a growing need for these low-code applications and what are the typical types of customers that come to you?

[0:37:02.5] RC: Okay. So I actually really like your analogy in terms of the movie processing software and the fact that they have evolved to allow anyone with a computer to be a movie producer and to actually publish stuff on YouTube. That's essentially what we expect to do with software also. Basically, a local platform provides people with a way to design and develop a software very fast and with minimal hand-coding.

You don't have as many skills required as you do, for instance, if you want to develop something in .NET or Java. So that's a big part of it. That's what OutSystem software, and the way we actually do it is by providing tools to do this. What we do is we provide developers with a development environment, and this is a simple environment where you have a visual IDE, things works through drag and drop. You have a what you see is what you get editor for your mobile applications, for your web applications. Everything you do in terms of business logic is visual. It's something like visually you have a workflow where you define exactly what happens, and even building the database model, for instance, is all done through a drag and drop and a few clicks.

This is how you design a model and this is something that you can put in the hands of someone that knows about Excel, has some knowledge of databases, but it's not quite a proficient developer, and we actually had customers that did this. They had people there working on the business that weren't developers. They took a bit of training with OutSystems. They got to know the platform and now they're building applications that integrate with IoT devices and all of that. This is really empowering people to build applications.

[0:38:52.7] JM: Who at a company is responsible for building these? You mentioned somebody that might just be proficient in Excel, and there are a lot of these types of people. I went to school with a lot of people who they went to maybe a business school or they studied accounting and they found that maybe the conventional paths for them were not super interesting and they wound up at a tech company, and the only "code" that they knew was sort of Excel from maybe an investing banking job, but they got really good at Excel and that was enough for them to become a product manager or a project manager, somebody like that at a tech company, but you're saying that this would enable them to also build applications. Is that the typical type of user? Are we talking about somebody who's even less technically savvy than that?

[0:39:42.7] RC: Usually, yeah. It's that type of user. What we found is that if people have some kind of pleasure in building applications, if they're comfortable with computers, then they are able to deliver these applications very quickly. So these are business users or sometimes they're also called citizen developers, and these are users that have a business idea in mind, and for some reason IT doesn't have enough time to implement their project. So this is something they want to try out, and it's very easy for them to take the platform and do a first version of the application and have it running and have it published in production and so on.

We've seen this happening in some of our customers. Usually, these are applications that tend to have a very specific need of the business, and so they require more business knowledge than actually IT knowledge. So that's what we provide, a way for business to be translated very quickly into software.

[0:40:42.1] JM: Yeah.

[0:40:42.9] RC: Usually what happens is after you have this first version of the application, if in fact the application is successful and it proves to be valuable to the business, if it starts making the business money with this application, usually then the application moves to IT and IT continues to develop the application. So at this point, it enters the whole architecture of the factory and so then IT has a word to say. But the first versions, the first prototypes can be done by citizen developers very quickly.

[0:41:16.7] JM: Yeah. This is kind of a nascent category, this low-code application category, but I've seen this be really relevant at a lot of different organizations that I've talked to, like we talked about the mobile banking application. Banks actually woke up one day and found that they were technology companies and they were extremely understaffed in terms of engineering when they had that wakeup call and that's happened to other enterprise verticals that don't have the money to pivot to becoming engineering. For example, a school — A university, for example. Like I think about my university and the lack — I went to University of Texas, which is a well-funded school and they've got plenty of money. Even so, it felt like all of the — Not all of the — I shouldn't be so harsh, but a lot of the application infrastructure was just outdated, and the students would have been much better served if they would have had access to a wider variety of domain-specific applications that had been built specifically for students to be productive within the university.

With that in mind, can you talk more about some of the enterprises or the verticals or the types of applications that you see people building and maybe how the organization start to build them? Do they train people within these organizations to build low-code applications?

[0:42:48.7] RC: Yeah, definitely. The example you gave of the university is very interesting and it highlights one of the problems these local platforms try to fix. So there is definitely a skill shortage, the amount of job offers for developers is huge and it's very hard to find good developers, while at the same time everything is becoming digital. This need becomes ever more important. Not only that, but there is also — Let's say people are more demanding and they want things now. So they want it really fast. All of the things have made this perfect storm, which turned out in what we call now local. So this is becoming a known term in the industry, even analysts are talking about local platforms and so on. Yeah, it's becoming ever more famous.

