EPISODE 324

[INTRODUCTION]

[0:00:00.1] JM: Application program interfaces, otherwise known as APIs, define how applications interact with each other. When a developer creates an API for other developers to consume, it is important to design that API intelligently. The best APIs are simple and clear, but also flexible enough to be useful for a variety of consumers.

Andy Beier has experienced a broad range of API quality in his role with Domo and creating integrations with other businesses. His mission is to standardize good API design practices with an emphasis on making the right information easily accessible without having to download more than necessary. He's worked on promoting standards to make APIs easier to create and consume.

In this episode, Andy joins guest host, Dave Rael, for a conversation about API design standards. They discuss what makes for a good API and steps to move the broader technical community towards more useful and secure APIs. Dave is guest-hosting this show and he's also the host of *Developer On Fire*, which is a fantastic program about the stories behind engineers and how they became the engineer they are today. It's a great compliment to Software Engineering Daily, and I encourage you to check it out.

[SPONSOR MESSAGE]

[0:01:27.1] JM: Software engineers know that saving time means saving money. Save time on your accounting solution. Use FreshBooks Cloud Accounting Software. FreshBooks makes easy accounting software with a friendly UI that transforms how entrepreneurs and small business owners deal with the day-to-day paperwork. Get ready for the simplest way to be more productive and organized. Most importantly, get paid quickly.

FreshBooks is not only easy to use, it's also packed full of powerful features. From the visually appealing dashboard, you can see outstanding revenue, spending, reports, a notification center, and action items at a glance. Create and send invoices in less than 30 seconds. Set up online

payments with just a couple of clicks. Your clients can pay by credit card straight from their invoice. If you send an invoice, you can see when the client has received and opened that invoice.

FreshBooks is also known for award-winning customer service and real life person usually answers the phone in three rings or less. FreshBooks is offering a 30-day unrestricted free trial to Software Engineering Daily listeners. To claim it, just go to freshbooks.com/sed and enter Software Engineering Daily in the “How Did You Hear About Us” section. Again, that’s freshbooks.com/sed.

Thanks to FreshBooks for being a sponsor of Software Engineering Daily.

[INTERVIEW]

[0:03:07.2] DR: Andy Beier is a director of software engineering at Domo. Andy, welcome to Software Engineering Daily.

[0:03:12.7] AB: Pleasure to be here.

[0:03:13.7] DR: Great to have you here. You’re focused on integrations and connecting various services in the wild-wild-world of the internet, probably led you quickly to some conclusions about the state of API proliferation and the need for standards. Why are API standards important?

[0:03:28.4] AB: Yeah. To be clear, one of the key things I’m focusing on, we have a tremendous amount of thought leadership around how to build a really good RESTful API. Certainly, in the industry, and I’ve done with us for a very long time, there’s not this debate in terms of what the web service technology is, right? Everyone’s building, basically, to RESTful APIs. Most of the listeners out there will know, as you do too, that REST is obviously not a strict protocol where things like XML-RPC or SOAP are were that way. It’s more of a philosophy, right? We’ve got all these great minds and frameworks and people really giving a lot of thought leadership towards how do you construct these RESTful APIs, how should it work?

One of the things that I noticed in the course of my duties at Domo with all the work we've done connecting to third-party APIs is that while there's a ton of thought leadership around that, there hasn't been as much leadership around the content of the APIs. One of the examples I like to use, and it's simple, is this, which is HDTV. How interesting is your HDTV without the content?

[0:04:45.9] DR: Sure. Yeah.

[0:04:46.7] AB: Right? We have this great APIs, but we start to get — and they're structured well, but when you start to get into the content you're trying to pull through those APIs, and many of them fall flat.

[0:04:58.0] DR: You gave a picture there of — There are some existing standards. We're talking about HTTP kind of at a lower level, and then REST. Obviously, writing on top of that. What you're talking about is kind of some standards for what is the content that rides on top of REST? Rest, I guess, you pointed out that it's varying degrees of standard. There's some consideration of using HTTP verbs the way that they were intended, and using those for the actual proper actions that you want to do against an API, but that doesn't really tell you a whole lot more about the content. Am I characterizing kind of your position there?

[0:05:34.1] AB: Yeah, I think that's absolutely correct. To kind of further extend no that, there are some areas of it where example — I categorized APIs into three basic categories, which is essentially administrative. Very familiar with that, it's like, "I got to get users into the system. I got to get users into groups," kind of a management API usually for a product or some kind of directory service.

In terms of trying to do anything we're trying to report, not terribly interesting. Not to say that there are use cases per se, and, "Hey, how many users do I have in this group?" or "How many users do I have in this system?"

