EPISODE 836

[INTRODUCTION]

[00:00:00] JM: Gaming is becoming mainstream. Popular multiplayer games, such as Fortnite and Minecraft present players with a massive virtual world to explore, to build and compete within. Turn-based games such as Hearthstone and Magic are breeding a new generation of board game and card game aficionados. Social media networks, like Twitch and YouTube, have turned gaming into a voyeuristic sport that is outcompeting many physical sports games for attention.

Guilded it is a platform for managing gaming teams. On Guilded, there are teams for games like League of Legends, Fortnite and World of Warcraft. These teams use Guilded to manage calendars, discord bots, forum software, documents, statistics, and recruiting. This might sound confusing. Why does a gaming team need document management and calendars and analytics? Are we talking about a videogame team or a software company?

To understand Guilded, you need to understand the rapidly changing modern gaming ecosystem. Eli Brown is a founder of guilded.gg. He joins the show to talk about the world of gaming, its intersection with social media and the fascinating engineering problems involved in building a platform for gaming teams.

Before we get started, the FindCollabs Open is our second hackathon for FindCollabs. FindCollabs is the company I'm building. It's a place to meet collaborators and build projects. Our second hackathon has $2,500 in prizes. We've got with prizes for the best machine learning project, music project, art project, podcasting, cryptocurrency, computer game design. I'm sure some of the listeners to this episode might be interested in that one, and it's a great place for creativity, collaboration. I hope you check it out. I hope you are inspired by FindCollabs and would love to get your feedback. You can go to findcollabs.com/open to find out more, or just go to FindCollabs and make a project.

Thanks for listening, and let's get on with today's show.

[SPONSOR MESSAGE]

[00:02:19] JM: Codacy helps development teams of all sizes to automate their code quality by identifying issues through static code analysis both in the cloud and on-premise. The Codacy product notifies users about security issues, code coverage, code duplication and code complexity in every commit and poll request directly from their current workflow.

Codacy has been designed by developers to be easy to set up and use and it's completely free for small teams up to four developers, and it's also completely free for open source projects. You can find out more and try out Codacy by going to softwareengineeringdaily.com/codacy. That's C-O-D-A-C-Y.

Codacy is a tool for static code analysis and issue identification. It will help you find security issues and code duplication, all these other issues that you can find through static code analysis. Checkout softwareengineeringdaily.com/codacy, C-O-D-A-C-Y.

Thanks to Codacy for being a sponsor.

[INTERVIEW]

[00:03:38] JM: Eli Brown, you are the founder at Guilded. Welcome to Software Engineering Daily.

[00:03:42] EB: Yeah, thank you. Thanks for having me.

[00:03:44] JM: It's 2019. Describe the state of online gaming.

[00:03:49] EB: Online gaming, yeah. So I think the first thing which is obvious to most people is that it's growing really, really, quickly. So, now, depending which games you count, no matter which sort of games, which genres you count, we’re well above 800 million people playing these online games. One of the big trends that I think most people have kind of overlooked is that there's been this big shift from games that are more of like solo games to games that are more team-based.

So 5 to 10 years ago, a lot of the really competitive games were games like StarCraft, and Warcraft, and Hearthstone and stuff like this, and all of a sudden now, all of the [inaudible 00:04:28] sports are team-based. So we have games like CS Go, League of Legends, Dota, Apex Legends, Fortnite and all these games, and it really seems like esports an online gaming is kind of following this trend that we obviously see in traditional sports, where the largest sports franchises are all based on teams. So we have the NBA, NFL, MLB and all that kind of stuff. I think if I could summarize, I would say there's two big things. One, that it's growing really, really fast. Two, we're sort of seeing the shift into sort of team-based competitive play.

[00:05:04] JM: Are you a gamer?

[00:05:06] EB: Yeah, I am. So that's kind of what got me into software engineering. I was a gamer before I was an engineer, and I got into it I think like a lot of people, because I wanted to make games. Yeah, that's sort of I think motivated my whole career.

[00:05:20] JM: What are your favorite games historically?

[00:05:22] EB: So I think I'm going to have to give it to Age of Empires II: Age of Kings. That was the first game that really got me sucked into online play, and StarCraft was an awesome game. I play that a lot, but it's a little bit sad now because the RTS genre is dying a little bit. But I just remember Age of Kings just getting me sucked into online games.

Then I think I'd have to also give a shout out, the old Diablo series. Yeah. Really, really –

[00:05:49] JM: Are you just a computer gamer or did you ever get into things like Magic, or Poker, or like terrestrial games?

[00:05:57] EB: Oh, yeah. There's a lot of overlap between these communities. So, I was also into Magic. So I played Magic the Gathering a lot of my early life, and it's interesting because there is also this overlap with people that play poker and stuff like that. I never got into that, but I do know people that did it. There's a lot of overlap with the shooter community and poker, which I think is kind of interesting.

[00:06:15] JM: And the Magic community –I'm a longtime Magic player, and actually I was just talking to a friend last night because I'm going to – There's a prerelease this weekend apparently. I don't play that much Magic, but like I happen to be in town. I'm going to Austin, hanging out with some people, and there's a prerelease, and there's a new set coming out, and we're actually doing a two-headed giant event. I actually think what you're saying about people increasingly wanting to play on teams, that was kind of why I took the circuitous route of questioning, because I wanted to probe it whether or not you were a Magic player. Because it's interesting that even eating Magic, there seems to be an increase in this team-based ploy.

