EPISODE 725

[INTRODUCTION]

[00:00:00] JM: Building software was simplified by cloud providers. With the cloud, it became much easier to deploy a server, to spin up a database and to scale an application. Cloud providers like Amazon Web Services gave developers access to these infrastructure primitives, like storage and compute. On top of those primitives, numerous API companies have been built. An API company offers a more specific set of services. Twilio offers SMS text messaging API services. Stripe offers payment API services, and these APIs give developers another level of tooling to build software out of.

In addition to the cloud infrastructure level of databases and servers, developers can now think of entire applications in terms of APIs and the number of APIs is growing rapidly, from business services such as booking a flight, to machine learning models, like image classification, the API economy has given developers a huge catalog of tools. Since developers have this additional leverage, software can be built with smaller teams. The codebase can also be smaller, because you're just making simple API calls to complex APIs.

But one area where the complexity is growing for the average developer is the number of APIs that need to be managed. For each API, there's a different system of integrating the API into your application. Different API providers also have different levels of reliability. There's another area of difficulty in the world of APIs, and that is discoverability. If I don't know about a flight search API, I'm never going to think of what applications I could build on top of that. There are APIs for generating memes, APIs for easily querying what music is trending across the world. You could combine those two together and have memes generated on-the-fly as music trends change across the world.

There are lots of opportunities to blend APIs together and build products on top of that. RapidAPI is a marketplace for APIs. RapidAPI includes search and discovery features for the wide variety of different APIs that can be found across the internet. RapidAPI is also a system for integrating with multiple APIs through its API management system.

Iddo Gino is the CEO and founder of RapidAPI and he joins the show to discuss the motivation for creating an API marketplace as well as the engineering behind RapidAPI. Before we get started with today's episode, I want to mention that we are looking for journalists. We're also looking for an entrepreneur in residence and we're looking for many other roles. If you go to softwareengineeringdaily.com/jobs, you can find a list of what we are hiring for. If you think you'd be a fit and you want to work with us, please send us your job application.

With that, let's get on with the show.

[SPONSOR MESSAGE]

[00:03:09] JM: Kubernetes can be difficult. Container networking, storage, disaster recovery, these are issues that you would rather not have to figure out alone. Mesosphere's Kubernetes-as-a-service provides single click Kubernetes deployment with simple management, security features and high availability to make your Kubernetes deployments easy. You can find out more about Mesosphere's Kubernetes-as-a-service by going to softwareengineeringdaily.com/mesosphere.

Mesosphere's Kubernetes-as-a-service heals itself when it detects a problem with the state of the cluster. So you don't have to worry about your cluster going down, and they make it easy to install monitoring and logging and other tooling alongside your Kubernetes cluster. With one click install, there's additional tooling like Prometheus, Linkerd, Jenkins and any of the services in the service catalog. Mesosphere is built to make multi-cloud, hybrid-cloud and edge computing easier.

To find out how Mesosphere's Kubernetes-as-a-service can help you easily deploy Kubernetes, you can check out softwareengineeringdaily.com/mesosphere, and it would support Software Engineering Daily as well.

One reason I am a big fan of Mesosphere is that one of the founders, Ben Hindman, is one of the first people I interviewed about software engineering back when I was a host on Software Engineering Radio, and he was so good and so generous with his explanations of various distributed systems concepts, and this was back four or five years ago when some of the

applied distributed systems material was a little more scant in the marketplace. It was harder to find information about distributed systems in production, and he was one of the people that was evangelizing it and talking about it and obviously building it in Apache Mesos. So I'm really happy to have Mesosphere as a sponsor, and if you want to check out Mesosphere and support Software Engineering Daily, go to softwareengineeringdaily.com/mesosphere.

[INTERVIEW]

[00:05:28] JM: Iddo Gino, you are the founder and CEO of RapidAPI. Welcome to Software Engineering Daily.

[00:05:32] IG: Hey! Yeah, thanks for having me.

[00:05:34] JM: You're the creator of RapidAPI. Explain what RapidAPI does.

[00:05:39] IG: RapidAPI is an API hub, and what that means is we help developers, A, discover APIs so they can come on to Rapid, search for any function that they need in their app, like sending a text message or sending an email or encoding an image or doing some computer vision, seal the APIs that support that and actually test and connect through them to our platform.

[00:05:57] JM: You started RapidAPI in 2014. Sometime in the last few years, I started to hear the term API economy, and I think RapidAPI fits into that buzz-wordy term. How would you describe the API economy?

[00:06:14] IG: Yeah. The way we think about APIs and the broader API economy, is really is this disintermediation between creating the applications and the software actually used by users and then creating the infrastructure required to power those applications. I'll explain what that means. If you think about it, even if you build a simple app like a ridesharing app, like Lyft or Uber, you actually need a lot of separate moving pieces to work together to make that happen. So you need a map and you need push notifications and you need text messaging to let the driver know where to go. You need payment processing to actually charge the customer for that.

You need to do background checks for the drivers, and there are actually a lot of different pieces.

In the past when you went to develop an app like that, you'd actually need to go ahead and build and later maintain all of those separate pieces. A lot of the work and money that would have been invested in building that app would have actually gone into building those individual components. What's happening with APIs now is there're actually a huge market and a huge economy of companies providing those as a service. So if you want to do payment for instance you use Stripe, and for text messaging you can use Twilio, and you'd later send an email of the invoice, you can use SendGrid. To show the map and the application, you can use Google Maps or HERE. Then instead of focusing on building all these separate pieces, you can just kind of choose the best reads and paste them together to build an application.

[00:07:33] JM: In a marketplace for physical goods, you have ecommerce companies like Amazon. How does a marketplace for APIs differ from one for physical goods?