Taking a step back, what I find far too common in the case is people have an API and it's essentially what I call a base state API, where if I want to ask a question, and that question might be, "How many sales did you have last quarter?" I would have to basically pull every day of the entire last 90 days and then bring it all into my system and then run an aggregation on it.

As supposed to the third type of API, which is a reporting API, where I can just simply say, "What were the total sales last quarter; metric and dimension," and get that answer.

[0:06:49.8] DR: Sure. There's a place for all of those. Obviously, you want to be able to do the administration tasks and those kinds of things, and there's really no substitute for that. The base data APIs are probably useful for some of those nonstandard cases. I think a lot of kind of the idea of adapters in design patterns, or kind of the idea of an anticorruption layer and domain-driven design. The idea that you've got some way of speaking to a particular system in different ways for different needs, and those reporting APIs are probably most of your use case. Those base data APIs are probably useful too to give you more flexibility in those times when your use case may not be covered by what's kind of the standard case. Is that kind of your way of thinking of it too?

[0:07:37.2] AB: Yeah. I think the key thing is not necessarily saying that they're all bad, because they all have specific purposes and they do specific things. I think the issue you run into at this point in the evolution of data is when you get to those base data APIs, sometimes they're sitting on top of just massive data lakes, and the practicality of having to pull all that data just doesn't work. Looking at the industry as a whole, how much investment has been made in essentially map-reducing data? Just simply for the fact that the sheer volume of it is too much.

All I'm really calling out cautionarilly is that there may be a place, because, definitely, there are times when you need to remove all the data for a system for a very good reason, or you need to put a large amount of data in. Definitely, focusing on bulk operations, because the example is — I think you would agree with me here. If I have a case where I'm literally trying to pull out years' worth of data, it probably makes more sense to do that in a more condensed format, like a CSV than it does a paginated JSON set.

[0:08:44.8] DR: Yeah, that makes sense. You have different needs for different data, and so you want to be able to get to it in the format that makes sense for your particular use case.

I think, at this point, let's step back a little bit. I think we've talked about APIs in REST, and REST, particularly, is something that — It's one of those things that software developers should know what they're talking about, and we use it all the time. For a person who finds themselves

in the situation where they really don't know what REST is, but they might look silly if they ask that kind of basic question. Could you describe in your own words, what is REST anyway?

[0:09:22.2] AB: Sure. I think in this most basic form. What we're really looking at here is the ability to make a simple HTTP call, which could be as simple as a GET. A simple GET to a system, at which point, I'm going to get back, what we would call a document body, which is essentially a stream of, essentially, text. Usually in the case of most APIs these days, that would come back in the form of JSON, JavaScript Object Notation Language, or in the more earlier days before, I think, we'd really standardized JSON, you would see a bit of XML, for sure. Even today, and I still think it's an appropriate use case, you may see actual formats like CSV, because those provide, like I was saying earlier, a very compact way to transmit large volumes of data using the protocol effectively. As we know, CSV streams exceptionally well as supposed to some of the other document formats.

[0:10:26.6] DR: Sure. Yeah, it's different formats. I guess you're kind of getting in to some of the technical details right there of things like using verbose things like XML and JSON less so, kind of the evolution of kind the evolution of kind of getting more practical in our transports, in our serialization. What about things like protocol buffers and stuff like that? Are you seeing some use of that and, perhaps, advocating that as well?

[0:10:52.5] AB: The vast majority of the stuff that I've seen traditionally — I have not seen as much of that take place. I think that as the data sizes increase, that it's certainly something that I would expect to see more. Out of the vast majority of APIs, I think one of the beauties in terms of the product management aspect of it is people have spent some efforts, "I'm trying to make them very user-friendly, so you don't have to bin someone who had been coding for 20 years, that they're very approachable." I think that's a really good thing.

I think the other thing to consider is I am talking for my use case, which is primarily a backend use case. Obviously, there is an immense use case for also using some of these APIs from, essentially, a front-end application as well.

[0:11:44.5] DR: Yeah, that makes sense. There's definitely different needs for different use cases. Who should care about APIs and their quality, and why should we care?

[0:11:54.4] AB: I think the most interesting thing about this is we have to come to a fundamental understanding especially in the age of SAS software and most things being in the cloud, that there are products out there, but that data is ours.

I'm sure you're familiar with Facebook, and they have a feature where you can actually go in their URI and you can download all of your data. What I like about what they've done there is they have made a point, which is we're not keeping you on Facebook just because we got your data. If you want to take your data and go, you can go. We're confident in our product.

The reality is, is that whether or not I've been using a product for one year, or 20 years, that is my data. That's all the data of my business, and I should have the ability to access it and use it for any of my business purposes. That doesn't give me a universal license to abuse whatever product that is, but that data is fundamentally mine.