[00:06:55] EB: Yeah, I would definitely, definitely agree with that. One of the things that I thought was probably the most fun I had playing Magic was when I kind of got back into it a few years ago, and I was at Microsoft at the time. This was up in Washington where Wizards of the Coast is. They had like a company league. So it would be like Microsoft and Google and Amazon and all these companies and you play in this team and go compete against other companies, and it was one of the most fun ways to play Magic that I think I remember. I think Wizards is trying to get into that more, because they've been moving into sort of this online Magic the Gathering thing, which is interesting in its own right, because magic has super complex rules and there's a weird challenge there around making it like an online game. But besides that, I think that's also bringing like a larger focus on team-based sort of multiplayer. Yeah.

[00:07:42] JM: As a software engineer, why does that seem like a difficult problem? Why does implementing Magic – I mean, Magic has kind of a checkered history in terms of moving online. I don't know if you remember the whole leaping lizard. Originally, like the first version of MODO. They tried to outsource it, and that didn't work so well. Then like – It's been a slog. I think they're in a much better place now, which is I'm thankful for that, because now they've got Blizzard breathing down their neck with Hearthstone.

But from an engineering perspective, why do you think magic is a hard – Speculating, of course. Why does Magic seem like a hard game to implement to you?

[00:08:17] EB: Yeah, I think about this a lot, and I'm not sure I have a great answer. But I think Magic is interesting, because the rules actually lend better to human communication than like computer, like software communication. For example, Magic has these really complex phases. If you play Magic by the book, it's like six or seven phases and you're supposed to move between each phase, and like all these stuff, and there are sort of like soft rules around the way you communicate with someone to sort of move through those.

When you put that into a computer and you lose all that human interaction, first, if you do it by the book, it feels very mechanical and sort of formulaic and it loses a little bit of the fun. If you don't, then you end up in like weird ambiguous situations, because Magic has a lot of really complex mechanics that can – Just so many edge cases.

It's like one of the hard things about programming is when you don't have sort of like this symmetry, and Magic just does not have that. There're just so many weird edge cases and strange mechanics, and Hearthstone is an interesting case, because Hearthstone is kind of like what a card game would look like if it was designed to be played on a computer.

So all of the mechanics work really well, like it fits in like a certain amount of time instead of like these phases. Yeah, I thought that was just really interesting, and I'm curious to see if they can make Magic work online. I mean, sort of overcome the inherent difficulty in having created a card game that's for humans to play and not for computers to sort of compute.

[00:09:46] JM: We will get to discussing Guilded and online gaming a little bit more. But since we're on the topic and you seem like you're willing to speculate a little bit. When you look at these open AI competitions and they master things like Go or they master things like StarCraft even, how does that compare to what you would consider like if they would've mastered Magic? Because I look at Magic and I'm like I'll start getting scared of the AI if it can master Magic, but it seems pretty far from that.

[00:10:18] EB: First of all, I think it's kind of inevitable that at some point AI will master all of these games. I do think that there's this sort of hierarchy of difficulty based on the game mechanics. In particular, one area that AI has a big advantage on humans are all the mechanical interactions.

So, on StarCraft, computer does a much, much better job of doing things like splitting marines to avoid banelings. If you're not familiar with that, basically, it's really mechanically-intense movement where you have to split up a bunch of units really quickly.

So most game sort of have this balance between being mechanically demanding, first, being strategically demanding. I think the first games that AI is going to conquer are games that are more mechanically demanding. So, for instance, I think it's actually easier to create an AI that is top-level at StarCraft than it would to, for example, let's say Age of Empires, because Age of Empires has higher strategical requirements and lower mechanical requirements.

Dota is an interesting case, because it's a team-based game. Some of the heroes have very high mechanical requirements. Some of them have low ones, but you actually – As long as you get the mechanics down, the challenge is in actually coordinating all of the players. So I think it's really cool that open AI in particular is tackling that one first, because I'm not sure exactly how I would rate the difficulty of mastering Dota against a very strategically demanding game like Age of Empires.

So, yeah, I'm curious to see how that works out. If I had to guess, I would say that StarCraft will be conquered before Dota, and then at some point in the future we'll have games with a lot more human interactions that are more strategy and less mechanics. Kind of like I think Magic. I'm not sure if many efforts have been made there. But, yeah, I'm really curious to see how that would go.

[00:11:59] JM: well. I think there have been some efforts that have been made. In fact, I remember I had a friend in college who was kind of thinking about this a little bit. He was trying to work on it. It just seems so hard. I think there's something about Magic that makes it very, very hard to figure out how to approach modeling an AI that can solve for so many different circumstances. Anyway, that's for another show.

So getting into online gaming, I get the sense that Twitch, and to some extent, other social platforms, have had a dramatic impact on some of this team-based play, some of the other

market trends in gaming. How has the interaction of social networking and gaming affected the industry?

[00:12:47] EB: Yes. I think you're right that Twitch is one of the really big ones. I would say before that, Steam did a lot as well. I think Steam was one of the first platforms that really got PC gamers sort of together and playing these games like CS Go and Dota and all of these kind of stuff. I think Twitch did a lot to bring it to the mainstream, because when we talk about like esports, that term, there are a lot of people that play games. But what Twitch did was actually opened it up to all of these people that might just like watch someone play Fortnite while they're eating cereal in the morning.