[00:07:45] IG: Yeah, and I guess this also goes back to why we figured there needs to be a marketplace or a hub for APIs. When we started seeing developers relying more heavily in APIs, we realized that there were a few challenges around connecting to them. The first one is actually just finding what APIs what you want connect to. Even for SMS, for instance, which sounds pretty simple, there are actually a lot of different companies providing that service and there are all very different depending on what geography you're working, what kind of messages you're working in, if you care more about price or about deliverability and so forth.

Discovering the right APIs is actually a challenge, and when we look to that, we actually realize that the same thing exists with ecommerce and physical goods, because you go on Amazon and you know that you need a TV, for instance, but you don't necessarily know what TVs are out there and you don't know which exact model do you want to get and you need a search engine or a platform where you can find and make that decision.

[00:08:39] JM: The cloud providers themselves, they have marketplaces for disintegration points with third-party services. AWS has a marketplace. Azure has a marketplace. How do these differ from an API marketplace like RapidAPI?

[00:08:56] IG: Yeah, I think that a lot of the cloud providers have recently gone and identified a trend too of partnering with third-party providers. Then when we look at Rapid – And a lot of them have a few dozen or a few best and little over a hundred integrations. When we look at Rapid, we really try to the un-opinionated. We bring in currently a catalog of a little over 8,000 APIs that are available on the platform and we actually partner with everyone. Unlike some of the cloud provider marketplaces, we actually bring APIs from all the different cloud providers and then from all the different players in the market to really bring as much choice as possible and let developers using the platform choose which APIs they want to use.

[00:09:34] JM: It's useful to have this central aggregated marketplace for all these different APIs. As you said, one reason for that is that there are some API verticals where you have so many different players. For example, text messaging, or text analysis. When I was preparing for the interview, I looked at text analysis as one of the things to search – I had to search for it on RapidAPI, and I found 10 APIs or something like that, maybe more, that were doing text analysis or topic modeling. I'm sure that they vary in quality, because this is not a straightforward thing to implement. What are the kinds of tradeoffs that you see? Are there tradeoffs in like price versus quality? Are the more expensive APIs necessarily better? How are the different tradeoffs explored by different API providers?

[00:10:30] IG: Yeah, and I think that when it comes to discovering and choosing APIs, you actually kind of hit the nail on the head, because – Again, looking at your example of text analysis, there are multiple dozens of API doing that on the platform, but then it's not – Because a lot of people think – And that was a lot of people's thesis when that API economy was created, that for every vertical, there's going to be just one player who is best in breed. Everyone is using them and that's the end of the story.

But the reality is that it's actually a lot more complicated and nuanced than that. For text analysis, you could want to be extracting tags out of tasks. That's one use case. You could want to be summarizing tasks. The actual corpus or taxon you're using could be news articles or it could be chat messages or it could be books. Then it can be in English or in Japanese or in Hindu or in any other language. Depending on that use case, so what you're trying to extract,

from what kind of taxon and what language, it would actually be a different API that would be best to read for what you're doing.

That API that you'll choose will actually depend a lot on your actual use case, which means that you need to go there and actually test a bunch of different APIs with the exact kind of text or exact kind of inputs that you would be using in production and then use that to make the decision about which API works best.

[00:11:47] JM: I called the API economy a little bit of a buzzword earlier, but it's a very real thing, and that's because so much of the application development that people do these days are integrating with APIs, connecting to the APIs, wiring different APIs together to build really rich applications with less upfront work than you had to do prior to this proliferation of APIs.

I think RapidAPI fits into this not just in the search and discovery part, but there's also the central point of management. Once a developer connects to RapidAPI, they might want to use several different APIs. You want to use Twilio, and Stripe, and SendGrid and something for text analysis. The API providers, like Stripe, make their API really easy to use, but there are still some time spent onboarding and integrating it. If you talk about that across all those different APIs, Twilio and Stripe and SendGrid, you're talking about a lot of different accounts that you're managing, a lot of different integrations you have to do.

If you use RapidAPI, you can have some easier management experience and some easier monitoring experiences. What's the advantage of using RapidAPI in terms of API management?

[00:13:12] IG: Yeah, and the way that we view it is – And we always call that process, like our nickname of that process of connecting to all these different APIs without using a centralized marketplace, is we call it plumbing. What you end up seeing is even if all the APIs that a user very well-documented and all use something that resembles a standard. A lot of APIs are not, but even if we assume that best case scenario, what you still end up seeing is every API uses a different data format. So it could be using XML, or YAML, or JSON or something [inaudible 00:13:44]. It could be using Raspberry. It could be using [inaudible 00:13:47]. It could be using something that's not really either. It could be using a different way to authenticate users. So it could be using Basic Auth or oAuth2 or a token or an API key and it could be requiring an

encryption and for you to actually sign the request that you're sending or not depending on – Again, depending on the API.

At the end of all that, it will have a different dashboard for actually managing it. It will require you to have a separate account. It will require you to have separate API keys. It will send you a different invoice at the end of the month. When you think about it, as you said, in the context of now connecting to 10, or 20, or 30 different APIs, suddenly a lot of the work that goes into it is in setting up all that plumbing and connecting to all of the different APIs that you're using.

When we figure out that we want to create an API marketplace we figured, APIs are really empowering developers to create software. But on the other hand, all these unnecessary overhead of doing the plumbing and doing the connection work, and we want to make that easier. That's where I see the most value with Rapid. So being able to instantly see an API, test it to make sure that it works well in the actual production use case and then just copy a code snippet, and they all look the same, so that you can easily integrate that into your application.