As in this age, we as consumers, have to absolutely demand that every product we use have an API, and we need to clearly message, that data is ours. Keeping it locked up in that system is fundamentally wrong, because you're basically trapping — You're holding what is mine from me, which I've been paying to use a service, but the data is still mine.

[0:13:27.3] DR: I like that a lot. You're putting it in terms of ownership, that this information about me belongs to me, and having rights really expressed explicitly about who this belongs to. That's really important to make sure that those things are known and accessible. That's a good example of that.

What is the problem really? You described kind of the different forms of API, and that if I have a question that I want to ask, that these reporting APIs aren't necessarily always there and being accessible to answer the question that I want making me go through some gymnastics and some hoops to get the answers. What other problems do you see? Are there some more things that some API standards could address?

[0:14:12.8] AB: Yup. I think the other thing is just size of data, and it's a very fair thing. People place API call restrictions on most APIs. It's a very fair thing, because it's kind of like the same

way the reason why I walk my front door. It sets a clear boundary on where the rules are. That's okay, but then what you find with a lot of APIs, to answer some of those questions, particularly if it's not a reporting API, I bump up against the API call limits.

Here, I've got a case where I want to answer a pretty in-depth business question that really requires me to do quite a bit of, if you will, potentially data mining or large extraction of data, because if I can't ask the fundamental question. There are APIs that, because of the API call restrictions, you may find yourself, "For me to get an answer to this question, it will take me two weeks. Because if I properly throttle and use these request, two weeks from now, I'll have the question. Of course, I'm sure where this goes. By the time I get the answer to that question, it's two week's old.

Now, I'd like to know what the next state of that question is, and it'd be another two weeks. It that begets another question, maybe that's four weeks' worth of data mining." Looking at call volumes, but more importantly, how you structure API to maximize the data transmission, is got to be paramount as well, because it's not very useful if it's going to take me a month to answer one question.

If you haven't tried Datadog at your company or on your side project, go to softwareengineeringdaily.com/datadog to support Software Engineering Daily and get a free t-shirt.

Our deepest thanks to Datadog for being a new sponsor of Software Engineering Daily, it is only with the help of sponsors like you that this show is successful. Thanks again.

[INTERVIEW CONTINUED]

[0:17:18.3] DR: Yeah. I guess that kind of goes back to the last question I was asking about. You've really painted the picture of both consumers and producers of API having some strong incentives for presenting the right thing in the right way in the first place. Producers who are paying for this bandwidth and need to have some controls on that and considerations as to what it is that they're sending back and all of those things, and the consumers as well, being able to easily answer the questions that they need and not running up against limits and all of those kinds of things. It benefits everybody. I think that's a pretty good picture of that.

If you pull out your crystal ball and look forward into the future, what do you think the future holds for where API technology is going to go? Do you think we're going to address some of these things?

[0:18:06.1] AB: I think we absolutely are. There's no doubt that with just the explosion of big datasets, that those are only going to get larger as time goes on. I think that it's very exciting when you're out kind of talking to people in the industry to see just how many places have made investments into large data store type, Hadoop things, with MapReduces on top of them. I think that's going to become more and more common place to where we're very much going to get to a place where we can say, "These are the basic metrics and dimensions that I want to answer this question, and systems will be able to address that question very specifically and very quickly."

Of course, it will always the use case for then after answering all the summary questions. You almost always want to start digging deeper and data mining. Those may be bigger and longer

and more expensive operations, but we'll really see, I think, a whole entire optimization on top of whatever stack you're using to just get those answers almost immediately.

[0:19:12.0] DR: Yeah. That's definitely a bright future. When there's pain, people kind of tinge to come together to try to resolve that pain and make things better, especially when they're feeling it themselves.

We've talked about kind of the importance of good APIs and the importance for providers. More fundamentally, who should have an API? It's starting to grow more and more that a lot of services are providing APIs and hooks for integrations and all of those kinds of things. If I'm putting up a new web project, what are the conditions where I should think about exposing an API for my system?

[0:19:48.5] AB: I think that's an absolutely great question, because you can just see how the explosion of essentially cloud-powered applications. I think the key thing is if you're collecting data — Because there's different kinds of things in terms of APIs, but I think you could almost make an argument for just about any system.

Here's a great example of an application I came across the other day. Are you familiar Pocket at all?

[0:20:20.3] DR: Yeah. It's a little bit like Instapaper. It's kind of saving something to read later.

[0:20:24.9] AB: Correct. Here's an example of — I understand the content is — Those are essentially news articles, and those don't necessarily belong to me. What is fundamentally interesting is, is I do still feel like the process of me curating that collection and what that collection is, is mine. There's an example of — And they do have an API, that I should be able to keep that data.