Twitch had a massive, massive impact there, and I think I would say Steam was big. Twitch was big. Discord is really big now as well in terms of the way that they are essentially making it really easy to form these groups and sort of communicate and play together all that stuff. So I think that Twitch is actually responsible for maybe like almost single-handedly for broadening the reach of esports in like a pretty substantial way.

[00:13:48] JM: What is a guild?

[00:13:50] EB: Yeah. So there's weird terminology around this. So, traditionally, a guild is like a group of people usually on like an MMORPG. So the different terms you'll here on games are guilds, which are traditionally, like I said, MMORPGs. We have clans, which sometimes are on MMORPGs, but more often shooter games, like CS Go. Some games refer to them as teams. So you have a team on Fortnite, like I think most newer games like a League of Legends team.

Yeah. Then we have like EVE Online and Star Citizen, which I believe refers to the groups as orgs. So I think over time this is going to standardize, but right now we're calling them teams, because I think that's where we're going to land. But the terminologies are a bit fractured at the moment.

[00:14:35] JM: Describe the interactions among members of these teams or these guilds.

[00:14:41] EB: Yeah. So there's a whole sort of set of things that teams and guilds want to do with each other. So I think the basic one is communication. So that's chat. So it could be text chat, real-time text chat, voice chat. Then there's a whole bunch of sort of higher-level organizational behavior on top of that.

So if you play World of Warcraft, for instance, you might have these like 20, 40 person raids, and they're just like in no way to actually coordinate that many people without using something like a calendar. It just doesn't really work to do that over voice chat. Once you get into big groups like that as well, you also sort of want to be able to recruit people to your guild. You often have like application process and people fill out a form and submit it. So there's that whole thing.

Then on the shooter and like mobile games, there's also this sort of behavior where teams want to find other teams to play against. In almost all games, the in-game matchmaking doesn't really work that well, and it seems like that's just the sort of whole other means of communication that is sort of important to teams.

[00:15:47] JM: So it sounds like there is the set of tools that teams need, and these sets of tools or these tools are – You could do the thing where you just like grab the tools that exists, like grab Google Calendar, grab Slack, or grab Discord, or whatever, and patch these things together into a workflow for a gaming team. But my understanding of Guilded, your company, is that you kind of have a vision for tools that are more specifically designed for the world of gaming.

[00:16:22] EB: Yeah, that's exactly right. So sort of when I had this idea, we played this MMORPG called Rift, and we just wanted all that stuff. We just wanted a place where people could submit an application to join our guild, and we could post some strategies, and all you could really do is, like you mentioned, either use like Google Docs or Google Calendar or create a website and try to put like a PHP bulletin board on it. It was just very broken and very stupid.

So I think, yeah, those are pretty much our options, and we created Guilded to kind of solve that. So you can create a team and in 30 seconds we sort of give you all those tools. One limitation that I think a lot of companies haven't realized, I think Facebook actually ran into this

issue when they tried to incorporate streaming into their platform, is that Google Docs and Google Calendar don't work for a lot of gamers, because gamers have their sort of online pseudonyms and they don't like to use their real-life accounts for all of their gaming communication.

So there're exceptions to this, but very, very large populations of gamers still operate in that way. That's one of the things that's sort of challenging about using Google Docs and Google Calendar. It's like all of a sudden everyone is using their real-life accounts and now they have another website to check, and it's just too much friction and it just didn't work very well for a lot of guilds.

[00:17:37] JM: So pseudonymity is basically something that is important to the gaming world.

[00:17:42] EB: Yeah. So I'm interested to see how this plays out over time, because as esports become more like real sports, I'm not sure if in the future everyone will just go by pseudonyms. But right now, people keep their gaming life mostly separate from their real-life. So they have a different name and a different identity, and it's usually not tied back to their real-life identity. So products like Slack, products like Google Calendar, Google Docs, Facebook streaming, like I mentioned, sort of have this problem where unless they embrace this completely different sort of system of identity, it's really hard for them to build competitive products in this space.

[SPONSOR MESSAGE]

[00:18:26] JM: DigitalOcean is a reliable, easy to use cloud provider. I've used DigitalOcean for years whenever I want to get an application off the ground quickly, and I've always loved the focus on user experience, the great documentation and the simple user interface. More and more people are finding out about DigitalOcean and realizing that DigitalOcean is perfect for their application workloads.

This year, DigitalOcean is making that even easier with new node types. A $15 flexible droplet that can mix and match different configurations of CPU and RAM to get the perfect amount of resources for your application. There are also CPU optimized droplets, perfect for highly active frontend servers or CICD workloads, and running on the cloud can get expensive, which is why

DigitalOcean makes it easy to choose the right size instance. The prices on standard instances have gone down too. You can check out all their new deals by going to do.co/sedaily, and as a bonus to our listeners, you will get $100 in credit to use over 60 days. That's a lot of money to experiment with. You can make a hundred dollars go pretty far on DigitalOcean. You can use the credit for hosting, or infrastructure, and that includes load balancers, object storage. DigitalOcean Spaces is a great new product that provides object storage, of course, computation.

Get your free $100 credit at do.co/sedaily, and thanks to DigitalOcean for being a sponsor. The cofounder of DigitalOcean, Moisey Uretsky, was one of the first people I interviewed, and his interview was really inspirational for me. So I've always thought of DigitalOcean as a pretty inspirational company. So thank you, DigitalOcean.

