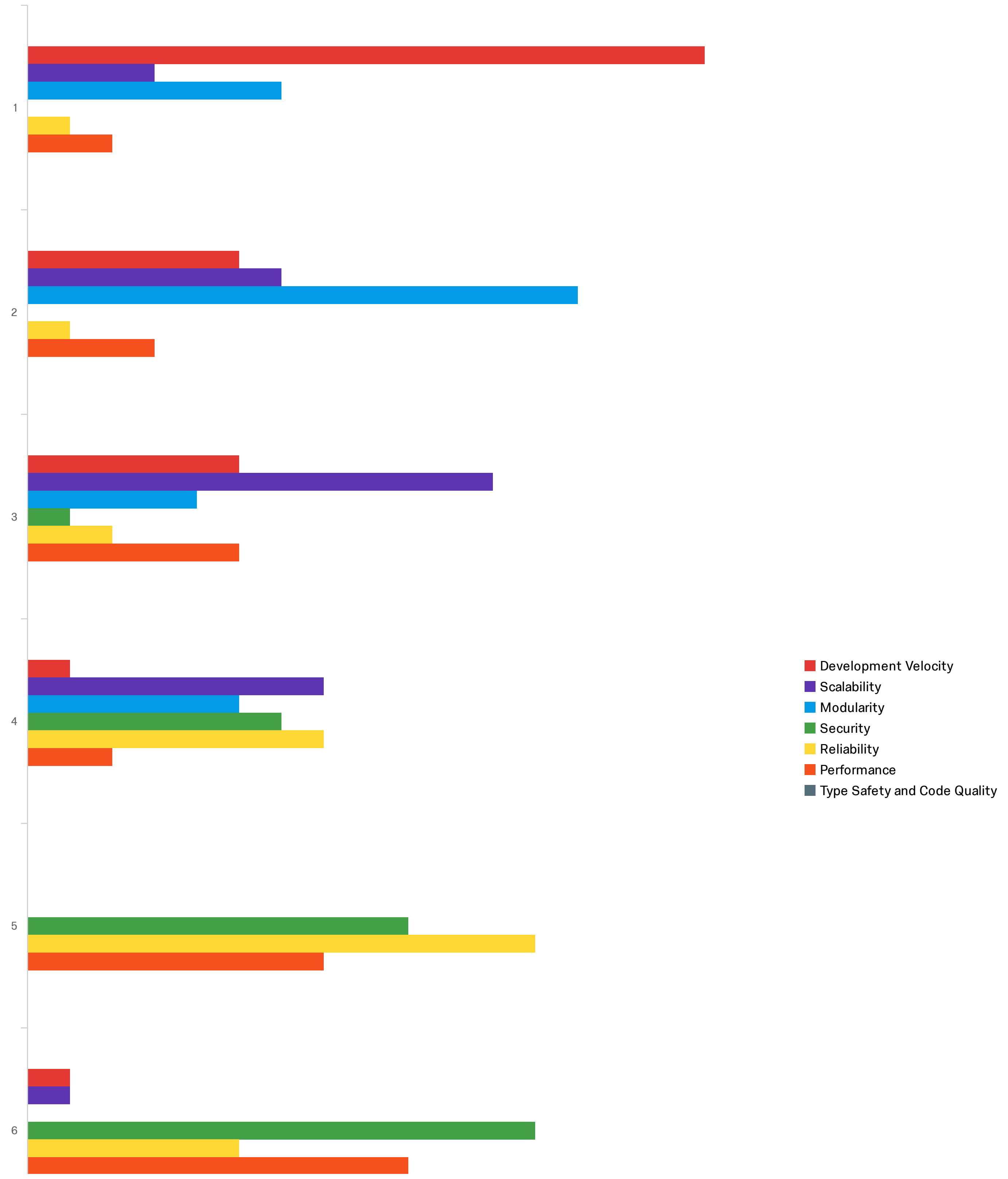
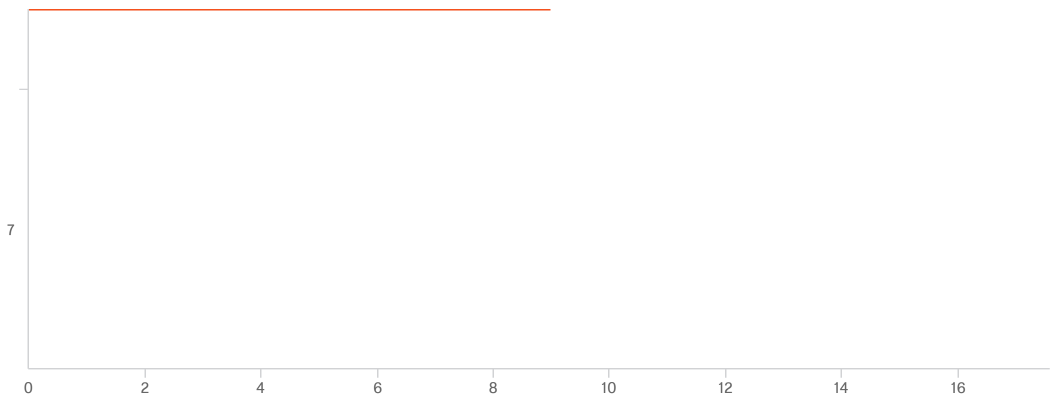
Default Report