I think anytime you're essentially building a metadata store or actual data — You know what I mean? So you could look at it from a CRM perspective, something like Zendesk, where I've got incoming support tickets and I handle those. All those support tickets are definitely mine and

that should absolutely have an API. Then, you get into some of these more social aggregators from metadata collectors.

In the case where it's an aggregation service and it's metadata on top of that, that metadata should still be available to an API so that I can curate that and keep that for myself. I think the key thing is, is look at the successes of people like Facebook who created an entire app ecosystem and how many people are writing apps and integration and building in there. Extending your platform to allow people to enhance it further, I think you can't deny the crowdsourcing element of that.

[0:21:42.6] DR: Sure. Exposing — We already talked a little bit about exposing your data so that you can have access to it. I'm thinking, basically, of the example there. You were talking about Pocket. Feedly I think has some similar kind of ways that you can get access to your feeds. A lot of podcast players too will export to a standard XML format that can then be imported to another application so that you can have your same feeds and all of that stuff. Doesn't have any of the meta information about what you've already listened to and some of those thing, which I think would be ideal as well.

Really, democratizing some of those things, making it accessible to you in a way that you can use it, just like you said, with Facebook; I can take my information, use it somewhere and be confident in the product that this product is something worth using on its own rather than just being a matter of locked-in data.

[0:22:34.0] AB: Yup, exactly. I think one of the most important things here is people are making great software. Be confident in your software. I think the API is an extension of that. People will stay because your product is awesome. They'll eventually find a way to leave if the only thing keeping them there is the data.

[0:22:52.1] DR: Yeah. Compete on awesome software. That's really a great way to put it. What standards already exist? Are there some of these things out there already that are kind of pushing people toward this stuff, or is this kind of a completely wild-west kind of a thing?

[0:23:08.3] AB: Once again, obviously, you go to any tech conference, there is a ton of leadership around how to structure RESTful APIs. There's a ton of frameworks pushing all that. To be clear, the differentiator that I think I'm bringing to the forefront is the fundamental content that we're trying to push, or you're trying to push through the APIs, because I've seen far too many cases where we've built an entire integration into a system and then to find out that this fundamental piece of data isn't available through the API, or you can't get at that measure, or metric.

Even though it's the most beautiful REST JSON API you've ever seen, with perfect pagination, those fundamental gaps make it then almost worthless. There's a lot of great people out there, like I said, pushing the other standards, but the standards for APIs .org is the one where we've been trying to push getting people to think more holistically about these guys. Certainly, some of what we push is a little bit of design structure.

One of the things I do want to call attention to is people are definitely in the REST age, really should version their APIs, meaning that there should be a current version minus one, because you shouldn't force people to update to the latest API. Also, if you're going to cause breaking changes, you really need the N minus one so people have a chance to migrate.

[0:24:38.9] DR: That's a great point; one of those pieces of making a good API. The URL that you cited there about the API standards going forward, is that controlled by you? I kind of got the impression that that was your place. Is that right?

[0:24:54.2] AB: Yeah. Mine, I think, is a little too strong. It is an effort that the company I currently work for, Domo, has stood behind. There's a bunch of us here going out and support of that, and we went on a three city tour and had a discussion with people across a variety of industries and open discussion. Just trying to bring about these thought ideas into the community.

The point is, for it to be ours, I don't think is the case. We really do want to hear from people and really want to engage in essentially a thought leadership that's not just known by us.

[0:25:38.1] DR: Okay. It's created by Domo and owned by the community I guess is kind of the gist of what you said there. Am I saying that right?

[0:25:46.0] AB: Yes. The long term vision is, is want to build a community around it, of people that are adopting some of the standards we're trying to push. We're also taking input from the community, because a lot of — When we went on a roadshow, if you will, we were asking people for their input and their thoughts. There was quite a bit of interesting feedback in the process.

[0:26:07.8] DR: Okay. I was going to ask you about why should I listen to this particular organization, this particular thing, but I think you've answered that. You've addressed that this is more of a community effort. I guess the question then is, why Domo? What's the insight? Why should the listener out there who's interested in API quality listen to what you guys have to say?

[0:26:31.3] AB: I head up the area of connectors, and we have built over 500 connections into different APIs. We have done this time and time again with a variety of systems across a variety of industries. We've seen a lot of what is offered in the industry and a lot of the approaches. We have that why perspective.

Part of what we're trying to do is saying, "Hey, after doing this as many times, here is what we've seen. Here are some areas of concern, and here are some things to think about that will make this better for all of us." Because as I talked about earlier, the idea that you can democratize data within your company, your organization, and across your products, I just don't see how you're going to be able to push forward into this, basically, golden age of data we're in without being able to interchange data like that.

One of the great things that our product does is give you a vision, an insight across your entire business pulling data from all these data sources so that you can have really engaging conversations about that data and really optimize your business.