[INTERVIEW CONTINUED]

[00:20:34] JM: It's interesting looking at the question of pseudonymity, versus anonymity, versus real names across the spectrum of different social platforms that we have. I think about Twitter. Twitter allows pseudonymity. Perhaps that is one reason why Twitter is so successful. Facebook does not allow pseudonymity. On Quora, you have basically either your real identity or pure anonymity. There's not really much in between. I guess you could make like a company account if you wanted to do pseudonymity. Yeah, I don't know. Its' kind of an interesting trend in the world.

[00:21:13] EB: Yeah, I think this is really important, and I think that social media has exploded so fast that we haven't really had time to sort of introspect on why these things are important and where they're important and sort of what different benefit you get from these different systems of identity.

So the fact that Facebook enforces a real name policy is bad in some ways, but in some ways it's very good. It does make the platform feel safer. It makes it feel easier to trust the people you see on there. To some extent, I think it reduces the amount of flame wars that can happen, because there is some component to this where people are less likely to be toxic if it's attached to their real-life identity.

But I think on the other hand, there's real value in anonymous and sort of pseudo-anonymous systems, because often those platforms have more real conversations, because you're able to ask questions and answer questions in a way that isn't sort of biased or some sort of influenced by your concerns about attaching that question or opinion to your identity.

I think Reddit has actually done the best job of this. So Twitter is kind of a mix, like you said. But, in many cases, for a given topic, you'll find the most real and honest conversations on Reddit, because people don't have the motivation to sort of think about how the things they say affect their image. There're pros and cons there. But there's a lot of value to it. I think that, hopefully, we're going to figure out how these all sort of piece together. It feels like we're not quite there yet, and I don't really know exactly how to solve that.

[00:22:55] JM: Describe the tools that you have built in Guilded.

[00:22:58] EB: Yeah. So when we first launched it, we had forums. We had a calendar, media albums, recruiting tools and documents. It's sort of like longer form documents. The first one that really caught on was the calendar, because there's no good way really for gamers to schedule events right now. We have really popular tools like Discord, and Discord does an awesome job of real-time chat, but real-time chat doesn't work for everything. One of the things that particularly doesn't work at all for is scheduling events.

Forums are interesting, because both in gaming and increasingly in like the sort of work communication tool space, it feels like there's been this sort of backlash where it's like, "Hey, Slack isn't good for asynchronous communication, because it distracts you and we want to have longer forum discussions." It feel like there's almost been like this sort of like revival towards rival of forum software, which is interesting to me. But gamers in particular really love forums. I think it's just this sort of ingrained cultural thing. Like every game has a game forum and you go and read strategies. I don't know. I think it's really interesting in that regard. So we started out with just those and we've added a few more things on to that overtime.

[00:24:10] JM: It is hilarious that the persistence of forums, the never dying social network entity. Speaking of gaming, when I was learning poker, I would spend hours and – Because I

was one of the people who did do the conversion from Magic to poker. I would spend hours and hours and hours on –There's a form called 2+2 that you may or may not of heard of, but basically it's the dominant poker forum. For I know, it could still be – It wouldn't surprise me at all if it was still the “dominant”, I guess, social networking platform for poker, partially because the game is so simple and it's like all you need to do to really share – The equivalent of like where you share your live stream from a Fortnite session like on Twitch. It's pretty easy to share a single hand history of poker on a forum, and I don't think like there's that much like technological improvement you could – I mean, people do live stream their poker sessions. I guess that's a little bit different. Anyway, just hilarious forums. Man! Forums!

[00:25:11] EB: I think it’s really interesting, because the tools, sort of like the tools and the communication platforms you use very much influence like the content and the discussion people end up having, and I think the fact that forums are a little bit more inaccessible and you have to let go and type like a real well-thought out thing. It's like you can't just like type five words and just bang, enter. You have to actually think about it and the expectation is that you write like a paragraph or something.

I think just that barrier to entry actually significantly elevates the conversation. I just think that's so interesting, that you could just take a chat room and just visually turn it into a forum, and I think it actually would like substantially change the conversation for better or worse.

[00:25:53] JM: So you got a lot of things that I see you could potentially work on. You’re already working on a lot of things. Just to kind of fast forward the conversation a little bit, can you tell me the hardest engineering problem that you have worked on or that you're working on right now within Guilded?

[00:26:10] EB: Yeah, for sure. So, what I had sort of noticed before we made Guilded was that a lot of the problems that we were solving, I’ll give you one as an example, is let's say you start playing a game and you want to find other people to play with instead of just playing alone. So if you take any given game, like let’s just say Destiny, there are tools for that. There's like destinylfg.net. There's a Destiny subreddit where you can try to find groups.

What I noticed is that for every game, there's like these tools that exist only in that game. One of the things we wanted to do from the start was we wanted to make sure that basically we could spin up any game and all of our tools worked across all games. The difficulty there is that we had to basically create some abstraction over a game, and it ends up being really, really hard. Because if you're trying to figure out who can play with other people, the rules get really complicated actually.

So, for example, people cannot play with – If you play Call of Duty on PS4, you can't play with people on PC. So there's this platform aspect. If you're in North America, you probably don't want to play with people in Japan. But that's not always true, because if you play Fortnite, actually, people can play across consoles. So Xbox and PC can play together. On games like World of Warcraft, you can really only play with people on the same server, but wait, you can pay $60 and transfer to another server.

So what we ended up kind of doing is building this big abstraction that essentially classifies games and allows us to describe every game as sort of like this set of properties. Then as we go, we can describe this game – Basically, you can think of it as like a big JSON blob. It sort of just plugs into our system now. But it took a lot of work to do that, because it's a hard problem to deconstruct. I think that that was one of the engineering problems that we had to take on kind of early to make sure this worked at all. Because otherwise we sort of get into the thing where we have to spend a whole bunch of effort every single time a game comes up. So I think that was probably one of the early ones.