[00:14:55] JM: The other interesting use case for an API marketplace is kind of just exploring and seeing what's there and finding new things that you didn't know existed. I've looked through API marketplaces. I looked through other aggregators of different API patterns and I sometimes just get inspiration by looking at these tools and seeing just how high-level some of these APIs are getting. What are some of the APIs that you've seen on RapidAPI that people might not be aware of? Things that have surprised you or impressed you or inspired you?

[00:15:32] IG: Yeah, I think that – This is actually – What you're saying is super interesting, because we've seen something similar from our perspective as well where we're kind of starting to see more business decision makers and product managers actually leveraging RapidAPI as well to start to understand what capabilities exists in the marketplace and then use that to ideate and bring new solutions.

You can see there are now APIs for anything. Just some of the examples that we've seen in the last few weeks, you can actually create insurance quotes and buy insurance via an API, which may not sound as exciting to some but is actually I think pretty cool. There's an API to know how much snow there is on the mountains for skiing season. There's an API that allows you to

search for flights. There's an API that allows you to send messages to users across multiple channels. There really is between the 8,000 APIs that we have on the platform and about 30,000 APIs that exist publicly in general, we actually do see an API for everything now.

[00:16:31] JM: Yeah. I mean, there's that classic question of if you want to start a startup, what's your best strategy for thinking of an idea? I think a modern strategy that has developed now, is you can like look through APIs that are out there and just like what APIs can you mash together and find an arbitrage? There're probably a lot of combinations out there.

[00:16:53] IG: Yeah, and those API mash-ups are so cool, because you take data from one, you pass it to another and suddenly you have a whole new use case that no one has ever thought about before.

[00:17:02] JM: Definitely. Let's talk about the engineering of RapidAPI. Can you describe the first version of building the API marketplace?

[00:17:10] IG: Yeah. I guess when we started – I don't even know if you could have called it an API marketplace back then. We basically started with a GitHub repository that contain a lot of APIs that we thought were really useful and then standardized kind of code snippets that connect to the – It was like one SDK that can connect to all of them, and we had that for Node.js originally. You'd be able to pull that and then you'd still need to go to different API providers, sign up, use an API key, but suddenly you had one SDK that connects to all of them.

[00:17:38] JM: When did that transform into the API marketplace business?

[00:17:42] IG: Yeah. We launched that on GitHub earlier in 2015, and for most of that year we just kept pushing data and telling our friends about it, seeing more people rely on that. Around the middle of the year, what we started seeing, which we thought was really interesting, is not only where a lot of people fetching data on NPM and embedding it in their applications and starting to use it to connect to APIs. But we actually started seeing API providers and people who create APIs making poll requests to that repo and adding support for their own APIs. When we started, we had like 20 APIs. By middle of the year we had about 200.

[00:18:18] JM: So did it become a business instantly? That sounds more like kind of an index and a little bit of a tool, but not exactly a business quite yet.

[00:18:28] IG: Yeah. I think it was around that point when we started seeing a lot of people consuming APIs through the platform. But then also actually seeing that API providers and people creating APIs realize what the potential was and we're actually starting to add their APIs when we figured out like, "Hey, there is an opportunity to actually create a business here," and there's still a lot of work that we can do to improve the experience of discovering and connecting to APIs. So actually doing the API key and account provisioning in one place, consolidating the billing in one place and brining all the API usage and analytics into one platform. Towards the end of 2015, we decided to actually go and spend some time and do a lot of engineering to turn that into an actual marketplace. Around that time we actually went ahead and launched rapidapi.com.

[SPONSOR MESSAGE]

[00:19:21] JM: We are running an experiment to find out if Software Engineering Dailydaily listeners are above average engineers. At triplebyte.com/sedaily, you can take a quiz to help us gather data. I took the quiz and it covered a wide range of topics; general programming ability, a little security, a little system design. It was a nice short test to measure how my practical engineering skills have changed since I started this podcast. I will admit that though I've gotten better at talking about software engineering, I have definitely gotten worse at actually writing code and doing software engineering myself.

But if you want to check out that quiz yourself, you can help us gather data and take that quiz at triplebyte.com/sedaily. We have been running this experiment for a few weeks and I'm happy to report that Software Engineering Daily listeners are absolutely crushing it so far. Triplebyte has told me that everyone who has taken the test on average is three times more likely to be in their top bracket of quiz course.

If you're looking for a job, Triplebyte is a great place to start your search. It fast tracks you at hundreds of top tech companies. Triplebyte takes engineers seriously and does not waste their time, which is what I try to do with Software Engineering Daily myself, and I recommend

checking out triplebyte.com/sedaily. That's T-R-I-P-L-E-B-Y-T-E.com/sedaily. Triplebyte, byte as in 8 bits.

Thanks to Triplebyte for being a sponsor of Software Engineering Daily. We appreciate it.

[INTERVIEW CONTINUED]

[00:21:19] JM: If I'm a developer and I'm looking at using a service like this, RapidAPI, as my single integration point for APIs, I'm going to be pretty concerned if all of my API calls have to go through RapidAPI, and RapidAPI becomes this single point of failure for all of my APIs. Is that something that you had to consider in the architecture?

[00:21:43] IG: Yeah. For us, a lot of it was understanding the we are kind of becoming a very critical piece of infrastructure for a lot of apps that are relying on us. There are a lot of decisions that we made to make sure that, first of all, the request go through and they go through as fast as possible and as reliably as possible. Only then do we actually log in all the analytics and verify the billing and know all the things in the background to make sure – Because the thing that we never want to do is cause any application to crash. On the flipside of that, we also add a lot of capabilities that actually take away the risk from using so many APIs.