[0:27:47.0] DR: Okay. Yeah, nothing like experience. You've run up against a lot of the problems just because you are consuming so many APIs. It makes sense that you would know something about this. Especially from the consumer's perspective, what it's like to deal with

APIs that are less than desirable and less than what you're looking for. That makes a lot of sense, and I think there's a lot of lessons learned from that.

How about the road trip? Tell me a little bit more about that. What did that look like? Why did you decide that that was the right venue to get this discussion going and just — How did it go? Did it bear some fruit?

[0:28:21.5] AB: It definitely bore some fruit. The idea was it's one thing to kind of put out a website, put out some papers which, certainly, we did that too. It's a totally different thing I think, to connect individually one on one with developers, because I think it's disingenuous when you say, "Hey, here's some problems that I've seen with APIs. There was a reason why people made some of those decisions.

To go back to some of the use cases where it's like, "I really needed this one piece of data," and then it turns out, architecturally, in that given system, that that's actually a really hard thing to do, because it may create some system performance problems. You'll talk to people and they're like, "Yeah, I could really see how you need that. The reason why we didn't do that was this." It starts this discussion of, "Well, I could really see how we need to make this investment."

It's more about, I guess, trying to say, "Hey, here's some pain we feel." What pain are you feeling and understanding what it felt like to be on both sides of the issue? With anything, the more points of view, the more people you include, the more insightful those conclusions become. Going out there and walking the pavement a little bit I think really helped us.

[0:29:37.3] DR: Contributing to understanding, I think is great. It is. I think you made a good point there about that if I'm saying, "Hey, this is bad. I've seen examples of this." It's really easy to get defensive and say, "Hey, I've done that." To say, "Hey, who are you to tell me that I'm doing it wrong?" Rather than taking a you're doing it wrong approach, you're talking about your experiences, "Here's what I've had to deal with and why I've had some problems with it. Here is the case for some ways that we might do this better that creates a better experience." I think that's really cool.

In a perfect world, if you project forward to a place where everything is exactly the way that you would like it, what would it be like? What would the experience be like for a developer consuming an API without having any prior knowledge of, it just kind of how it works and what it's doing? What do you think would be the ideal experience?

[0:30:29.4] AB: I think the first thing is every API in that vision would have API Explorer tool. I'm assuming you're familiar with those where, effectively, there's a little web council I can go into and hit the end points and what returns. I think that fundamentally gives me an experience with it before I've written any code.

Now I can see what all the different behaviors are without having to stub that out my end, because it's a little bit more of an expensive process, and I'm not saying you can't do it cURL and Postman, but it's really nice when someone guides me to the flow of the API, and you lose that context.

[0:31:06.0] DR: You're seeing kind of a sample user interface that you might be doing something along those lines. You're seeing what requests and responses look like as you interface with the — As you interact, I should say, with the particular end points and all of that stuff. Just some good examples of the thing in action, I guess, is a big part of that.

[0:31:26.9] AB: Yeah, absolutely. I think the second piece to that is to have an API that allows me to look at specific questions, and to your point, may be able to extend into a more base state of use case, but in a scenario where I can get very compact result sets; or if I need to, I got a way to transmit that data that travels in bulk well, too. You kind of get a look at the bulk operations, and you got to look at the individual operations. I think those two distinct ones looking at both reporting and the administrative functions makes a lot of sense.

[0:32:04.7] DR: I think that's great. You've done a lot of consuming of APIs doing these connectors to expose for Domo. I imagine Domo itself is probably exposing a lot of APIs as well, maybe not as much, but do you have some experience in the world of being an API provider as well?

[0:32:23.2] AB: Yes, absolutely. Just like anyone else, we have an API. We have an administration portion of our API to manage, as we talked about, those user and group-related tasks as well as have a RESTful API for getting data in and out of our system. Then, further extending it from there, we even have an SDK whereby which you can actually extend our system and make what we call apps that run within our system. That's essentially an angular type of web application that fits within our product that allows you to connect to our data layer, and create your own very, very custom and specific visualizations.

[SPONSOR MESSAGE]

[0:33:20.9] JM: You are building a data-intensive application. Maybe it involves data visualization, a recommendation engine, or multiple data sources. These applications often require data warehousing, glue code, lots of iteration, and lots of frustration.

The Exaptive Studio is a rapid application development studio optimized for data projects. It minimizes the code required to build data-rich web applications and maximizes your time spent on your expertise. Go to exaptive.com/sedaily to get a free account today. That's exaptive.com/sedaily.

The Exaptive Studio provides a visual environment for using back end algorithmic and frontend component. Use the open source technologies you already use, but without having to modify the code, unless you want to, of course. Access a k-means clustering algorithm without knowing R, or use complex visualizations even if you don't know D3.

