EPISODE 658

[INTRODUCTION]

[0:00:00.3] JM: The main user interfaces today are the smartphone, the laptop and the desktop computer. Some people today interact with voice interfaces, augmented reality, virtual reality and automotive computer screens like the Tesla. In the future, these other interfaces will become more common. Developers will want to be able to expose their applications to these new interfaces.

For example, let's say I'm a developer who builds a podcast playing application, I have a website in a mobile app, but what if I want to expose that app to a voice interface, or what if I want to expose a specific piece of functionality from that app to make shortcuts easier? Android Slices are user interface components to that expose pieces of application functionality to Google search, Google assistant and other applications in Android.

Jason Monk is a software engineer who works on Android Slices at Google and he joins the show to discuss how mobile user interfaces are changing. He talks about the motivation behind Android Slices and the engineering behind this newer building block for Android developers. We haven't done too many shows about Android-related subjects, but I hope we do more in the future.

[SPONSOR MESSAGE]

[0:01:23.9] JM2: DigitalOcean is a reliable, easy-to-use cloud provider. I've used DigitalOcean for years, whenever I want to get an application off the ground quickly. I've always loved the focus on user experience, the great documentation and the simple user interface. More and more, people are finding out about DigitalOcean and realizing that DigitalOcean is perfect for their application workloads.

This year, DigitalOcean is making that even easier with new node types. A $15 flexible droplet that can mix and match different configurations of CPU and RAM to get the perfect amount of

resources for your application. There are also CPU-optimized droplets, perfect for highly active frontend servers, or CICD workloads.

Running on the cloud can get expensive, which is why DigitalOcean makes it easy to choose the right size instance. The prices on standard instances have gone down too. You can check out all their new deals by going to do.co/sedaily. As a bonus to our listeners, you will get a $100 in credit to use over 60 days. That's a lot of money to experiment with.

You can make a $100 go pretty far on DigitalOcean. You can use the credit for hosting, or infrastructure and that includes load balancers, object storage, DigitalOcean spaces is a great new product that provides object storage, and of course computation. Get your free $100 credit at do.co/sedaily.

Thanks to DigitalOcean for being a sponsor. The co-founder of DigitalOcean Moisey Uretsky was one of the first people I interviewed and his interview was really inspirational for me, so I've always thought of DigitalOcean as a pretty inspirational company. Thank you, DigitalOcean.

[INTERVIEW]

[0:03:30.6] JM: Jason Monk, you are a software engineer at Google. Welcome to Software Engineering Daily.

[0:03:34.3] JM2: Thanks, Jeff. Happy to be here. You work on Android and one of the features that we're going to discuss today is Android Slices. More generally, I think Android Slices represents machine learning and search features becoming more deeply integrated with the Android operating system. Describe how the experience of the mobile user is changing, thanks to machine learning and search being more deeply integrated into the operating system.

[0:04:01.6] JM2: Okay, that's interesting. I would characterize it slightly differently. I think that enabling machine learning and search more deeply in the operating system is definitely a goal, but at a higher level, Slices really what we're trying to do is we're trying to break the monolithic app model, which is on your phone right now. You normally open an app, do some stuff, interact with that app and then when you want to do something else, you open a new app. Slices is

trying to break that up and say like, “Look, let's try to bring more of apps outside of themselves and into the OS, so that users don't have to do as much jumping around.”

[0:04:34.8] JM: In that perspective, that seems like a real paradigm shift even from if we're talking about desktop applications. If I'm using a browser, I'm using the browser. If I'm using Slack, I'm using Slack. I guess to some degree, an analog might be, you get – on Chrome, you get these browser notifications sometimes and even if you're on a screen where a browser is not open, the Chrome browser notification might still appear on your screen. In that sense, the monolithic browser application is broken up. Do you have any other analogues for how applications are broken up on other platforms, things other than mobile?

[0:05:15.7] JM2: Ooh, things other than mobile; anything widgety jumps to mind. I'll be honest, I'm not a huge expert on other systems. I'm a pretty big Linux user and a lot of them have – in GNOME Shell, there is in the top right-hand corner, there's a task tray where you can add various things there, and there's a way to get ongoing apps and bottom left. Anything ambient indications, I think are similar models. Since I don't have many examples non-mobile, I'll give more mobile examples.

[0:05:47.3] JM: Okay, sure. You can go ahead and jump into that, the mobile examples of Slices.

[0:05:51.1] JM2: Because there's definitely – well, and not just Slices. The idea of having UI outside an app has existed for a long time. We've had notifications on Android for forever, and we've had remote views on Android for forever. That's how widgets and custom notifications work. The new attempt with Slices is to try to come up with something which is generalizable enough, but still feels integrated, and that's the trade-off between the notification space and the widget space existing, and that we’re like trying to think about when we were tackling Slices as a new thing.

[0:06:24.9] JM: Can you give a few examples?

[0:06:26.1] JM2: Yeah. When you're in launcher for example, if you have a widget and like a weather widget or something like that on your home screen, a lot of people have these, you'll

notice that the widget doesn't always necessarily feel very connected to the launcher. It's very just like something which sits there. Well it can have rich content and lots of custom layout and feel really branded, there's no good ways for it to interact, because basically to the home app, to the launcher, it's just a black box. It's like, “Hey, there is some remote view here.”
[0:06:54.2] JM: Launcher by the way, that's the desktop, or the home screen?