For instance, if you connect to an API through Rapid and it goes down, you can actually setup a backup policy to switch over to a different API and you can do that through the platform, again, because we intercept all the requests and we're already kind of setup and connected to all the different API. We can reroute the request between them pretty easily. We can basically define a rule that says, "Well, if you get a 500 error from one API, just reroute the request to another to make sure that the application still works." You introduce Rapid as a dependency, but then you can alleviate a lot of the dependencies on other API providers. I think there's some sort of give and take around that.

[00:22:52] JM: That's pretty cool. So you are a dependency though? If RapidAPI were to go down, there would be some issues with people who have integrated Twilio using RapidAPI, for example.

[00:23:06] IG: Yeah, and that's also why we deploy – We actually take a lot of care to, A, deploy across a lot of regions. We deploy the actual component that proxies the API requests on multiple AWS regions as well as Azure and GCP regions. Then we keep them all independent. Each one of those regions has their own copy of the database that we sync in the background and has its own – And is basically self-contained. Even if it gets disconnected from everything else, it can still work and function as normal. That means that even if one or two or multiple regions drop, we can still service all the applications.

[00:23:39] JM: That's interesting. Can you take me through an end-to-end call and maybe some failover cases? If I integrate a Twilio through RapidAPI and a user is going to get a text message from Twilio via RapidAPI. Explain how that API call is going to be processed end-to-end.

[00:23:59] IG: Yeah. You're going to make a request to, let's say, twilio.p.rapidapi.com, which is our production environment. That when you make that request, that will actually reach our global DNS setting, which will route you to the closest data center that we deploy on. Hopefully if you're on AWS or GCP or Azure, where it actually be the same data center where you're sitting. So the routing should be pretty quick.

From there on, we get the request and we actually run that proxy component using nginx and writing a lot of Lua code on top of that. We will get that request and then we'll verify a couple of things. First of all, you'll pass your RapidAPI API key. We will actually hit that local database to make sure that it's a valid API. Can you actually have [inaudible 00:24:44] with RapidAPI? Then we'll also verify that you're subscribed to the API. So make sure that you've actually went on and subscribed to Twilio, because it's a Beta API and you need to agree to that.

Lastly, we would verify that the endpoint that you're trying to use is within the plan that you're subscribe to and that you actually have enough credit remaining to use that API endpoint or you're on a pay as you go plan and then we just let it pass through. Once we verified all of those things, we route the request to the underlying Twilio API and let them actually handle their request. In that sense, send you the text message. Then we get the reply from them and forward it back to your application.

In that scenario, we actually do all of that processing, so before the request and then routing it back to you in a little under 10 milliseconds. We do that pretty quickly to make sure that there's no noticeable latency added on top of the API. Then after that API request is sent, the response is basically sent back to the application and the application can keep forward. Asynchronously, we log that API request so that we can then show you in the dashboard and show you the analytics around making that API request.

[00:25:50] JM: I think that 10 millisecond lag time, or that sub-10 millisecond lag time, that's like the – Is that the threshold for what people notice? I think I remember reading that at some point.

[00:26:00] IG: Yeah. Think for that from a user perspective, you would want to keep most of those requests under, I think, 100 to 200 milliseconds and that's where most APIs that we work with are. Of course, there are ones that are very fast or ones where you're uploading files or doing some more massive queries and then they're a little slower. But that's like the – I would say the min. Then we try to have our min at around 5 millisecond and never go above 10 just so that we can keep that as a pretty unnoticeable additional latency.

[00:26:34] JM: This multi-datacenter fail – Or multi-cloud provider failover situation, I could imagine some potential consistency issues if – I don't know, for example, if you have a DNS issue, or there's something wrong with the API provider itself. Have you had any interesting consistency issues when dealing with the multiple cloud provider infrastructure setup you have?

[00:26:58] IG: Yeah. For us, it's always, I guess, a give or a take or a balancing act between, on the one hand, wanting to be very reliable and have very high availability and make sure that when we get a request, we forward it to the API no matter what. But on the other hand, if you hit one proxy and you finish your quota, but then you hit another proxy before that data syncs, then you might actually be able to go a little over quota.

But when we're faced with those choices, we always go for whatever option we'll min that where the most highly available and the most responsive to our clients, because we understand that – Again, as you pointed out earlier in the conversation, RapidAPI can become a single point of failure and we want to make sure that we're very reliable first and everything else comes second.

[00:27:48] JM: All right. Are there any engineering decisions you've made in building the infrastructure that might be surprising to the people listening?

[00:27:56] IG: I'm trying to think if anything would be surprising. I think that one thing that hopefully wouldn't be surprising is that we actually rely on a lot of APIs ourselves. One of our IDMs in the company, if someone has built it, let's use that rather than build it ourselves. But then I think beyond that, it really, it's a lot thinking around using things off-the-shelf. For instance, we process billions API requests every month and then we show all of our users and developers a lot of information and analytics around how they're consuming those APIs. We fully rely on the Amazon Kinesis platform to run all these analytics. Again, that's like an off-the-shelf platform that we're using as part of that approach.

[00:28:35] JM: Yeah. It kind of surprises me when people run their own Kafka, like run their own Kafka on AWS instead of going with something like Kinesis. I think there's still a little bit of residual, not built here or biased towards open source solutions versus cloud provider solutions that save time. It's nice to hear that you are going with the higher leverage approach of that. Because it's kind of a buzzwordy question.

But how are you evaluating decisions around how you run your core infrastructure. I talked to a company a while ago that was real estate AI company called Skyline, and they had built their entire infrastructure, almost their entire infrastructure on serverless APIs and backend as a service and stuff and they were able to move so fast. They were doing very complicated things, like complicated machine learning jobs and things like that, resource-intensive jobs. But they were just doing it with the latest tools. I'm not sure they even managed their own Kubernetes cluster or something. But how are you getting leverage out of some of these newer APIs and backend as a service tools?