Spend your energy on the part that you know well and less time on the other stuff. Build faster and create better. Go to exaptive.com/sedaily for a free account. Thanks to Exaptive for being a new sponsor of Software Engineering Daily. It's a pleasure to have you onboard as a new sponsor.

[INTERVIEW CONTINUED]

[0:34:52.5] DR: Make it pretty easy to test out what you're doing, too, I imagine is probably the case there. Back to that perfect world that we went to for what it's like for a consumer. Can you

paint me the picture there as well of what it looks like to be creating an API? What's the experience like for the producer of APIs in this idealized world?

[0:35:14.8] AB: I think definitely, we will have flushed out the use cases to a larger degree than we have before, meaning that we're now, where we're putting out these ideas about base state APIs, reporting APIs, administrative APIs, I think it will be by default that you'll have all three to some extent as it makes sense for your system. That will be there, and I think that the same way the frameworks have done an amazing job — For those who have said they've been with us for a while. Back in the day, you had to do quite a bit of work to structure your responses.

Now, the frameworks make it very easy to create JSON objects to parse JSON objects. I think they will be extending those frameworks into those use cases where you fall into success very easily, trying to create an administrative API or base state API or a report in API, that all those fundamental stubs will be there, and it will just push you in that direction very easily without having to break a lot of inertia.

[0:36:12.8] DR: Yeah, that sound great. Standards are kind of like the interfaces for the classes in your code. We're often creating those explicitly and thinking through the abstruction before getting into implementation. I think it's often also the opposite, that we've got an existing implementation and we're trying to pull out the abstractions a little bit. APIs are probably pretty similar to that.

In the real world implementation of APIs, who is out there that could serve as kind of the exemplars that could become a standard? What are some of the people who you think are doing it right already?

[0:36:50.8] AB: I would definitely say Salesforce. For anyone familiar with that API, they've got an amazing REST API. What it does is they've created this so-called query language. What it allows you to do is, essentially, it's an object query language, but you can use that to tailor the exact result set that you want to get back from their API.

If I've just focused on a certain number of records with a certain number of fields, I can absolutely just query the system for those and return them. I think that that's probably — The other thing that's really key here is authentication.

One of the things that I don't like in the current age is doing anything other than OAuth, because the wonderful part about doing an OAuth implementation is the user has the ability to, one; revoke the — The person granting the access has the ability to revoke it at any point. However, you never ever, ever share any type of user name or password with the person asking the access. It also clearly requires that person to represent to you, "These are the exact things I'm pulling from this other system on your behalf, and these are the permissions I want."

I'm sure we've all seen this in Facebook where you go to do a Facebook app, and it will say, "Do you grant this app permission to see your friends' list?" In some cases, you're like, "Yeah, I'm okay with that." In other cases, you're like, "I don't know if I am, but I'm given that choice upfront. Everything is clearly spelled out to me."

At a certain point, if I need to revoke access, I know that that access is immediately burnt. I knew what the scope of the access was, I knew what the duration was, and that's really murky when you're doing it daily with user IDs and passwords and sometimes API keys, "I got to go into this system. I got to revoke the token," and it's just a much cleaner, safer, better way to go when it comes to authenticating APIs.

[0:38:52.7] DR: Really nice to give consumer choice about what they are able to authorize and those things, so that is absolutely wonderful. I want to ask you a little bit more about OAuth, but before that, I want to back up to talking about that Salesforce API and that query language being available there. That sounds to me like a hybrid between the base data APIs and the Reporting APIs, that I'm able to query some of that base data but do some filtering and bring it down to the useful piece of it. Would characterize it as either base data or reporting, or maybe am I looking at it all wrong?

[0:39:32.9] AB: No, I think you're looking at it right, but they're a further exemplar, because, to your point, I can use the circle interface to do that, but they also provide a bulk operations API, meaning that there is a way for me to bulk, put records in and pull records out, if I want to do

that at mass. That's another example of how they've really thought that use case through quite effectively.

[0:39:57.1] DR: Back to OAuth. I think it's a wonderful standard, and it enables all of those things that you were talking about of the consumer having control and being able to go back to Twitter and revoke access for particular applications and all of those things. That's cool. Are there some downsides to OAuth? Are there some things you wish were better about it?

[0:40:18.0] AB: At the current time, I'm really looking at the OAuth 2 standards. At this point, I think we probably done — I have to get the exact number, but it's definitely over 80 OAuth implementations. I feel OAuth 2 did struggle early on, because the standards weren't completely solidified. We did see a lot of variations. I think in terms of OAuth, it would be nice if we could get some more standardization at least in terms of the part where — The most key part for those people familiar with the protocol, there's the part where I need to essentially redirect the user back to the — The transaction starts with me. For example, if we want to connect up Salesforce, I will forge you from us to a landing page in Salesforce that will clearly spell out what actions I want to do on your behalf and what you're granting me rights to. Then, if you agree to that, you're redirected back to my site.