[0:06:57.8] JM2: Yeah, yeah, yeah. On Android, you hit the center button on the bottom of your screen and you end up on launcher, or home, or whatever anyone's current name for it is. Yeah, and so on the other extreme end of the spectrum is if you look at notifications, not the custom ones, if you look at notification templates, in the past few releases in Android, we've had some really great integrations there. What we call it is Dice and I'm not sure what it's called publicly, but the thing where the notification icons, they follow from the status bar as you drag down your notification shade and you have that visual continuity, and there's all sorts of animations between states. That's the deep integration with the surface we can do when we have information about data inside. There's this trade-off that you pay between how much the app has control, versus how much they can integrate with the surface.

[0:07:47.6] JM: A few examples I've seen are, for talking about Slices explicitly, you can have these actions that are decoupled from the actual application itself. For example, summon a ride with Lyft, or play a song with Spotify, these application-specific ways of interacting with content, with apps, can you give a few examples for how those are accessed through Slices and what is new about that?

[0:08:16.0] JM2: Yeah. Clarify nomenclature, because actions is a very overloaded word. When we were telling the Slices story at IO and Sense, we've talked a lot about them tied in with App Actions at the same time, and I think that's what you're referencing here. There's two concepts in there; there's like, “Oh, I just want to take an action in app,” and then there's App Actions, which is a thing, you know what I mean?

[0:08:34.6] JM: Right, absolutely. I should have been more correct with my language.

[0:08:37.7] JM2: No, no, no, no. That's totally fine. I wanted to identify that and that's a point that we even see internally we get confused a lot of the time talking about it. Yeah. The way App

Actions works is Google defines a set of actions that it thinks a user might want to take. Yeah, and those are things like book a ride, or play a song, or whatever. The real meat of what App Actions does is it also defines a schema for input parameters. In Android, there are lots of built-in intents, which are open the web browser, or whatever.

This is really a parameterized way to say, “Hey, I want a location parameter here, or I want an artist here.” You can say, “I want to play music with an artist,” and it can go and look up and say like, “Oh, I know what apps actually know how to play music and will receive an artist.” The way that ties into Slices, and so this is a very interesting model, because it's really easy to think of these things as one thing, because they're both trying to bring out parts of an app closer to the user, but the model I like to think of it as is App Actions is a deep link discovery mechanism.

You think of like you're building a way to link way into an app; you are going to go straight to an artist of Taylor Swift, or whatever, you're going to go straight to booking a ride in Uber, or Lyft and something like that. Slices is the display mechanism for that. The way they link together is through intents, intents of the common language of Android of how everything works. Once you have an action built, that basically becomes a URL/an intent deep link into your app. We have a translation mechanism in Slices where you can link basically any intent, which is going towards your app to the concept of a previewable, or an extracted state of your app, where you can see that in a different surface like Google search.

[0:10:25.8] JM: If we're talking about those three vocabulary words; intent, action and slice, if we were talking about for example, the slice, the high-level interpretation of play a song by Taylor Swift on Spotify, what would be the intent and the action that might correspond to that slice?

[0:10:46.2] JM2: Yeah. The action would be something like play a song and the parameter would be the artists name, or artist ID, or something like that. The way App Actions works is it tells – is the app gets to define how those parameters transform into an intent, so the app gets a control there, so it's really hard for me to define it, but they could easily just say this is package name/play a song/artist ID. That could just be their scheme of how they received them in their app.

[0:11:17.6] JM: Was I completely mistaken around the search and machine learning features being connected to Slices, because my understanding was in the ideal world, so at some point in the future, we should be able to control so much more of our Android device through the assistant, or just through entering the voice assistant, or just entering a simple search query command and you should be able to have these deep links exposed intelligently, or indexed in a way that a voice command can correspond to them. Is Slices connected to that kind of conversation, the ability to trigger certain behaviors of your phone through the assistant, or through some just single-text NLP search protocol?

[0:12:02.6] JM2: Yeah. No, they absolutely are connected. I'm sorry, because I started off by walking away from that. No, it's just because I wanted to start by proposing Slices as more of a foundational technology of the way we extract that information. I think that App Actions and Slices absolutely do service those building blocks of how do we even present that. If you were to say, a year ago before we have these, okay, we want to start building more intelligent things and we want to start proposing stuff to the user. We didn't even know how to get ahold of that, or let alone show it to them, other than just guessing deep links, like crawling the web and figuring out what URL schemes various apps have. This gives us a way to address those basically, to say if we know what the user wants, now we can construct that link into an app and even present that data sooner through Slice.

[0:12:56.9] JM: Yeah, tell me more about the design decisions that led to this model of where you have actions, intense – or intents, actions and slices, because I can imagine that three years ago, this was total greenfield and you knew that voice was going to become a big deal, you knew that the interfaces that people are going to be using are going to change over time. How did you come to the conclusion that this model for the Slices being the path to the deep linking would be the way to do things?

[0:13:29.1] JM2: Yeah. That's interesting. I can tell a little bit of backstory on Slices. I wasn't involved in the early days of actions. It's interesting, these two projects came up organically in Google just all on their own, and then we got paired together, people being like, “Hey, you should really talk to this other team. They really seem related.” We're like, “Yeah. Okay, we need to tell some unified story here, because they make sense together.”

Yeah, in terms of Slices it's interesting. There's a future-looking Android UX group within Google, which is where Slices were born. Slices were very different from where they started to where they end, and I think that's true for a lot of projects. I think in the beginning, there was a lot more about the transformative properties of a slice, which was not only about bringing stuff out of an in-app, but we were thinking about moving things between surfaces, which when you think about is about user control. Like if the user has a piece of content and they say, “Oh, I don't want to interact with it on this surface. I want to drag it over to my home screen and interact with it here.”

[0:14:26.3] JM: Or with my Chromecast.