[00:28:12] JM: Wow! Okay, so I can imagine every game as a de-normalized entity where you have a Discord channel, and a subreddit, and a Twitch stream and the etc., etc., etc. So I can imagine it being useful to kind of have this, I guess domain specific language or config system where you sort of represent everything in just a JSOn blob.

[00:28:36] EB: Yeah, that's exactly right. I think is a byproduct to this, we've probably built one of the biggest catalogs of just game data. It's like for every game we have this very sort of, now, verbose description of how it works. We have the Twitch channel, the subreddit, all these stuff.

Another really interesting dimension on this is that let's say you play League of Legends, and on League of Legends, you have to have different roles. So you need like a support, and a carry, on all these stuff. So if a team has four people and no support, then we're sort of – We build this engine where we'll recommend you teams.

Well, we actually have to know that and we have to have some concept of what teams fit together. Your first instinct would be to go and say, "Hey, Mobus have this behavior where it's like this," but this actually is like – It's actually a characteristic that applies to a broad set of games. For example, Overwatch is exactly the same.

So overwatch has actually taken some of these like mobile concepts, and now the shooter game has the same sort of thing. So at first we modeled this is a mobile thing and then we had to pull it out. Basically, it's been a lot of work to make sure that we can model these games in a way that it sort of allows them to plug into our system and allows us to support like hundreds of games at once.

[00:29:47] JM: What's your engineering stack look like?

[00:29:49] EB: Yeah. On the frontend we use React. So it's a web app. We use MobX, which we've really, really loved so far. I think we're probably one of the bigger, if not the biggest MobX, 100% MobX app out now.

[00:30:00] JM: That's like a Redux-like state management thing.

[00:30:04] EB: Yeah, it is. So the quick TLDR of the difference is that Redux basically use these action creators, and it's been a while since I used it. Yeah, reducers, all these stuff, and you sort of describe your state in this like immutable set of information that is kind of global variables. MobX, you basically defined usually data as properties of components or on stores and then it's reactive. So you basically store your state and then everything is sort of computed from that using some MobX magic in the background. So that's worked really well for us. On the backend, we use node, PostgreS and Redis, and this is all running in AWS.

[00:30:44] JM: Now, why did you go with AWS rather than some more – Because you're a millennial, like me. Why now go with like a Heroku, or a Firebase, or some other hipster backend as a service?

[00:30:58] EB: Yeah, it's a good question, because there's always this balance between how hipster do you want to be and how much do you want to get stuff done. For us, I think a lot of people over complicate deployments. I think AW – To be honest, this wasn't like a very principled decision. It's not like we think AWS is the best out there. But what I do think is that AWS is very proven out. A lot of engineers know how to use it. You can Google anything and you get answers, and that's really good for us.

I mean, our deployment process is just like this automated thing where you just put a zip file into Elastic Beanstalk and it just rolls it out, and that's really all we've needed and probably all we'll need for the foreseeable future.

[00:31:37] JM: Amazing. Okay. Well, more specific to what you are working on. So you've got this like config language that describes every game. I can imagine, there's some kind of like basically compilation or a rendering phase that – I could imagine, there's a lot of interesting problems around this config language. So first of all, you need to like either crowd source or manually gather the data yourself to add to the config language, and then you need a way to render a different config thing. I guess tell me more about your config fits into the overall model of Guilded.

[00:32:15] EB: Yeah. Yeah, there's a few interesting challenges there. One is sourcing all of these data. Early on, so when I started this, it was just me just like hacking stuff together. All of a sudden I needed a data for like hundred plus games, I and didn't want to do it myself. It took a really long time, but what I was able to do some was break that apart and use – What is that? Oh, yeah. Amazon Mechanical Turk to just sort of allow people to plug in information.

[00:32:42] JM: Beautiful. Beautiful. That's hilarious.

[00:32:44] EB: Those are really, cool use of that that it took me too long to think of, because I was –

[00:32:48] JM: So they were capable. The Turks were capable of doing this.

[00:32:52] EB: Yeah. If you break it apart into simple enough problems, it works pretty well. So we don't do that anymore, because now we have actual admin tools and now we have people on the team. They can go and just fill all these stuff out and it's actually faster than trying to break it into Mechanical Turk tasks. But, yeah, that was one interesting problem.

The second one was – Yeah, at first we just had a big JSON blob and we just added to it. One of the problems that we run into very quickly is a lot of this kind of information is derived. We also have things like, for example, each game has like its own set of colors, and then there's like slightly different colors that are computed that are using different parts of the UI and all of a sudden that runtime were computing the stuff like 400 games. So we had to create like this sort of pre-build step.

Now, the problem that we're solving now is that we have this gigantic JSON blob of data and it's really hard to break it apart and we don't really want to make client load 200 kilobytes of JSON for our games at the time we started up. Yeah, this has been something that we've run into kind of a series of technical challenges for. I think that if we do it right it will be a differentiator. But I think it is like important prerequisite work. Yeah.

[00:34:03] JM: Yeah. That sounds pretty hard, because it's like I assume you have some pretty rich objects that you need to load there, like loading a Twitch channel. I don't know to what

degree you can like do server-side rendering with some of this stuff. I mean, how are you playing with this problem?