At that point, I need to exchange, essentially, what is called the code for the access token. At that layer, there's a lot of ambiguity with some of the implementations. Salesforce has done their implementation exactly as you would expect to happen with the spec. I see a lot of people in that interchanged part where sometimes they just redirect you back and give you the access token, which is really not the protocol. Then, I see a lot of different payloads for when I go ahead and exchange the code for the access token.

I've see some come back as XML, I've seen some come back as JSON, I've seen some would come back as URL and coded document body. It would be really great if we could really get everyone to just give a JSON object back with the access token, the refresh token, and the expiry ubiquitously all named the same, because there are a lot of customizations you have to do on that flow to kind of adapt to that. It would just be nice if that were automatic.

[0:42:16.6] DR: It sounds like it may be that the standard is not specific enough or maybe just that there are a lot of individual actors who are not necessarily working according to the standard. I'm not sure which is true, but getting more consistent in that flow just makes things easier for everybody to not have to re-implement that all over the place.

The idea of the standard — When I first saw this idea that you're being bounced around as a user in your browser, and I come to a particular site, and then I'm being bounced around to another site and back. It seems like a lot and it seems complicated, but for the consumer, it's really a very simple experience. It's just, "I go back to people and I say, "Yeah, do this thing," and then all of a sudden, I'm back on Domo, and I'm often running." Is that the experience that you typically see as a consumer with these things?

[0:43:12.7] AB: Yeah, I think that you've hit on probably one of the most important points, user experience. Under the API key example, it's really hard to sometimes know, "Okay, I need an API key. Where do I get that? How do I provision it? Do I need to be an admin in this system?"

The OAuth model makes it as easy as, "Oh! Look. Now, I'm at Google. Okay, I know my Google user ID and password." Type it in, get prompted, understand what I'm granting access to, and click okay." Next thing you know, I'm back in the product, the authentication step is complete, and now, I can start working. I think that's just a far better experience.

You can argue that if you just code username and password in the front of whatever you're building, you can also provide that, but then you circumventing, I think, the wonderful part of the security that OAuth implements. The most key thing that I think people forget — How often do you change a password? If people don't do that of their own accord, it's usually forced on them pretty regularly.

Now, your integration breaks, and it's broken. Now, the person has to come back, re-authenticate their application, sign back in, make all that happen, versus where the OAuth has the refresh token model so that can go on in perpetuity.

[0:44:37.2] DR: Yeah. The other thing, too, about making you give me your password for some other service, is that in order for me to use that thing — If your password in n my site, then I can

use salting and hashing and all of that stuff so that I don't know what your password is. If I have to actually use that password to log in to another service, it has to be retrievable by me. That's a lot of trust for you to put into somebody to say, "Hey, here's my password to my Google account," or perhaps something even more sensitive than that to say, "Hey, here's my password, and you can use it on my behalf. You can recover it, and you've got administrators in your system that can go in there and probably get access to my password. Whereas, with OAuth, there is never any of that, so I think that's really important.

[0:45:22.1] AB: I completely agree. I just think, security-wise, it's the best way to go. It doesn't take a ton of analysis to look at people like Google, people like Facebook, people like Salesforce, people like Twitter. They are all using OAuth implementations. That's a really good thing to emulate.

I don't see very many — They don't offer anything in the API key realm and they have strict user policies against accepting a user ID and password. This is a clear violation of their terms of use, and they're doing that for everyone's best interest, because they want to protect people's data.

[0:46:03.1] DR: Makes sense, yeah. Security is obviously important in a big piece of OAuth. What else should developers in businesses know and keep in mind about API security?

[0:46:14.2] AB: Obviously, it doesn't — A lot of the stuff is well-documented, well-published, and we just need to follow it. I think one of the best examples is if you cannot at all avoid it, do not be putting query string parameters in. Transmitting most of the key information as an HTTP header is the best way to do it; document body as a second, query string as a third.

Obviously, in this day and age, if you're not using encrypted connections, that's hugely egregious. An SSL Cert is so inexpensive at this point that we can't ignore that. I think that when you start to build your API, just remembering making your authentication model with the OAuth, and how you manage your sessions, making that the first thing you design, will always lead you in the right direction.

[0:47:12.9] DR: Makes sense. Yeah, SSL — Even to the point of being free now with Let's Encrypt. Lots of good things happening there, and security just getting more and more accessible, much like APIs. All good stuff with that.