[0:14:27.6] JM2: Yeah. Or even other devices. That's where it started and I mean, we still continue to strive for that. We like them to be universal and movable between surfaces and between devices someday. If you look at the internals of Slices, you'll see some of that structure and how that comes out. At the top level, Slices is very much just a builder API. You're like, “Hey, I want to make a list of things; here are my titles, here are my summaries.”

If you dig into that one level deeper, you'll find that they sit on top of a tree structure of nodes, which are full of the base level elements of content of text and icons and pending intents for actually triggering actions and a set of presentational hints. That layer of abstraction underneath them is meant to make them more flexible and moveable.

[SPONSOR MESSAGE]

[0:15:28.5] JM: Leap.ai matches you with high-quality career opportunities. You are more than just your skills and a job description and a resume; these things can't fully capture who you are. Lea.ai looks beyond these details to attempt to match you with just the right opportunities. You can see it for yourself at leap.ai/sedaily.

Searching for a job is frustrating and Leap tries to reduce the job search from an endless amount of hours, days, weeks, to as little as 30 seconds trying to get you matched to a job instantly, by signing up based on your interests, your skills and your passions. Leap works with

top companies in the Bay Area, Unicorns and Baby Unicorns; just to name a few; Zoom, Uber, pony.ai, Cloudera, Malwarebytes and Evernote.

With Leap, you are guaranteed high response and interview rates. You can save time by getting direct referrals and guaranteed interviews through Leap.ai. Get matched to jobs based on your interests, your skills and your passions instantly when you sign up. Try it today by going to leap.ai/sedaily. You would also support Software Engineering Daily while looking for a job. Go to leap.ai/sedaily. Thank you to Leap.ai.
[INTERVIEW CONTINUED]

[0:17:07.4] JM: Let's go back to that example of the playing a song. If I'm searching in my Google search, so on Android for people who are not Android users, at least on the default for the pixel, which is the phone that I have, there's a search bar at the bottom of the home screen and you can just enter in stuff into the search bar. If I enter in play a song by and my phone knows that perhaps I like Taylor Swift, who doesn't? I have Spotify on my phone and I'm logged in on Spotify and the phone is able to recognize that if I just preface things with play a song, it can recommend play a song by Taylor Swift and it has a deep linking thing, and I can just click on that and it'll go directly into Spotify and it'll open up Spotify with that song playing.

Now in order to present that search result in the autocomplete suggestions, the search application needs to have some permissions into Spotify, and Spotify needs to have some things in memory presumably, so that it can present some suggestions to me that are tailored based on my preferences. How does abstraction in that permissioning in the connection of the data model between the search application and Spotify itself how is that modelled?

[0:18:23.6] JM2: Yeah, that's a really interesting space, because it's so privacy-centric. You could easily be typing things into your search box that you don't want going to Spotify, right? There have been lots of grounds with this, because of the way it works is there's this API called Firebase app indexing, that's a pre-existing API. What that does is it lets apps push content at Google saying, “Here, I've got some personal content on my device that I know is about my user and I'm going to send you some keywords and information about that content and usage data, so that you can surface it and deep link in to me.” That's been existing, but not shown as prominently until recently.

What we're doing is we're building on top of that API, and so now you can actually link together what was previously a Firebase app indexing entity to now a slice. They can use all the pre-existing cues to say like, “Oh, this is clearly trying to play music. This should go to Spotify, and I know it's this entity in our database.” Now they can say, “Oh, hey. This is slice that I want to show right now.”

[0:19:31.3] JM: I see. Spotify is presenting some information to something in the cloud, the Firebase indexing service, and then your search application is also communicating with the same data storage system in the cloud. It feels like it's all happening on your phone, but it's actually taking place in the cloud.

[0:19:51.6] JM2: No, that's actually on your device. Those Firebase APIs interact with –

[0:19:54.5] JM: Oh, I see.

[0:19:56.4] JM2: Google Play Services on your phone and store that all locally. Now that's just the base case. Let me take that and make it a little bit more complicated. Even if we have that all set up and we say, “Oh, okay. We want a deep link into some app. We want to show their slice,” the default behavior of a slice is when you show it the app would know, because they need to present that data and would need to be live, but that's really dangerous with the security implications of the search box.

What we do instead is we cache that data for things that are general term queries. That means if you type in the name of an app like YouTube, you will get the default app slice and we consider that non-sensitive data. Yeah, okay, YouTube can know you typed in YouTube into the search box. If you type in something random, like an artist name, in that case when you're pushing keywords and entity data at the Firebase app indexing API, it will also grab a snapshot of that slice and it will hold that in its database, so that it can show a cached version of it whenever the user is querying.

Once you interact with it, or give us a signal by touching somewhere saying, “Hey, I actually want this,” then we'll wake up the app and make sure that all of the data is live. Until then, we won't give them any signal that it's there.

[0:21:11.3] JM: Okay. Can you go through that example again with – so you went through it with YouTube there, go through it again with Spotify. For how Spotify is fit – Yeah, good for Taylor Swift or for example.

[0:21:20.9] JM2: Okay. Spotify, if you type in, say Spotify.

[0:21:24.3] JM: Into Google search?

[0:21:25.4] JM2: Yeah. If you type in Spotify into Google search, will immediately wake up the Spotify app and say, “Hey, what's your primary slice? What's your main slice?” They can say, “Oh, here's our discover weekly playlist,” show that as our default slice. If you type in Taylor Swift, ahead of time Spotify will say, “Oh, I know that Jeff likes Taylor Swift. I'm going to send that to the Google index and I'm going to say I also have a slice at /artist/Taylor Swift that you can show for this.”