[00:29:46] IG: Yeah. Our approach to this is always – We will always strive to choose the best of breed solution. If someone else is doing it and it's been proven, we'll take it. At the same time, we do try to stay away from – Because I think that the pace of development in technology is very fast now and there are also a lot of downsides or risks to always going with the most recent

technology for the sake of going with the most recent technology and you pay a lot of penalties for being early on that learning an adoption curve.

We try to definitely not to be laggers with those things and we have recently adopted a whole host of new technologies. We use React for all of our frontend. We use GraphQL for all of our internal APIs and rendering the pages. We do use actually a lot of serverless and lambda. We do depend a lot of newer technologies. But we try to adapt them when we have enough confidence that it's been proven, it's been tried out. We can have at least two or three people on the team who've actually had experience bringing them to production and it means that we can actually do it in a reliable way. Because, again, we try to be somewhat risk-averse on a startup scale with the technologies that we're using to make sure that our solution is always up and reliable.

[00:30:57] JM: Yeah, lambda seems perfect for RapidAPI, because it sounds like you have so much glue code to write and lambda's kind of perfect for that.

[00:31:06] IG: Yeah, a lot of transforming API requests and then sticking together different APIs. We do that all using lambda. Then the thing we like the most about it is not having to do a lot of planning in terms of how many resources we need. For instance, for the billing service, we can only have the end of the month and we need to generate 10,000 invoices for API consumption, and that's actually a very resource-intensive task and they need to spun up a lot of machines and then spunning them down is a lot of overhead and can cause a lot of latency, where lambda just solves that perfectly for us.

[00:31:39] JM: Are there any other scalability issues you've had that might have some interesting lessons people could take away?

[00:31:46] IG: Yeah. I think for us, especially everything that's in the critical path line of forwarding requests, we've pretty much hit up every weird scalability issue that there is starting from actually limits that we've found on some cloud machines and in certain cloud providers on how many packets per second they can send over the network, and we just ran into those when we started seeing packets drop in the network.

We ran into a lot of those weird quirks, and I think that this also instilled with us this approach internally of actually doing load testing and stress testing on everything that we build before deployment. We'll try to find the peak of what normal consumption looks like and then test for 10X that just to make sure that everything is actually built to handle scale.

[00:32:31] JM: You mentioned GraphQL there and I know you had a recent redesign of your website and how some of the backend engineering works and the frontend interactions between the backend. Can you talk about how you used GraphQL in the redesign of your platform?

[00:32:51] IG: Yeah. I guess to put that in context, I'll take step back and talk a little bit about our general architecture. So pretty earlier on, we decided to go with a heavily micro-serviced architecture. So we run everything in little services. One of the benefits that we like about that to kind of touch on your previous question is it allows us to more readily adopt the different in your technology where we see fit. One service could be running on lambda and serverless and another one could be running on actual machines. Then some stuff we have running with Dockers and Kubernetes. We can actually be pretty flexible around how we adopt technologies.

But then we realized that we're writing on the frontend a lot of glue code, sitting together data from all of these different microservices. When we went into that redesign – I guess another thing that I can put out there just as far as context goes, is when we started building the version one of the system. We had about 5 services that we're using. Now we have closer to 20 services that the frontend is consuming data from. We realized that because of that increasing number of services, we're doing a lot of stitch code. GraphQL kind of landed itself perfectly for cleaning up a lot of the frontend coding, stitching that more easily. When we realized that we had that challenge, that's when we started – Or made the decision to rely on GraphQL.

[00:34:07] JM: How does your GraphQL server run? Is that running on a Kubernetes cluster?

[00:34:13] IG: Basically, the way we set it up is we have all the different microservices. Then we have what we call the aggregator sitting in front of them, which actually connects to all the different microservices and defines the GraphQL models on top of them. Those we actually run straight on EC2 machine. So we don't use Kubernetes or Docker there.

[00:34:33] JM: What's the reasoning behind that?

[00:34:35] IG: I guess, for us, we actually started by running everything on Kubernetes. But we figured out that it added a lot of complexity that was unnecessary to the system. A lot of the advantages that we saw around it initially around utilizing machines better and introducing cost savings with scale just didn't make sense, because now we're suddenly running 12 machines for just a single service and all of them are at pretty high utilization.

But on the other hand, being able to actually go into the machine, debug it, see the actual logs had a lot of advantages. Then we also had a lot more experience in our devops team of just running those machines straight. So spunning up straight EC2 machines. We just switched back to using that older style of deployment.

[00:35:15] JM: Are there any tradeoffs to that? I guess cost-efficiency perhaps?

[00:35:20] IG: Yeah, I think for services that are not being very highly utilized, from a cost perspective, it makes more sense to do it on Kubernetes, because you can share them more easily. But when we started seeing that we're using pretty big machines and every service was actually having high-utilization of multiple of these machines, then those – It'd probably still make more sense from a cost perspective to do Kubernetes, but the delta became pretty small. That's when we switched back.

[00:35:46] JM: Your team is distributed. I think you have an office in San Francisco, in Kiev and in Israel. Is that right?

[00:35:57] IG: Yeah, that is correct.

[00:35:58] JM: Tell me about managing those three teams and how you think about maintaining consistent company culture or maybe you don't need consistent company culture if those offices are doing different things. What's your philosophy on remote work and distributed teams?