[00:34:21] EB: So in terms of server-side rendering, we actually took a very different approach to this, which I think has worked out really well so far. So server-side rendering does a few things. Well, one is that it can decrease the load time on the client if the HTML is hydrated correctly. The second is that it's good for web crawlers and all that kind of stuff.

I've been thinking on this for a long time and I really wanted to avoid server-side rendering, because it's a huge pain in the ass and it's just something that I just totally wanted to avoid. The solution we came up with that I've been really happy with so far is that we basically have our single page app and an index.html, and this is just sitting in CloudFront. So it's always cached, always on the CDN. Yeah, AWS has this thing now call AWS Lambda, and it allows you to basically run JavaScript on the edge.

What we can kind of do is if when a request sort of hits the edge, the CDN to request some certain URL, if it's a crawler or if it's something that is like building like open graph data like Twitter or Facebook or whatever, we can just hit our server and then just append that information to the index.html and then they get like an HTML with like hydrated information.

But the net effect, which is kind of cool, is that on the client, whenever you load Guilded, you always load the same cached JavaScript bundles, the same index, and you don't have to wait for a server to render anything. In our test, it's actually a lot faster to do that than it is to hydrate the HTML on the backend. The client can usually do that faster as long as you can serve at the files quickly.

[00:35:56] JM: This is amazing, because I've talked to people like Fastly and Cloudflare, and they're thinking about this problem quite deeply, like, "Oh! Edge is the thing. Edge workers, edge Lambdas, edge functions as a service. These things are going to be necessary to run the kind of performant workloads that we need. We need to move computation to the edge. We've already done that with content, with CDN's. We're now doing it with computation." I think you are the first use case, to my recollection, that really has – That's a really, really compelling and kind of subtle use case.

[00:36:39] EB: Yeah, I was surprised, because we had looked around for answers to this for really long time and it took us too long too long to figure it out. But I think a few months into it, it just kind of clicked and we tried it out and it was very, very cool. I do sort of agree that there's a lot of value in running this stuff on the edge.

We've had a few other use cases that have been really, really cool. I can't really think of them at the moment. Yeah, I think we're finding more sort of scenarios where we do just want to run a small amount of JavaScript on the edge instead of having it hit our backend servers and come all the way back. So I think that's a trend we're going to see continuing. I think there's a lot more to explore there.

But I should probably do some – It'd be interesting do like a write up about this Lambda thing at some point, because it's worked really well for us and I expect that there'd be some downsides, because I was like, "Why don't I see other people doing this?" and we haven't run into any so far. So, yeah, it's kind of interesting.

[SPONSOR MESSAGE]

[00:37:40] JM: FindCollabs is a tool for managing hackathons and innovation within your company. FindCollabs allows anyone within the company to create new ideas and build momentum around a new initiative. FindCollabs allows your smartest, most driven employees, to build projects organically. If your company is looking for new ideas and innovations, check out FindCollabs. It's free and it was started by me. It's something I genuinely believe in, and if you have any ideas or complaints or criticisms of FindCollabs, you can always email me, jeff@softwareengineeringdaily.com.

FindCollabs lets people within your company create new ideas. Whether you want to run a hackathon and generate new ideas, or you want a long-term system to manage innovation within your company, check out FindCollabs at findcollabs.com.

[INTERVIEW CONTINUED]

[00:38:45] JM: This is kind of a weird question, but if you build like a strategic core competency in running stuff at the edge,, like broadly speaking, or running these kinds of workloads, whether you want to call it like a late binding compilation workload for your configuration language, or just more generally, like running computation at the edge and serving highly performant a complex frontend applications. How do you think that core competency could be useful to the future of Guilded?

[00:39:17] EB: So I think that one of the challenges that we'd sort of wanted to solve here is sort of think – So here's an interesting case. So Discord for instance is a single page app. It's like a JavaScript app, and it all just kind of runs in the browser. I think one of the challenges that Discord has is all the servers are private. So you can't really like Google a server. If you do click into it, there's nothing there. For us, it's like really important for a team that they have some sort of public identity.

So we're not going to succeed unless our presentation not only to users but also to search engines, to different sites that do present open graph data, and things like oEmbed, like all these stuff once you go on this rabbit hole. We have to actually nail that, because we give teams of public presence. It's not clear at the moment with the latest sort of a trinity JavaScript stuff. How you create a really good, really performant, single page application and have all of that stuff without constantly maintaining server-side rendering?

Server-side rendering has gotten better in a lot of regards. But in my opinion, it is still a very high engineering cost and there are kinds of issues that can kind of creep up and bite you there. So I do think that if we get this right, and I don't think we've totally gotten it right yet. But if we do get it right, I think we'll sort of be a differentiator for us in some sense or it will at least allow us to build the product that we want to build without sinking a whole bunch of engineering energy and server resources into server-side rendering.

[00:40:52] JM: What is the product you want to build? What's the big vision?

[00:40:55] EB: Yeah. So our mission is to connect esports teams. So if we succeed, then for every game on every platform, you go to find a team, you go to Guilded. When you Google the name of a team, the first result is Guilded.

. Basically, if you think of like Twitter or Facebook as like a platform for people and Twitter as a platform for kind of people and fake people and venture capitalists, then I think of Guilded as sort of the platform for teams. We should be – Just like when you at Instagram, you can type in any celebrity and they show up. That what I imagine Guilded to be. I imagine that we give you all of the tools you really need to organize a guild and to sort of find other teams, discover other teams in one place.