Google Play Services ahead of time will go grab that slice and say, “Hey, what do you actually have to show me? It'll get all the content from Spotify and serialize it into its database.” Now, you type in Taylor Swift into the search box and it knows that it wants to show something from Spotify, but rather than fetching it from Spotify, it fetches it from the database cache that it has, and you see Shake It Off. Now you say, you want to listen to that song, you click on it, at that point we know there's an intent, we know that you're actually looking for Spotify and you're not just typing in random words into the search bar, and then we connect you with Spotify app.

[0:22:28.1] JM: To some degree, Spotify needs to manage this collection of suggested slices, right? They need to be able to either build some internal recommendation system, or define just some statically configured recommendations.

[0:22:45.0] JM2: Yeah. Well, that's where the link between App Actions and Slices comes in. In the model that we just talked through, that is all without App Actions. Now some of this is very

greenfield and hypothetical, but you can imagine a scenario hypothetical, because we have to launch it and figure out how that even works. If you have App Actions and Slices implemented in Spotify, you can imagine a world where you type in Taylor Swift and Google recognizes that Taylor Swift is an entity. It doesn't do any lookup from what Spotify has. It instead says, “Oh, Taylor Swift is an entity, I know that they care about this.”

Then it looks up an action related to that, so it’s, “Oh, it could play songs from Taylor Swift.” It says, “Oh, hey. Spotify has an action for playing songs for Taylor Swift. Let me translate that into an intent.” That's the first stage of that's all actions doing all of its magic. Then you can take that a step beyond, which is if you have all the information and you have an intent, which is a deep link you can say, “Hey, is there a slice for this?” Rather than just showing an icon in the words ‘Taylor Swift,’ you can say, “Hey, please give me a slice for this, which is a playlist containing a bunch of songs.”

[0:23:57.9] JM: In the current model, it is this necessity that in the app on your phone somewhere, Spotify has declared here is some default slices and then maybe they have some other batch job that is generating Jeff's recommended slices?

[0:24:17.0] JM2: Yeah. That would be the visceral implementation of Spotify adding support for Slices.

[0:24:22.4] JM: Okay. What's to prevent – I mean, could they just bloat that slice file so much, that every time I'm searching anything in Google, they're recommending some podcast, or audio file to me, or is there some limitation on the number of slices that can be recommended?

[0:24:42.0] JM2: Yeah. There are some limitations. Luckily, we get to sit on top of the app indexing architecture for this stuff. The firebase app indexing team has done lots of really great work in the past, so we don't have to think about this as much, which is really nice. Yeah, there is a limit on the number of I think they're called documents I think is what they're called. There’s a limit on the number of documents you can push at it. Then on top of that, there's a limit on the number of keywords that you sit on top of that.

Also at the core of it sits the magic Google proprietary ranking, obviously. I know it's really hard to rank on device content, but it's something they're working on and getting way better at. To do

that, we actually take into account usage signals. Some of this is self-reported, but it's relatively easy to determine if an app is just pushing at you random usage data, rather than actual usage data.

Spotify, every time you go in and listen to Taylor Swift, it can notify Google Play services and say, “Hey, Jeff's listening to Taylor Swift again.” That will give Google a little bit of information saying, “Hey, we should bump those ranking up a little bit more.” Even if all apps register on the same keyword ‘Taylor Swift,’ it will only show the ones that it has a reason to believe that you want.

[0:25:56.7] JM: What's so interesting about this is that we went for many years with – the other common phrase you would hear is dumb device smart cloud, and here it's gradually becoming the opposite, if you imagine this Firebase app indexing service that has to index the ranking for all the apps and their intents, all the apps and their slices, or their actions on your phone, if you have Spotify and Uber and Postmates and Instacart and Amazon and all of them have their slices and you enter in ‘shake’ on your search bar.

Your phone to some degree, or there's some interaction between your phone in the cloud where they're determining should they give you a on-device recommendation from one of the slices, like either it's Shake It Off from Spotify, or order a milkshake from Postmates, or get directions to Shake Shack from Google Maps. Is there any trade-off between how much work you want to delegate to that on-device indexing service, versus the traditional Google search in the cloud thick indexing service?

[0:27:16.6] JM2: I think there's a huge trade-off there. I will say, this is speculation because I'm on the Android team, I'm not on the Google search team, but waivers landed. Yeah, so I think the trade-off is I think because the cloud is so much more powerful, you want to put as much content there as possible, as much of the ranking, as much of the decisions there, because not only do you have the power, but you also have the information from other people to feed that. Like click-through rates of other people and how that happened and what they actually wanted, versus what you actually want.

The key part about what's on your devices, your device is personalized, your device is yours, and so the Google Cloud servers may or may not, depending on account linking and whatever and logins, know what songs you actually listen to on Spotify. Whereas, on your device it knows absolutely, and it's really easy for it to know that.

[0:28:09.1] JM: Tell me more about how you expect people to change their designs in their UX decision-making process for applications where they are using Slices.

[0:28:20.9] JM2: I think that within their app, there would be very little change in UX, I think for the most part. Assuming currently they have a well-structured app, which has some amount of deep links. When there is clear user intent, there is somewhere in their app that you wanted to go. You assume that you wanted to be looking at the Taylor Swift playlist in Spotify, and assuming that Spotify has a screen where you could be looking at that.

I think that there's very little impact there. I think what the difference is is the accelerators to get there. Also, there's a lot that goes into the UX of thinking about a slice itself, like what content you put there really is going to affect the user's experience. A lot of that is just put in the app's hands of like, they could – it could be that you're typing in Shake It Off and they decide to say, "I'm going to show you artists related to Taylor Swift." Might be a good decision depending on your music taste, but it also might be that Spotify knows that you're going to be playing Shake It Off and they should present that button to you right there.