[00:36:17] IG: Yeah. I guess, for us, the way we divide it is we do all of our product, marketing and sales functions out of the San Francisco office. We then do a lot of the core engineering out

of our Tel Aviv office, which is our biggest one right now with about 25 people. Then in Kiev, we do some more auxiliary engineering and engineering support tasks.

So it's really divided mostly between those three locations. Then a lot of the reason that we chose to do that, we knew that we – Or we figured out that we needed to have some presence in the Bay Area, mostly because a lot of the API partners that we are working with are based out here. So we wanted to be close to them. A lot of the developers that we're utilizing RapidAPI were also based out in the Bay Area. So we wanted to be close to them as well.

Then a lot of the conferences and bigger events are out here. So we found our self on a plane a lot of time anyways. So we figured out that we wanted to have some big presence in the Silicon Valley. But on the other hand, we had a lot of good engineers that we knew back in Tel Aviv and the market there is slightly less competitive than the Bay Area market that we felt it still made sense to keep engineering there. So that originally is what led to that split.

Then as Tel Aviv became more and more competitive for engineering talent, we ended up opening another office in Kiev just to enable us to hire faster as we scaled. So that was a little over a year ago. Then we actually do really value having a consistent company culture and doing a lot of synchronization on that part between the different offices.

One of the things that we do to keep that is, A, we have weekly all-hands. So bringing the entire company together to sync on what the different priorities are and what everybody is working on. The second thing is about every 9 months we try to have an all-company offsite or all-hands. The last one was two months ago where we bring everyone together, sit in one room and can actually, A, on a more formal bases share where we see the company is going and what the different directions are. But also just spend time together and get to know people, because I think that actually creates very – Or much more productive work relationships afterwards.

[00:38:18] JM: Do you see anything distinguishing about the engineering in Silicon Valley versus these places, like Kiev or – Because Kiev and Israel, these are two places that have superbly talented engineers. I think the world of startups is trying to figure out to what degree you can hire people in places like Kiev or Israel or in India. There's plenty of super good engineers in India, and it's not just that they're really good engineers. It's like the knowledge

around how to think creatively as an engineer, how to think outside the box, how to think like a business-driven engineer. These are things that used to be relegated to Silicon Valley engineers. Now that knowledge has become much more dispersed. It's harder to know what exactly is the advantage of hiring engineers in Silicon Valley. Do you have a perspective on that?

[00:39:15] IG: Yeah. I think there's also, again, a balancing act and there are pros and cons to both approaches. To having all the engineering in one place or in Silicon Valley and then to having all of the engineering or some of the engineering in different locations. But I think that before diving into that analysis, the bottom line is that there's just not enough engineers in the Bay Area and a lot of companies are realizing that now. So most recently you've seen – I mean, even in Seattle and other locations you see Amazon with their HQ2 an apparently two and three initiative, because they realize that they needed more global presence. They actually also have an office in Tel Aviv.

Then you see a lot of the Bay Area based companies, so Facebook, Google. You see, from Seattle again, Microsoft. All of them are opening offices around the world, because they realize that there's a lot of really good talent out there and it doesn't make sense to limit yourself to just the Bay Area geography, because that doesn't allow you to leverage all of that talent. You see a lot of – I've actually seen a lot of companies going to Canada and opening offices there and a lot of companies have seen a lot of success with that. I know that Eastern Europe, and Kiev, and Belarus and all that area have been very prolific as well. Of course, Israel, with a lot of companies opening R&D centers there. I think that the reality is that there's just software eating the world. We need a lot of people to write that software and we just don't have enough engineers physically in one place.

But then I think that there's a lot of – When you make that conscious decision earlier on as a company, to have a distributed team, there are a lot of cons that you need to understand, especially around how you synchronize culture and directives and objectives between the different members of the team and then a lot of the communication hurdles that you get with having a distributed team.

Especially when you're doing it between the West Coast and then Europe and you have a pretty hefty time difference between those locations. There's not a lot of time even for doing face-to-face or voice communication during the day.

[SPONSOR MESSAGE]

[00:41:13] JM: Software engineers can build their applications faster when they use higher level APIs and infrastructure tools. APIs for image recognition and natural language processing let you build AI-powered applications. Serverless functions let you quickly spin up cost-efficient, short-lived infrastructure. Container platforms let you scale your long-lived infrastructure for low-cost, but how do you get started with all of these amazing tools? What are the design patterns for building these new types of application backends?

IBM Developer is a hub for open source code, design patterns, articles and tutorials about how to build modern applications. IBM Developer is a community of developers learning how to build entire applications with AI, container, blockchains, serverless functions and anything else you might want to learn about.

Go to softwareengineeringdaily.com/ibm and join the IBM Developer community. Get ideas for how to solve a problem at your job. Get help with side projects or think about how new technology can be used to build a business.

At softwareengineeringdaily.com/ibm, you will find the resources and community who can help you level up and build whatever you imagine. Thanks to IBM Developer for being a sponsor of Software Engineering Daily and for putting together a useful set of resources about building new technology. I'm always a fan of people who build new technology, who build new applications, and this is a great resource for finding some design patterns and some new ways to build and leverage these technologies, like serverless and containers and artificial intelligence APIs.

Thanks again to IBM Developer.

[INTERVIEW CONTINUED]

[00:43:13] JM: You started RapidAPI when you were pretty young. You were a teenager and you were also a Thiel fellow, which is a grant of $100,000 from Peter Thiel. When I first read about the Thiel Fellowship, the perception was – Or at least that I got from what I read about, was that these are all prodigies. All of these kids that – It's like people under 23 who get $100,000 to not go to college, to explore something that they're passionate about. The perception you get from reading about this is that there are some genetic trait that allows Thiel fellows, such as yourself, to be uniquely brilliant. You're born with something.