I think the next step for us as well is if we're the platform that has all the teams, I think that it makes sense for us to be the platform where teams go to play in leagues and to play in tournaments. I think it's kind of like Facebook Events. It's kind of interesting, because Facebook became the biggest platform in the world for scheduling events just because events are something. It's just like a manifestation of how people interact, and Facebook has all the people. So I think that the platform that has all of the teams should eventually own tournaments and leagues and all of that stuff.

[00:42:11] JM: Love the vision. Now, as with a good start up vision at the kind of stage, you're kind of series A, I think, or you just – I don't know, seed or series A. Whatever. You don't have to disclose. It's something early. Basically, you're kind of trying to have your product future converge with the future of the industry. The industry is not quite in the place that you're describing yet, but you're kind of like running as fast as you can to catch up to the future when it arrives.

[00:42:45] EB: Yeah, that's a great way to put it. I think we're taking kind – The bet we're taking on the future. one of those is that, obviously, esports are going to continue to grow at the pace it's going to. But I think the one that we're not totally sure about is that we're taking a bet that the future of esports is going to be based on teams. People are going to organize around teams, and the same way that people do with traditional sports, like NBA, NFL, etc., teams kind of become the basis of identity and organization and everything from there.

This is kind of an important bet, because we've had a lot of esports startups sort of come and go and some are still around and whatever. Some of them are like social networks for gamers, and they're basically taking a bet that the future of esports is going to be based on players. We're

taking a bet that it will be based on teams. Yeah, just like you mentioned, we don't know exactly how it's going to turn out. I think the future is going to trend in that direction and we're kind of taking a bet on that.

[00:43:40] JM: Why is server-side rendering so painful?

[00:43:43] EB: I think a few reasons. I think, inherently, it's just a hard problem to sort of build out all of your HTML on a server. It gets way more complicated when you have async loading. We know there are these sort of products and frameworks that now advertise that it's really easy. I could be wrong here. When I did it before at my previous job, we did this for Instagram web a bit. It ended up creating a lot of issues, and I did not see any clear way out of it.

So unless something has gotten a lot better in the last year or two, I think it's just like a fundamentally hard problem to solve. I think it also has just like some inherent downsides, like you're not going to get around the fact that all these requests, you have to run on a server somewhere, and rendering a ton of JavaScript on a node server is just not – It's like not of performant thing to do. So if you're going to end up serving a ton of people with this, you're also going to invest a lot of resources in that. That not only computing resources, but it's engineering resources, because you're going to have this separate environment for it. You're going to scaling them all the time. You have to monitor them, and I would just really like to take all that engineering energy and put it into building our product.

[00:44:48] JM: So when I think about the problem of server-side rendering, it's basically I want to have my pages as rendered as possible when the user requests them. In contrast to a situation without server-side rendering where like I request a set of React files and HTML and the rendering happens in my browser.

Now, I can see you hitting bottlenecks where the content is more dynamic and like can only be loaded on the browser side. What are the fundamental issues of doing that just like precompiled HTML stuff?

[00:45:32] EB: So I think one of the fundamental ones is that you don't know all of the resources that you're going to need or how the client is actually going to render until you've

rendered it in a real browser. So especially if you do async loading, like when you hit guilded.gg right now, you load our main bottle, and then depending on what different things you load on the site, we'll serve you sort of these code split bundles. Then that will make web requests, and depending on the content of the request, it might load some widgets that require some other JavaScript bundle.

The thing is, is that this is determined by the browser engine. Like when you run, when you load Chrome, there are so many different things that can happen. It's like your network quest can fail, like your browser can load it in a weird way. It's just like there's this whole mess around like how browsers render things. Like different versions of browser, different operating systems. It just like a huge mess. You can approximate that as best you can on the backend, but just fundamentally, it's always going to be a little bit different to run something in an actual browser and go down those paths and then discover all these resources that you need to load in this HTML that you have to hydrate. .

I think the best we're going to be able to do is an approximation. To be fair, the approximation might end up being good for 95% of 90%-95% of requests, but that 5% is like this thing that just eats engineering resources, because it's really hard to debug. You won't know why it's not working sometimes, and I think this is something that we can mitigate, but I don't think it's fundamentally solvable.

[00:47:07] JM: You say your database is PostgreS? Not to change the topic completely, but –

[00:47:12] EB: Oh, I love talking about PostgreS.

[00:47:14] JM: Why PostgreS?

[00:47:15] EB: Honestly, I was not the database expert when I started this project, but I asked some people that I think are really smart and they recommend PostgreS, and I think it was in retrospect one of word better decisions that was a little bit based on luck, because PostgreS has been fantastic. The PostgreS team makes updates really, really fast. I've been really impressed with that. It's just one of the things, it's things like developers like to work with it and it makes it

easier to hire developers when you have a stack that isn't painful to work with, like PostgreS is great, Redis is great, React is great, and it just sort of contributed to that.

[00:47:48] JM: Why? Why do people like to work with it? I've heard that there's like a plug-in or extension ecosystem or something. What makes it easy to work with?

[00:47:56] EB: So, I think one of the big ones for us is just common table expressions. That's something that PostgreS has that a lot of engines don't, and it's actually really, really awesome. It saves you a lot of pain in a lot of cases. I do find that errors with PostgreS are generally Googlable. It's hard to sort of track down everything. I wouldn't stick it to like, "Oh, there's this one thing PostgreS does," but they have all these sort of utility functions and different patterns that, for example, MySQL doesn't. It's just a very rich language, a very feature-rich database. In a lot of cases, these really save you a lot of pain.