[0:29:23.2] JM: What else can – if Spotify wants to change the Slices experience, how else are they configuring that? Do they have the ability to configure that they want to present you artists related to Taylor Swift? What's the API for Spotify to present a variety of Slices and not just one that's as naive as play a song by Taylor Swift?

[0:29:47.6] JM2: Yeah. Well, so we do some basic checking to make sure that your keywords are tied to your slice in some way, that you're not just spamming keywords. Beyond that, pretty much all of the content is in the app's control. The way the API looks it, the best words I've heard describing this is like a simplified layout language. We wanted the templates to be really flexible. When you're building them, it's like you're constructing a UI, but it's a very restricted UI. You can say at the top level, all Slices right now are list, so you say, "Hey, I've got a list," and

then you say, “Hey, I've got a header and the title is Taylor Swift, and the subtitle is other artists related to Taylor Swift or something like that.”

Then you can add individual rows which have – and rows can have all sorts of content they could have. They could have a slider, they could have switches, they could have titles and subtitles. We also have ways to present stuff horizontally. If you want to show stuff in a grid, like if you were doing a weather forecast, or something like that. It's like composing simplified layout.

[0:30:53.3] JM: Tell me more about what your role has been in the development of Slices. What's been your involvement and are you engineering the actual code for Slices, or are you more of an architect for what's going on?

[0:31:08.0] JM2: Yeah. That's really changed over time. It's been interesting. I am the engineering lead for Slices. I started off by being the architect encoder for a lot of it. There were two engineers when we started trying to take over this project and productionize it, and say like, “Okay, let's take this UX concept of Slices and say what if we want to make this a real thing?” From there, it's scaled a bit. I would say most of my time these days is more in an architectural role. I spend a lot less time coding. I don't code not at all, but I do code some. I spend a lot of time thinking about what is our structure right now, what is our – what would it look like if we pick up new surfaces and what effect does that have on our APIs?

I have a really great engineer, Maddie, who was talking with me at IO and she pretty much handles all of templates on their own, so I don't even have to think about that most of the time. Yeah, does that answer the question?

[0:32:02.5] JM: It does. When you say for new surfaces, what is that look like? What services are you talking about?

[0:32:08.8] JM2: I don't think I can really answer that. We try not to speculate in the future.

[0:32:13.0] JM: All right.

[0:32:13.8] JM2: No. I mean, I would like to see a lot of new surfaces.

[0:32:17.1] JM: We did some shows with Flutter recently. We did a couple Flutter shows, and I think the idea of Flutter is interesting, because it's supposed to be surface agnostic. It's more about designing UI that's super flexible for any surface. Are you thinking more abstractly about just how you make this UI component work for any surface? You want it to be compatible with anything?

[0:32:41.7] JM2: Yeah, we do. That's something we have to think about every time we add a new template. One thing I'm not scared to talk about is we have said that we're looking at messaging templates for the future. We don't have anything messaging specific right now, but we want it in the future. You can see in our code base if you just look at AOSP that it's there.

When we think about that, we have to stop and say not only what is the backwards compatibility story if someone's running an old version, or whatever, but also what does that look like on other surfaces in the future? An interesting case of this is you have a pixel phone, right?

[0:33:14.5] JM: I do.
[0:33:15.8] JM2: Okay. On your always-on display, you're seeing some stuff like weather and date and upcoming calendar events.

[0:33:23.4] JM: Yeah.

[0:33:24.6] JM2: Okay. Under the hood, that's actually powered by slice technology. We use Slices as the data transport layer for what that is, and at the UI layer we say like, the always-on displays is a very different surface than a normal slice view. I mean, it's got all of its own custom rendering. That's an example of when you are constructing a new template, you have to stop and think about what are the core building blocks of this data and what if we want to present it in a different way?

You'll notice that even in the APIs, if you look at things like, if you want to add a slider, it's called an input range and not a slider. Because we don't guarantee that it's going to be a slider. If a user is interacting with voice in the future, or if they're on a different device that doesn't have an

easy touchscreen, they might be hitting buttons, or talking at it. Who knows? The idea is that – the core data model there is you want an input range.

[0:34:12.5] JM: I don't understand. What's the difference between an input range and a slider? I guess, if you're talking at a voice interface, or like an augmented reality interface, I'm not necessarily sliding. I'm looking through my glasses, or my VR headset at an input range, and I may not have the physical sensation of sliding something. I'm just looking at this range thing?

[0:34:37.3] JM2: Yeah. I mean, you could imagine a world where you're interacting with the assistant and you say like, "Hey, what's my brightness?" You hear it's 45 and you say, "Hey, set my brightness to 75." That very much is not –

[0:34:51.2] JM: There's no slide.

[0:34:52.3] JM2: There's no slide, there's no even UI in that case.

[0:34:56.0] JM: Okay. Can you talk more about assistant and how this fits in with assistant? That sounds like a no.

[0:35:03.7] JM2: No. I can a little bit.

[0:35:04.9] JM: Okay.

[0:35:05.4] JM2: We've definitely talked to assistant. We think that they fit in in some way. I think the answer is we don't know how. I think the assistant has some very specific conversational APIs that are really, really good. It's really hard to think about like, do you want to generalize those, or do you want just to use them as is and let that be a specific thing? It's a trade-off between easy developer adoption and power and control.

[0:35:29.7] JM: What do you mean voice APIs that are really good?

[0:35:32.0] JM2: Actions on Google has – there's a whole API for way for developers to define grammar of how you interact with the assistant. I'm really not an expert here, I just like, I know of

it. It is possible for apps to do those kinds of integrations now. When you know that that's the landscape, you have to be like, “All right. Well, what would be our role if we wanted to be there?”