I'm really big into services like IFTTT and Zapier, and of those things. I think that has a lot in common with Domo, where this mash-up of information from different places and being able to connect events in one system to actions happening in another, and I think it's all really cool stuff. Do you see some big things coming in the future? Where do you see kind of this API mash-up kind of thing going?

[0:47:53.8] AB: Yeah, I think it's been one of the most interesting things to watch and it's just extremely enjoyable to see the things people create. When you talk about IFTT, just the Maker Channel. For those that aren't aware of it, they provide what they call a Maker Channel. This allows you to, essentially, without having to build a formal channel within IFTT. Example, if I wanted to integrate my raspberry pie within IFTT step, it allows me to, essentially, do a REST call into my raspberry pie.

It's enabled this consumerization, where if there isn't everything I need in the ecosystem, I can build my own thing. I think what's going to be — There's going to be an explosion in the next few years. It's just all these amazing internet of things devices that people are just going to, essentially, build in their garage that are going to beget just some of the most amazing consumer products we touched.

[0:48:53.7] DR: Definitely. That's a bright future. I like all of that stuff. One of the ways, I think, that we can think about good APIs, and we've talked a lot about things that make APIs good. We can think about the opposite, some of those things to avoid, and I think we've hit on some of those. Can you name some kind of API anti-patterns, or just tell me about some things that we definitely should not be doing?

[0:49:15.8] AB: I think the key thing is, is please put away all the SOAP APIs. At this point, I think the battle is lost. I think REST has become the ubiquitous standard. It's time to put those aside. The other thing that I see, which is also a major no-no, and this happens more as a result

of things like SOAP, where you're doing more of a protocol type operation, where you'll find that, all of a sudden, the object you're receiving is a language-specific object.

You go to get a daytime, but now it's a daytime object for a language that you're not working in, so the interoperability is completely destroyed. We've seen a few of those. What you do is you basically have to cast the thing to a string and then parse it. It's not that it can't be done, but definitely not the — Go ahead and use an ISO date format. Don't think of things in terms of passing direct objects, but think of in terms of what your object model is and your application. Think about what the universal object model should be as you're making your API available to all kinds of developers across all kinds of platforms, and focus on that.

I think that that's probably one of the anti-examples of — If you're really into one programming language and you code it so that your API really just works well with that. I think you're ignoring the entire ecosystem of people who want to connect.

[0:50:41.4] DR: Especially today, all over the place, there're so many different things in use. It is really nice, like you mentioned at some point earlier, that libraries are getting a lot better, that we can lean on some of those to serialize those out to standard formats and some of those things. There are a lot of good resources to avoid that kind of anti-pattern. Thank you for that.

How can developers and businesses, and consumers as well, how can they get better educated about APIs, and good APIs, and API standards and all of these stuff we're talking about?

[0:51:10.0] AB: Yup. Definitely visit our standardsforapis.org website. The other thing is, is I think what I want to really call attention to is there are lots of opportunities at conferences. There are lots of opportunities are local users groups. Don't make this — This isn't necessarily my cause. We're all going to benefit from this. I really encourage people to engage in thought leadership through all the avenues that they can; by writing papers, by talking to conferences, and let's really explore these idea, because I think the more of us that jump on this and the more of us that really make that a focus of our APIs, we're just going to make all these applications in the interconnections of the internet so much better and we're just going to enable just some amazing things.

[0:52:04.5] DR: Where do we go from here? You've given some good resources there. What are some other ways that listeners out there can say, "Hey, I'm going to get involved in this quest and do some better things for making APIs better for the future."

[0:52:20.2] AB: Yeah. I think one thing is we all owe it to ourselves to build an integration to an API. Most of the audience listening most likely knows how to code. Most systems offer a free sandbox to try. I recommend picking an API, something that you're passionate about, something that you're interested in. Build that integration. Work with that AP, something that you have no control over. Think about some fundamental things you're trying to do with that and then go after it. See where the problems are.

Then, really, start putting in the communities, "Hey, I used your API. I thought it was great, but here are some things that I was trying to do that it didn't support." Offer a lot more of those opinions and ideas to help us all get better, because in a vacuum of feedback, things aren't going to change.

[0:53:12.2] DR: Great. That sounds like great advice. There's no substitute for doing, so go out and do it, and just get it done. I like that. Thank you, Andy, very much for joining me on Software Engineering Daily and sharing your insight and passion around something of great importance.

[0:53:26.3] AB: It was a pleasure being here. Thank you so much for having me.

[END OF INTERVIEW]

[0:53:33.4] JM: Thanks to Symphono for sponsoring Software Engineering Daily. Symphono is a custom engineering shop where senior engineers tackle big tech challenges while learning from each other. Check it out at symphono.com/sedaily. Thanks again Symphono.

[END]