But I've met a couple Thiel fellows, such as yourself, and the common trait that I see seems to be more about that these people spend a lot of time on the internet growing up. Basically they grew up reading Wikipedia, building some software, talking on social networks. From what I can tell, the Thiel fellows are people who have taken education to their own hands, because the internet offers this fire hose of education. Some people prefer that to a slower pace of a classroom, or they're better accustomed to learning on the internet.

Maybe there's some component of genetics. Maybe there's some component of parenting. But there's also this big component of just using the internet as a child in ways to educate yourself and create a fashion. What's been your experience as a Thiel fellow and do you have a perspective on what it says about modern education?

[00:44:54] IG: Yeah. I think that – You kind of alluded to that as well in the question. The internet is actually – Especially around software development, has actually done something phenomenal for people who like software and lack technology and want to be a part of that, where it actually democratized that community and enabled a lot of people to get into software engineering and software development and technology without a lot of resources required.

If you think about it, let's say you're really passionate about cars and you want to be building and designing cars, the bar for entering that is actually super high, because you need to have access to all the instruments. You need to buy engines. You need to have a workshop where you can — there are things that are actually pretty expensive and not a lot of people can do that out of the box. If you think about creating a piece of software nowadays, all you really need is a macro core or another computer and access to the internet and a text editor. The bar to actually

creating software is really low, and using the internet there's actually a lot of resources available online that enable you to learn that pretty quickly.

I think that – This is what I find the most interesting about technology, you just need to be passionate about it, because all the resources are there and there really isn't a bar to enter it. As long as you're intrigued by technology, want to get into it and you start looking online for the different resources, you can very easily become a very or an on-par software developer.

I think that I see that with a class of Thiel fellows that I met in later classes as well. You'd think that everyone would be from a very specific background, but the reality is that there are people from all over the country and all over the world who all just have a passion for technology and for wanting to pack things and build things and craft new pieces of technology and had access to a computer and the internet when they were growing up.

[00:46:38] JM: There's this weird way in which Peter Thiel has become an oracle of Silicon Valley and yet there's still taboos around Peter Thiel, like talking about Peter Thiel or being affiliated with him. His book, Zero to One, is probably the best book about modern company creation. I don't hear nearly as much public praise about it as I do about books that have less controversial authors. It seems like there is this love-hate relationship with Peter Thiel in Silicon Valley. Why is that?

[00:47:19] IG: I wouldn't necessarily be able to fully analyze controversy around Peter Thiel, and I also think that if you look more specifically around the fellowship, a lot of people – it kind of became this very polarizing thing, where people kind of perceive it as if you're associated with Peter Thiel or if you take the fellowship, then you hate college education and you hate formal education and everything that was built there. If you don't, then you love college education and you believe in the old ways.

I don't think that that's what anybody is trying to say. I think that there is a more subtle message of you can go to college and learn a profession or learn something there and then work and be great at it, but it's not the only way or it's the not necessary de facto default way. I think that's the biggest message, but people took it to a very – This is also due to media and the social networks and echo chambers and everything that we see with the way media evolved. People

took it to that very polarizing point, which causes a lot of antagonism. I think that that also helped with some of the controversy.

[00:48:22] JM: I think it will eventually change, because you have these case studies such as yourself or Laura Deming, or Vitalik Buterin, or Kevin Lang was on the show a while ago. These people who have made very good use of their fellowship and built pretty important companies or technologies or ideas or they've become evangelists for technology. Laura Deming is kind of like an evangelist for biology now. It seems like a pretty uniformly positive pursuit, the fellowship.

In Zero to One he talks about this question that he always asks people, "What do you believe about the world that is true that nobody else believes?" Did he ever ask you that question?

[00:49:07] IG: Not in person actually, but it's definitely something that came up in a lot of the fellowship interviews.

[00:49:12] JM: Did you have an answer for it?

[00:49:13] IG: I don't know if I actually have or yet have crafted a precise answer to that.

[00:49:17] JM: All right. Coming back to the API world that you're living your day-to-day life in, how do you expect this market for APIs to change in the next five years?

[00:49:30] IG: Yeah. I think that the two big underlying trend is, A, we're going to see more public APIs cropping up. B, that same kind of two-layer or two-tier approach that we've seen created in the public marketplace, where you have people who create the basic Lego pieces, so the basic infrastructure in the form of APIs and then you have people taking APIs and turning them into a product. We would see the same thing happening within larger companies, where you'd have different teams creating the actual infrastructure in the company in the form of internal services and then other teams productizing those or using those APIs to create products that they can then deliver to customers.

I think that there's – When you kind of think of those two types of teams, there are very different requirements. On the one hand if you're building infrastructure, you need that to be very robust

and very fast and very reliable and there's a lot of nuanced work around that. But then if you're actually taking that infrastructure and productizing it, you need to be very fast to be able to A-B test and reply pretty quickly to customer demands and adapt the product. Then you also need to be very flexible, because the number of platforms that you need to run on is increasing all the time. You're building a product now and you want to be on the web, you want to have a desktop app, you want to have a mobile app, you want to be [inaudible 00:50:46], you want to be on a Messenger. You want to be able to be on voice. You want to be everywhere, and then wearables and all the different platforms. You need to actually be pretty fast and reactive. I think that that's where we're also going to see that two-tier approach happening in a lot of companies.

[00:51:00] JM: That's an inspiring vision, the idea that companies, it will become more and more the norm that if you're a company like a Spotify, for example, you've got these internal – These APIs that start as internal things, like what's at the top of the charts today. You want the top of the chart's API, or what was the most listened to song yesterday API. You want it as an internal API. But overtime, you start to realize, "Oh, there's actually a huge demand for this in the broader API economy. We need to externalize this service.