[0:35:54.3] JM: Oh, man. Can you tell me more about just how you navigate this developing for voice? I've talked to a few people who are working on voice stuff, both on the show and off the show. It sounds like it is so nascent. People really just don't know what they're doing today. Can you tell me more about just what that experience is like, because I imagine there must have been false starts and you don't want to over commit on any particular strategy, but it's hard to get significant testing without deploying something. Well, how do you develop voice?

[0:36:29.1] JM2: Yeah, I'm just going to skip that altogether. I have no idea. I talked to those engineers internally, like that happens, but I don't know. They certainly seem confident when we talk to them. They haven't figured out. You should have one of them on the show and then we'll find out.

[0:36:43.5] JM: Yeah, I absolutely should.

[SPONSOR MESSAGE]

[0:36:53.8] JM: Cloud computing can get expensive. If you're spending too much money on your cloud infrastructure, check out DoIT International. DoIT International helps startups optimize the cost of their workloads across Google Cloud and AWS, so that the startups can spend more time building their new software and less time reducing their cost.

DoIT International helps clients optimize their costs, and if your cloud bill is over $10,000 per month, you can get a free cost optimization assessment by going to doit-intl.com/sedaily. That's D-O-I-T-I-N-T-L.com/sedaily. This assessment will show you how you can save money on your cloud, and DoIT International is offering it to our listeners for free. They normally charge $5,000 for this assessment, but DoIT International is offering it free to listeners of the show with more than $10,000 in monthly spend.

If you don't know whether or not you're spending $10,000 if your company is that big, there's a good chance you're spending $10,000, so maybe go ask somebody else in the finance department.

DoIT International is a company that's made up of experts in cloud engineering and optimization. They can help you run your infrastructure more efficiently by helping you use commitments, spot instances, right sizing and unique purchasing techniques. This to me sounds extremely domain-specific, so it makes sense to me from that perspective to hire a team of people who can help you figure out how to implement these techniques. DoIT International can help you write more efficient code, they can help you build more efficient infrastructure. They also have their own custom software that they've written, which is a complete cost optimization platform for Google Cloud, and that's available at reoptimize.io is a free service, if you want to check out what DoIT International is capable of building.

DoIT International are experts in cloud cost optimization. If you're spending more than $10,000, you can get a free assessment by going to doit-intl.com/sedaily and see how much money you can save on your cloud deployment.

[INTERVIEW CONTINUED]

[0:39:18.1] JM: We see that you have to have a lot of interactions with disparate teams. You've got interactions with voice teams, with search teams, it sounds like with the Firebase team. How do you juggle all those different conversations and keep the interests of the different teams aligned? Because I can imagine this could lead to sprawl, or gaps in communication. What's been your, I don't know, your experience in inter-team communication?

[0:39:47.0] JM2: Yeah. You ask how I manage that, the answer is poorly. Yeah, it is really hard. We really regularly between – for the core part of Slices showing up in search, there's really been three core teams. I mean, you think about that there and that's us, the Slices team, Firebase app indexing like you talked about, and the search team and well, and specifically the sub-team of search that works on those really upfront suggestions that are crazy low-latency.

It's so hard to think about all of those different product angles. At the same time, I think that what it comes down to in the end as to whether like you can find something that makes it all the way to launch is whether it is something which is good for users overall. There was a lot of – there's a lot of back-and-forth and a lot of privacy questions like we talked about that are really deep and complex.

I think the reason we made it this far is that we all shared this belief that like, “Oh, if we had Slices on our phone in search, we'd like that.” I'd really like to have that personally. That keeps it going. Yeah. The difference in just technology, the way Android is developed an open source and we've got yearly release cycles. Whereas, they've got apps and all this craziness of mismatching release cycles happens too.

[0:41:08.5] JM: Yeah. If there's somebody out there that's listening and they work at –

[0:41:12.4] JM2: I stumped you. I love it.

[0:41:14.6] JM: No. This is actually something that people have asked for more discussion on the show about. If there's somebody out there who works at Salesforce, or Oracle, or Amazon, or one of these other behemoths and they are trying to learn the diplomacy and the education necessary to pull through, or to push through a highly technical change is going to impact so many different interest groups within a large organization. Do you have any suggestions, or tactics?

[0:41:47.3] JM2: Yeah. The thing that we try to do here is I say we all are unified now. We all have our own ways of dealing with this. The way my boss has trained me and I've tried to embody this, now I have a temper and I'm not always perfect at it. No one is, but you're always going to hit cases where you disagree on something. What you need to assume at that moment is that the other team is trying to do what's best for their product. If they aren't, they wouldn't be here. They wouldn't do this for their job every day, if they weren't trying to do what's best for their product.

You need to stop and think about, “Is there a path forward which is best for both you and them? Is there some scenario where this works out for all of us?” That's a really hard thing to do,

especially as an engineer, that is one of the hardest things I've ever gone through, is you have this design in your head, this architecture and you go into a room and you have a room full of people and they're like, “No, no, no. There's no way we can do that. That's a terrible plan.”

To actually stop and not be frustrated and not yell at them, and to be able to be like, “Wait, but why?” What information do they have that I don't have about their product, about what they're trying to do for their users?

[0:42:58.9] JM: Well getting back to lower-level engineering concerns, what had been the hardest parts –

[0:43:03.5] JM2: I see how it is. You don't like my answers there.