As you mentioned, it's still not completely mainstream. We're almost there, but it's not completely mainstream, and so we do offer training for people to start using the platform. Of course, there is this technical training where we explain what the platform is, how it works. We show you an example application, so very quickly you can start building your own application, your own web or mobile application without systems. So this is a very fast process.

Not only that. We also teach companies and enterprises how to use this technology in a real scenario in the sense that it's not only about technology. It's the way you work, that needs to change too, because suddenly you are much productive, you are much faster delivering. This means you need to have much better contact with the business. Make sure you have the backlog, the features you want to implement, prepared to do it. Take advantage of the speed, not only to deliver new applications, which is important, but also very quickly adapt to feedback from the customer. That's another advantage of local platforms. It makes it really easy to put something in production, have your users test it, gather the feedback and then, in a matter of a week, have all the changes that the user has requested.

These types of things actually changes the dynamic of organizations and this is something that we also teach and we are about to release a lot of content on this area just to explain our customers, people that want to adopt low-code, how to best take advantage of this great new capabilities.

[0:45:15.7] JM: Okay. So one example I think of, if I'm talking about my university. University of Texas has a sprawling number of like internal budget line items. You've got a cafeteria in all the different dormitories and you've got the gymnasium. You've got multiple gymnasiums actually on campus. You could imagine all of the different managers of these smaller organizations within the university having their own budgets, and they've got — At the gymnasium, they're purchasing basketballs. At the cafeteria, they're purchasing food and stuff like that, and maybe there are times when they need to enter those line items on their phone or maybe they want to have — Maybe they're just entering those line items on the computer, but they need a dashboard so that they can take out their phone at any given time and open up an app and look at how much they are spending. If I'm managing the cafeteria at Dobie, which is one of the dormitories at the university interacts with, then maybe I need to look at that on a regular basis and make sure I'm not overspending some budget that I'm getting.

That's just an example of a simple application that you would probably never want to give a software engineer who is getting paid $100,000 a year, the responsibility to build, because the upside is not high enough, but you could allocate that task to somebody who is less technically savvy, has a lower salary to build that type of application. I think the experience would be you look at this platform, like something like an OutSystems, and it's kind of like an IDE, but it's a

drag and drop IDE and you can make mobile applications with this drag and drop systems. You can drag and drop things that connect databases to each other on the backend, but if I'm the low-code developer, I don't need to know anything about databases. I'm just looking at this visual wiziwig IDE and in the background it's setting up the right bases, it's setting up the right load balancers, it's setting up the right network connections and I as the "low-code developer" don't need to know about any of that stuff. Could you talk a little bit about what is going on on the backend? If I'm the low-code developer and I'm setting up stuff in this wiziwig, what's going on in the backend?

[0:47:47.0] RC: Okay. So just to expand a little bit on the example you just gave, because I think it's a really great example. So it's even more important to consider a local solution, because of the way usually those things work. So in the example of the departments in a university, what happens is if you want to implement such a system, you hire a student, which is going to do — I don't know, his final year or his final course and he'll deliver an application. He'll [inaudible 0:48:15.1] something, let's say Python, and then what happens is you have an application running in Python somewhere in the server, build all the desks hidden somewhere in the department, and one day that application fails, or one day you come to the conclusion that because of some regulation that change, you need to adapt it somehow. All of a sudden, first, you need to find the application, and then the student is no longer in the university. He moved on and he's now in Silicon Valley and you have nobody that has any idea how to work or how to change the application.

Actually, that would be a great scenario for a local platform, because what you can do in this scenario [inaudible 0:48:56.5] because usually universities have a central IT. So you can have OutSystems or other local platforms as the way to deliver applications. From that moment onward, even if what you do is hire a student, delivery an application, he'll do it in OutSystems, and because it's a local platform, it will also make it much easier to share knowledge. Knowledge transfer is also another very important part of local platforms, because since everything is so visual and because we rely on a certain number of patterns and so on to deliver applications, it's very easy for a person to just get in the project, take a look at the code, understand what's going on and adapt is as needed.

So we have customers with extremely complex applications in the insurance area that can actually onboard developers in two weeks, and this includes the OutSystems training and learning the details about the application. Actually, the example you gave would be a great scenario for the use of a local platform.