In particular, I can think of one now. A lot of the PostgreS index constructs, in particular, JSON B and PostgreS is fantastic. The fact that you can – I think that PostgreS is a far better JSON database than like MongoDB is. Their JSON B functionality is fantastic. The fact that you can index JSOn B fields is awesome, and you don't always just want to dump things into JSON B columns. It's good to have a database schema, but there are some cases where JSON B works amazingly well and PostgreS has really impressed us with that implementation.

[00:49:05] JM: Why hasn't gambling become a bigger part of the online gaming world, or has it? How is the world of gambling and online gaming converging?

[00:49:16] EB: Yeah, it's a good question. So I think that it's actually going to grow a lot very soon. There is this recent Supreme Court decision maybe a year two ago that, if I understand it correctly, basically legalized gambling for online games in a lot of places. What we saw after that was a whole bunch of esports startups sort of a spin up around gambling. I personally think that as long as that holds, it's going to become a big thing.

I think that the first esports startup that can gamify gambling and make it really fun for espots matches is going to do really well. There's actually this platform that surprisingly not many people know about, it's called Skills, and they've sort of made a really big company around

allowing people to gamble on casual mobile games. When I first heard that, I almost couldn't believe that that could be a big market, because I couldn't imagine that all these people that play like this really casual mobile games would want to bet money on them.

But they found this really underserved portion of sort of the market. They battled through all the legal challenges there, and they created this product allows you to just bet money that you're going to beat someone at like Candy Crush or something. Not actually Candy Crush, but like some really casual game, and they've done really well with that.

[00:50:31] JM: It all comes back to Magic. Isn't it funny how the earliest versions of Magic had Ante, and like I think Ante is like an awesome mechanic. But they basically had to take it out, because it was too taboo.

[00:50:42] EB: Yeah, that's true. I totally forgot about that. When I was playing Magic, I would be at like card shops when I was like 14 or 15 and I'd feel pressured to play these Ante-games and older guys would come and try to like take my cards and all these kind of stuff.

[00:50:57] JM: Do you remember money draft?

[00:50:59] EB: No. What is that?

[00:51:01] JM: Oh, I'm sorry. I shouldn't have interrupted you. But since you've brought me back, money drafting is maybe actually not legal. So it's definitely something I didn't do. But it's like where you, hypothetically or theoretically can say like before the draft, like you do a three on three – Oh! It comes back to the teams also. Like three on three draft and everybody puts up 20 bucks or hundred bucks or $10,000 if you're an ex-poker player, and you draft. You do a draft and it's like first 1 to 5 match wins wins the dollars you put up and the cards.

[00:51:33] EB: Yeah, and that hypothetically sounds super fun. I don't know what the challenges here are legally really, but I think that if we end up with a game like Fortnite that kind of explodes and they have built-in gambling. Oh man! Whoever does that is going to make so, so much money, because it would be so fun to just get on with your friends and like bet some money on these games. It doesn't have to be a ton of money, but just enough to make it feel like

there's kind of stakes on it. I don't know the legal challenges around if you can put that in games or not. I'm really curious now, but if someone gets that right, I think it would be really, really bit.

[00:52:08] JM: I completely agree. Anyway, last question, and you can feel free to pass on this one. You were at Facebook for a couple years, or Facebook and Instagram for a year each. How is Facebook misunderstood especially in the common narrative today?

[00:52:23] EB: I think that Facebook is more misunderstood than it is understood. From my time there, I think one common misunderstanding is sort of based on the media portrayal. In large part, the portrayal of Facebook in the developer community, it's sort of perceived as this sort of – This group of people that doesn't care about your privacy doesn't care about protecting user information. There's all this false information about Facebook selling data. But while I was there, these sense I got from everyone was that everyone really did deeply care about – The users deeply cared about privacy.

One of the things that I think Facebook has actually done really well is that this sort of radical transparency value ensures that Facebook actually does more to protect people's privacy than I think people appreciate. Because everyone at Facebook, in my experience, very much sort of does buy into this concept that it's really important to protect user information. It's really important to make people feel safe on the platform. Any time that that doesn't happen, people are very vocal. There's like these internal forums and people will post, and there a lot of people at Facebook in my experience sort of protecting that. I think that's a misconception that many people have that it's sort of this evil entity that wants to [inaudible 00:53:39] your privacy.

[00:53:40] JM: Okay, Eli, it's been great talking to you, and I'm really excited to see where you go next.

[00:53:46] EB: Yeah, thank you. It was fun.

[END OF INTERVIEW]

[00:53:51] JM: GoCD a continuous delivery tool created by ThoughtWorks. It's open source. It's free to use, and GoCD has all the features that you need for continuous delivery. You can model

your deployment pipelines without installing any plugins. You can 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 cloud native project. With GoCD on Kubernetes, youdefine your build workflow. You let GoCD provision and scale your infrastructure on-the-fly, and GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn how you can get started.

GoCD was built with the learnings of the ThoughtWorks engineering team, and they have talked in such detail about building the product in previous episodes of Software Engineering Daily. ThoughtWorks was very early to the continuous delivery trend and they know about continues delivery as much as almost anybody in the industry.

It's great to always see continued progress on GoCD with new features like Kubernetes integrations so you know that you're investing in a continuous delivery tool that is built for the long-term. You can check it out yourself at gocc.org/sedaily.

[END]