[0:43:06.0] JM: No, no. I like it just fine. I think that was – this stuff is more about experience. From what I've seen is there's no easy way to learn diplomacy. You can read books, you can listen to people, give high-level advice like you just gave, I thought that was – that was about as actionable as you can get, but mostly you're only going to learn by getting into the fracas and picking up on people's latent and upfront signaling.

[0:43:31.0] JM2: Well, and I started there by giving you a hard time, just because that's how I am. That is so true. I receive pretty similar advice from my boss for years, and it still took me a long time to get to a point where you can even feel that way when you're in the moment.

[0:43:46.5] JM: Because you're never going to be able to know everything that's going on in the Firebase team, or the Google search team, or the Android permissions team, or the Flutter team, or the augmented reality team, or the assistant team. All these different surfaces that the implications of Slice will touch either now or in the future, and so there's no clear-cut answer of like, “Oh, well I go home every night and I read all of the internal communications between the assistant team and the Firebase team,” and maybe you do that for 10 minutes, but you can't do all of that.

[0:44:20.4] JM2: I can’t imagine anyone doing that.

[0:44:22.2] JM: Nobody could do that. There's some degree of you're doing this round-robin switching between learning about all these different technologies and understanding what's going to be the win, win, win, win, win for the 15 teams involved in ultimately being affected by some minor decision in Slices.

[0:44:40.8] JM2: Yeah. Yeah, yeah, yeah. Absolutely.

[0:44:42.4] JM: What's been the hardest part of building Slices?

[0:44:45.7] JM2: There have been a lot of hard parts. Backwards compatibility. The fun buzzword I haven't gotten to say is we support all the way back to KitKat and slices, which means we go to way up in the 99% of devices. Yeah, backward compatibility is a crazy hard scenario to think about. It's been very hard to get permissions working in that context. We haven't gotten into the details of permissions, but the way permissions for Slices work is they're URI-based permission.

You say you have some URI, which is basically a URL, and you can grant specific apps access to a subspace of that. You say, this app can read anything under /play. The difference from that in standard Android URI permissions is that a user can grant them. There's actually a grant flow for a user, the assumption that maybe you want to use Slices outside of whatever, anyone was thinking of ahead of time. The reason that was so complicated in back compat is that we encountered an issue very late in the game, where we found out that URI permissions have this undocumented limit of how many an Android device will hold on to.

You don't normally hit it, because in the past, URI permissions are used for things like, how many folders does an app have access to? You're not thinking about these in the context of there might be thousands on a device. There might even be thousands for a single app. Very late game, we found out there's this limit, and when we start scaling that's going to be a problem and we had to be like, "Okay, let's rethink our permission story from the beginning and basically build our own version of URI permissions into our back compat library." Yeah, that was something that was pretty hard.

Something which was hard, but not at a specific instance was just thinking about scaling for the future. I would like to get to a point where we have many thousands of these moving around your device and that means that they have to be small and compact and compressed and fast. All of that is something sitting in the back of your mind constantly. You have to think about like, “How big is the wire format of this when it's transferring between apps and how fast does this react? Are we building the APIs to encourage that developers are building these to respond quickly, so that this can happen?” Yeah.

[0:46:49.5] JM: How are they being – what's the communication protocol between for example, search and Spotify?

[0:46:55.8] JM2: Yeah. For Android developers who know content providers, they sit on top of content providers, we hit all the terrible parts, don't worry. I promise. When you get to the core part of a content provider, the really nice thing about it is you say that you have data addressable by a URI, and you can acquire the data and act on that data. A lot of what Alices are at their core. That you extend a slice provider, which is a content provider and you get a callback which is on bind slice, which is basically anytime you want to bind to data from a specific URI to the actual structure.

Then inside there, you have builder schemes and if you're in Kotlin, we have a DSL. Yeah, in Kotlin. That's mostly how it works. Then there's a scheme for saying like, “Hey, something changed. Please trigger a rebind. I need to bind my data again.” I think that covers it at a high level. In reference to the performance, there's a lot of things that we put in there. Originally, we were actually going to run it on the main thread to encourage people to think of this as UI. It's not UI in your app, but it needs to feel UI needs to be that responsive. We didn't end up doing that, but we put the same restrictions as if you were on the UI thread, which is like, you shouldn't be reading from desk, you shouldn't be talking to the network, you shouldn't be doing any long operations here. Instead, what you should do is return, “Hey, I've got to load some data.”

In that, within our API, we've got bunch of a bunch of structures which are like, “Hey, I've got to have a thing here, but it's loading.” You should do that asynchronously and let us know when you're done loading.

[0:48:18.6] JM: Now does that get tricky? Because you don't want to have to have Spotify be running when you see a slice in Google search, right? It would be nice if you could just access the slice data from disk, right, because you don't want to have that application running?

[0:48:36.7] JM2: Yeah. We do. The way that works is Google Play Services, which is the thing which caches them for search, when it asks for a slice from Spotify that it's going to cache, it waits until Spotify says, “I've actually loaded all the data.” It says, “Hey, Spotify give me a slice.” Spotify says, “Oh, here's my slice, but half of it is loading.” Google Play Services is just like, “Oh, that's fine. I’ll wait.” Then when it is fully loaded, it serializes that to disk.

I mean, there are a slice API specifically for apps that are hosting Slices to be like, “Here, I want to serialize this to disk and hold on to it for later.” We actually went into this enough that we made that a core part of the slice API concept, which is anytime you build a slice, you have to specify TTL, which is a hint to the presenter of how stale this data is. We’ll actually show different things in Google search depending on whether we think that this is with – whether this is within TTL, and that means like, “Oh, yeah. We think this data is fresh. We don't need to tell the user that it's old.” Or if it's outside of TTL and we'll give the user some hints like, “Hey, this was updated four hours ago,” and show them a refresh fresh button.