[00:51:35] IG: Yeah. I think that if you think about – Let's take that example of music. There would be the company that has the biggest dataset of music and has all the actual music files and has the CDN power to deliver them and has all the commercial relationships with the different studios to have the rights to that music. But then you want that music to be in a lot of places. One company could use that public API to build a platform for coffee shops that lets them play music tailored to the taste of their customers. You'd have someone else maybe embedding that into a shuffle player that's built into a ski helmet and that people can use [inaudible 00:52:11] down the mountains. This is just a personal dream of mine.

Then you will have someone else embedding that into inflight entertainment systems and airplanes so that people can listen to music on the plane. So you'll just – You'll have one company that's building the infrastructure and a lot of different companies that are better at very quickly creating products that use that infrastructure to deliver it to customers.

[00:52:31] JM: I've done a couple of shows with Auren Hoffman from SafeGraph, which is a company that's trying to make access to datasets more accessible, and this turns out to be

something that's really hard and it's kind of strange how proliferate web services have become. But the access to datasets is still really constrained. You can imagine a world in which there's more and more of an API economy into datasets. Have you thought about how that might affect your business, the dataset as a service business?

[00:53:07] IG: Yeah, especially now with the proliferation of machine learning and artificial intelligence and a lot of people building and constructing models, it actually requires a lot more access to datasets. I think that there's a very similar problem there. A lot of data exists online, but, A, there's a challenge around finding it. B, it has a lot of different formats and cleansing it and merging it and doing all the transformations and all the plumbing around that. Especially when you're talking about big data, so you can't just open it in like a text editor and change it. That also becomes a huge challenge.

As we kind of talked about earlier in the hour with APIs, there's also this interesting phenomena of once you can actually discover what data is available, you can then use that to ideate and build new models. The same as we discussed with APIs, whereby seeing what APIs exist in the marketplace, you can come up with product ideas.

I think that one company that proved that very well is – Or the potential around it very well is CAGL. Again, just by taking a bunch of different datasets, putting them in one place and letting people hack around with them. I think that there's a lot of similarities between what we're doing with APIs to what companies like [inaudible 00:54:10], like CAGL are doing with datasets.

Actually it's interesting, because we're also seeing a lot of marriage between those machine learning problems and those datasets that are being created and then using APIs to enrich them. For instance, one example that we've seen is there's this awesome dataset by Recruit in Japan around visitors to their shops and they wanted to create a model that predicts when people will visit the shops. You had the dataset, and the dataset was basically date, shop location, number of people visited, and that's it.

You can use that create a model, but that model would be missing a lot of data. For instance, you see a date. You can't really know if it's a weekend or a weekday. Maybe it's a holiday, because as you can assume, that will probably have impact on how many people visiting the

shop. You can't know what the weather was like that day, because if it was a superhot day, or if there was as snowstorm, that will maybe impact how many people visit the store.

So there's a lot of external kind of data to the original dataset that can actually help you enrich the data and make the model a lot more accurate. We now see a lot of people using APIs to enrich their datasets and make them a lot more full to then create a lot more accurate models.

[00:55:22] JM: Right. There are some aggregation effect there, where the better your model – If you're doing a machine learning model as a service essentially, the better your model gets, the more people are going to use it and the more training data they're going to be giving you. Especially if you can build some kind of system where as people are making API requests, those API requests are also serving as training data for improving those models.

[00:55:52] IG: Yeah. I think that letting people kind of track what they're using and giving access to a lot of that live data. What's important about APIs too, it's not just being able to get the historic dataset. Let's say back to that store visiting example, you now know that whether impacts – If it's a snowy day, people are 50% less likely to go and eat ramen, or maybe 50% more likely to eat ramen if it's actually snowing outside. Then you now have that data or that insight, but you still need to actually use that now to make predictions on a live basis. You still need to have access to that data live, and that's where APIs actually come into play. Because you can then use that same API that you use to enrich the data, but embed it into the production or into the runtime environment to keep enriching the data as it's coming in and make life predictions.

[00:56:41] JM: With Amazon, we've seen that a marketplace that has a lot of liquidity can be used as a platform to build lots of additional products, basically limitless products. Do you have ideas for what you're going to build on top of the RapidAPI marketplace as you continue to build out this? It seems like the bottom layer of something. I don't know quite what your vision is, and I'm sure you can't disclose exactly what your vision for the future is. But can you give me some taste of your vision for what's going to be built on top of the RapidAPI marketplace?

[00:57:17] IG: Yeah. From our perspective, the goal of Rapid is to really help and empower developers to discover the right APIs and then to connect to as many of them to make their

replications better. A lot of the work that we want to build on top of Rapid is, A, giving developers better tool when they need to consume a bunch of different APIs and consolidate them and merge and join the data between them. So giving better tools for consuming and merging the different APIs, but then also giving tools and helping people creating APIs and making that API creation more easy and making it easier to expose and build that into a public API.

I think that really working both sides of the marketplace that way. So helping developers consume more APIs and helping providers expose more APIs and do that more easily. That's where at least for the next couple of years a lot of our work is going to be focused.

[00:58:07] JM: Iddo Gino, thank you for coming on Software Engineering Daily. It's been really fun talking to you.

[00:58:10] IG: Thank you very much. It's been a great pleasure to be on the show.

[END OF INTERVIEW]

[00:58:16] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source and free to use, and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plug-ins. Use the value stream map to visualize your end-to-end workflow, and if you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on-the-fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations. You can check it out for yourself at gocd.org/sedaily.

Thank you so much to ThoughtWorks for being a longtime sponsor of Software Engineering Daily. We are proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]