[0:50:01.0] JM: Insurance is another good example, because like insurance — You got these companies where the amount of domain-specific knowledge you need to be a technical person at an insurance company is tremendous and you are learning technical knowledge about insurance, because you don't know anything about coding. You don't have time to learn that stuff. So I think it makes a whole lot of sense.

One thing that makes me a little optimistic is I was coding iPhone applications right when iPhone came out. I worked at a development shop for a while and I remember we were looking at — I think this is right when the iPhone came out. Maybe a little bit after, but I remember we were looking at these cross-platform things, like Appcelerator and these other ways where you could write an application once and it would spin it up on both android and iOS, and at the time those were horrible to work with. They were impossible to work — I mean, no offense to Appcelerator. I think it was just very early days. Appcelerator might be much better today, but I'm sure there were other things back then that were decent.

But my understanding is that today it's actually gotten much, much better where there are actually numerous cross-platform development environments where you develop something and there's enough of a cross-platform development experience that you can get it spun-up consistently on iOS and android.

Can you talk about what you do to enable that? Because if people are just building in this wiziwig and it gives them a mobile application in both platforms, what's going on under the hood?

[0:51:42.3] RC: Okay. So, yeah we definitely — We support building applications for android and iOS, and let's go under the hood, and what's going on is that we are actually using Cordova to build the applications.

So what we are actually doing is when you define the UI on an application, on a mobile application in OutSystems, that UI is converted into a React application that runs inside Cordova on the device. That's the way we support this both iOS and android.

It's interesting because we do that, but actually many times you want to develop only once, but you want to have slightly different looks because of the way people are used to working when using iOS versus what people are used to working when using android. So we actually managed to implement that. We have special themes that take care of this. So you can build a single application, and when you are writing it on iOS, it looks like an iOS application with the styles for the buttons, for the wings and all of that, but when you're running it on Android, it's going to have the Android look and we changed the way the menus look and the way the buttons look and so on.

The subtle differences that you expect to exist within the applications, but again, we don't want developers to spend time on that, so we automate all of that and make that one click away. That's the idea.

[0:53:13.9] JM: The backend? Are the backends similar enough from app to app that you can just have something that's standardized?

[0:53:22.8] RC: What we do for the backend, in the case of mobile applications, the backend essentially is serving REST APIs. The communication between the React application, the native application and the backend is all done through REST, and we make sure those goals are secure and they provide the right levels of authentication and all of that, but in the end it's basically — We're serving REST.

Backend then connects to a database. So we support either SQL server database or Oracle database, and of course then you can integrate both on the frontend or on the backend with whatever system is necessary, because we know that enterprise applications do not live in isolation, so we make it as simple as possible to integrate with other systems such as [inaudible 0:54:11.4], Salesforce, that type of stuff.

[0:54:15.1] JM: All right. Well, I guess to wrap up, what are you guys working on today? What's the future for your product?

[0:54:21.4] RC: Okay. So for our next version right now, we are working on three main things. It's going to be around deploying to containers. So we want to be able to take this application step. Right now we are deploying into a central server with IaaS and stuff like that and we want to be able to deploy those to containers.

We are going to work in micro-services. Here, what we are witnessing is that as our customers grow their installations, because we have some customers that are using OutSystems for a really, really long time and have very complex projects, the architecture of those applications is becoming ever more complex.

The traditional way to solve this is using micro-services. Micro-services have a series of disadvantages. Particularly, the fact that the connection between the pieces is very soft, so there is no validation if things are correct, if what you're calling is still there and so on. So we are going to fix that problem. We are going to release micro-services, but with the ability to do this impact analysis and to check for consistency before you actually deploy the application.

The final thing we're working on is — So frontend development, it's still something that's very hard to do. You need a specialized person with lots of knowledge of CSS, with lots of knowledge of JavaScript and so on. Now, we made it easier in OutSystems and we actually have courses on our website that teach how to do frontend development in OutSystems, but it's not yet at the level that someone with little experience in computer science can do it. So the example of the persons we were talking about earlier on that have experience in Excel and so on, it's still hard for them to delivery that edgy UX that's really important when you are building a business to consumer application. So that's another thing we're working on, and we plan to deliver that with the next version of the platform.

[0:56:23.1] JM: Okay, guys. Thank you for the explanation of load testing and the discussion of your platform. It's been great having you on Software Engineering Daily.

[0:56:31.0] PC: Thank you very much.

[0:56:31.6] RC: Thank you very much.

[END]