[0:49:43.5] JM: I see. If am searching on the Google search and it's rendering a slice and the slice is taking a while to load, that's not going to affect my Google search results, because the autocomplete for Google search results is this asynchronous behavior, and you can you can enter in the first P-L-A-Y and then you can wait three seconds. If you're on an old phone, then maybe after three seconds your Spotify slice actually will render, because the Spotify side of things will finish.

[0:50:16.5] JM2: Yeah. Well, so Google search actually does a bit more amazing stuff in this. They're really conscious of latency and things loading in asynchronously, because they know that that can be jarring. They'll do things where if you're typing and something is in a loading state, they'll hang on to it for the next character.

Generally, I mean, whether with some exceptions obviously, but for the most part, you won't see the UI in search change, unless you type a character. It's one of those things, at least for me personally, I didn't realize until they explained it to me and then I realized it and saw it on my phone. I was like, “Oh, yeah. It totally does do that, and that actually works really well.”

[0:50:51.7] JM: Definitely. Those performance improvements that you're really hoping for, that scalability, do you have a sense for what that's going to look like or what it will require?

[0:51:01.7] JM2: Yeah, a little bit. I mean, to some degree. Some of it you don't know, some of it depends on apps. You can never be sure what an app is going to do with your API and how they're going to use it wrong. Not to say that all apps are bad, but you get a few million apps, or whatever. I don't know if it's in the millions. I have no idea how many apps are – someone's going to use it wrong.

[0:51:21.5] JM: Yeah, permissions issues, weird privacy things. Like oops, well –

[0:51:26.0] JM2: Or just hanging. Yeah.

[0:51:27.8] JM: Hanging, right.

[0:51:29.8] JM2: Yeah. Yeah. Hanging is one of the most interesting ones, because if you're calling out to them on a – because there's an entry process communication there. If you're calling out to them on the wrong thread, you can just hang one of your threads. You have to be really conscious with that.

Yeah. In terms of performance and scaling, I think right now we're mostly in a wait-and-see. The interesting thing about Slices that is different from a lot of Android APIs is that it's a platform concept, but it's also heavily developed in the support library. There are accelerators in the platform to make it really performant on P and forward and we can even improve those over time, like Android Q or whatever, but we also have very high-level developer-friendly API sitting in the support library. Because of that, we can make adjustments in both places. We can make adjustments way down in the OS to say like, “Hey, let's perform better.” Or we can make

adjustments up in the support library to say like, “Oh, this isn't doing great on P. We're going to change how this actually worked.”

[0:52:30.3] JM: I see. Okay, well as we begin to wrap-up, what other improvements, changes do you foresee for Slices that are going to be made in the near future?

[0:52:39.6] JM2: Yeah. Well, so Android X recently moved to developing an AOSP, which means all of our – all of our support libraries/Android X development is completely public. Last night, I merged a prototype of a library, which is going to make pending intents much nicer for anyone who deals with pending intents. I'm hoping to ship that and out into an alpha release for people as soon as possible, because I think it's going to be – I think people are going to be big fans of it.

You blew a deal with Slices, or notifications, or widgets, or even a number of other things, deal with pending intents and Android apps, and it is a huge amount of boilerplate. Normally, you have to put together your arguments into a bundle and then generate an intent of what you would want to launch and then turn that into a pending intent and finally you have something which you can get to this something, someone.

We're using annotation processing to try to make that simpler in something we're calling remote callbacks, named TBD. Right now I'm calling it remote callbacks. That simplifies that into about two lines of code, which is like, “Hey, I want to make a callback and I want to call this method with these parameters, and it will just generate the pending intent for you.”

[0:53:45.2] JM: Cool, and by the way when you're making these kinds of changes, that have ramifications for so many different types of phones, so many different operating system versions, do you have some comprehensive testing, continuous delivery thing that you push the changes to and it runs on a bajillion different device configurations before actually getting accepted into Android core?

[0:54:11.1] JM2: For the most part, we don't. For the most part, we don't. There are two scenarios here related to what you are referencing, which is there's stuff which is in the OS code, which runs on your device. We do have a bit of comprehensive testing across pixel

phones and emulators of what we expect, but there's no real way to test that on say a Samsung device, because it's part of the OS and Samsung would have to build it, or LG, or Sony, or whoever.

Then there's the other end of the spectrum, which is support library. That testing is on a similar set of devices, I think just because of historical reasons mostly at this point, because in the past, support library when it started as support library, it was just a part of the OS being like, “Hey, let's backport this stuff for people.” Now it's developed into this whole Android X jetpack thing and we’re like, let's actually make developers life's easier and really think about that as a core tenet. I don't know where I was going with that, but that seems like a good place to stop with that. That's where I was going with that.

[0:55:06.5] JM: Agree. Well Jason, I want to thank you for coming on the show and it's been really great talking about such a wide variety of things. I appreciate you going there with the stuff that's probably on the verge of discussable with assistant and whatever other mysterious surfaces Slices will be deployed to, and maybe we can do another show at some point in the future when those surfaces have breached reality.

[0:55:27.9] JM2: Yeah, absolutely. It's been great being on here and getting to talk about this stuff.

[END OF INTERVIEW]

To find out more about sponsoring the show, you can send me an e-mail or tell your marketing director to send me an e-mail jeff@softwareengineeringdaily.com. If you're a listener to the show, thank you so much for supporting it through your audienceship. That is quite enough, but if you're interested in taking your support of the show to the next level, then look at sponsoring the show through your company.

Send me an e-mail at jeff@softwareengineeringdaily.com. Thank you.

[END]