GraphQL Survey

January 10, 2019 2:22 PM MST

Q1 - Rank the reasons why you moved to GraphQL



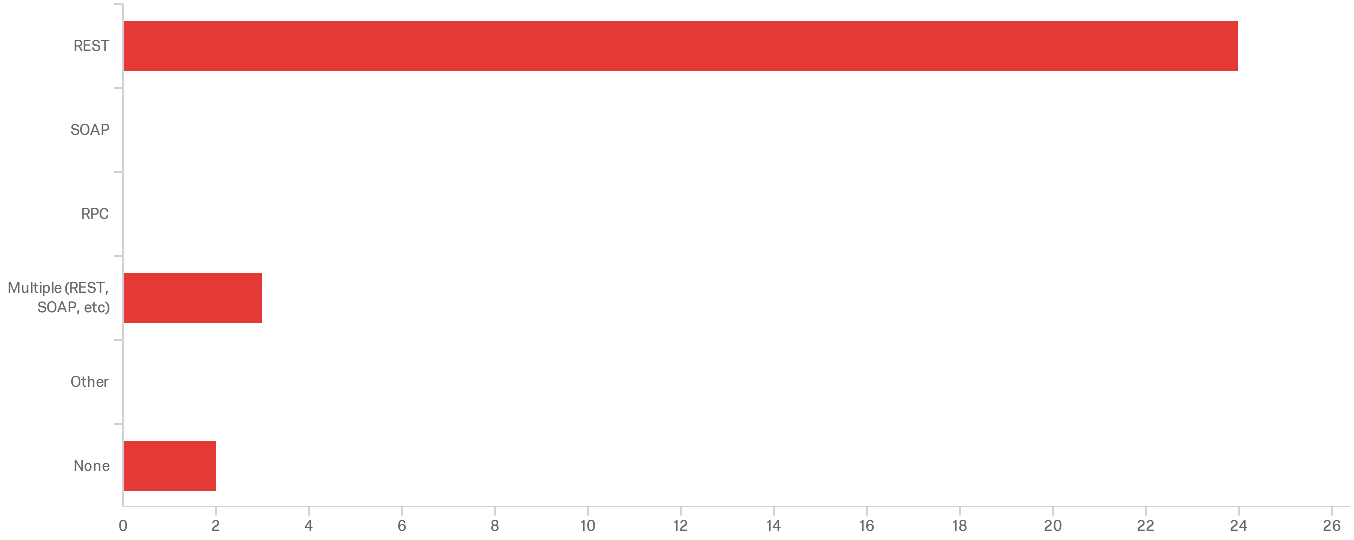


#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	Development Velocity	1.00	6.00	1.82	1.20	1.43	28
2	Scalability	1.00	6.00	2.93	1.10	1.21	28
3	Modularity	1.00	4.00	2.29	0.99	0.99	28
4	Security	3.00	6.00	5.14	0.87	0.77	28
5	Reliability	1.00	6.00	4.54	1.18	1.39	28
6	Performance	1.00	6.00	4.29	1.64	2.70	28
7	Type Safety and Code Quality	0.00	0.00	0.00	0.00	0.00	0

#	Field	1		2		3		4		5		6		7	
1	Development Velocity	57.14%	16	17.86%	5	17.86%	5	3.57%	1	0.00%	0	3.57%	1	0.00%	0
2	Scalability	10.71%	3	21.43%	6	39.29%	11	25.00%	7	0.00%	0	3.57%	1	0.00%	0
3	Modularity	21.43%	6	46.43%	13	14.29%	4	17.86%	5	0.00%	0	0.00%	0	0.00%	0
4	Security	0.00%	0	0.00%	0	3.57%	1	21.43%	6	32.14%	9	42.86%	12	0.00%	0
5	Reliability	3.57%	1	3.57%	1	7.14%	2	25.00%	7	42.86%	12	17.86%	5	0.00%	0
6	Performance	7.14%	2	10.71%	3	17.86%	5	7.14%	2	25.00%	7	32.14%	9	0.00%	0
7	Type Safety and Code Quality	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0

Showing rows 1 - 7 of 7

Q2 - What protocol did you use before you migrated to GraphQL?



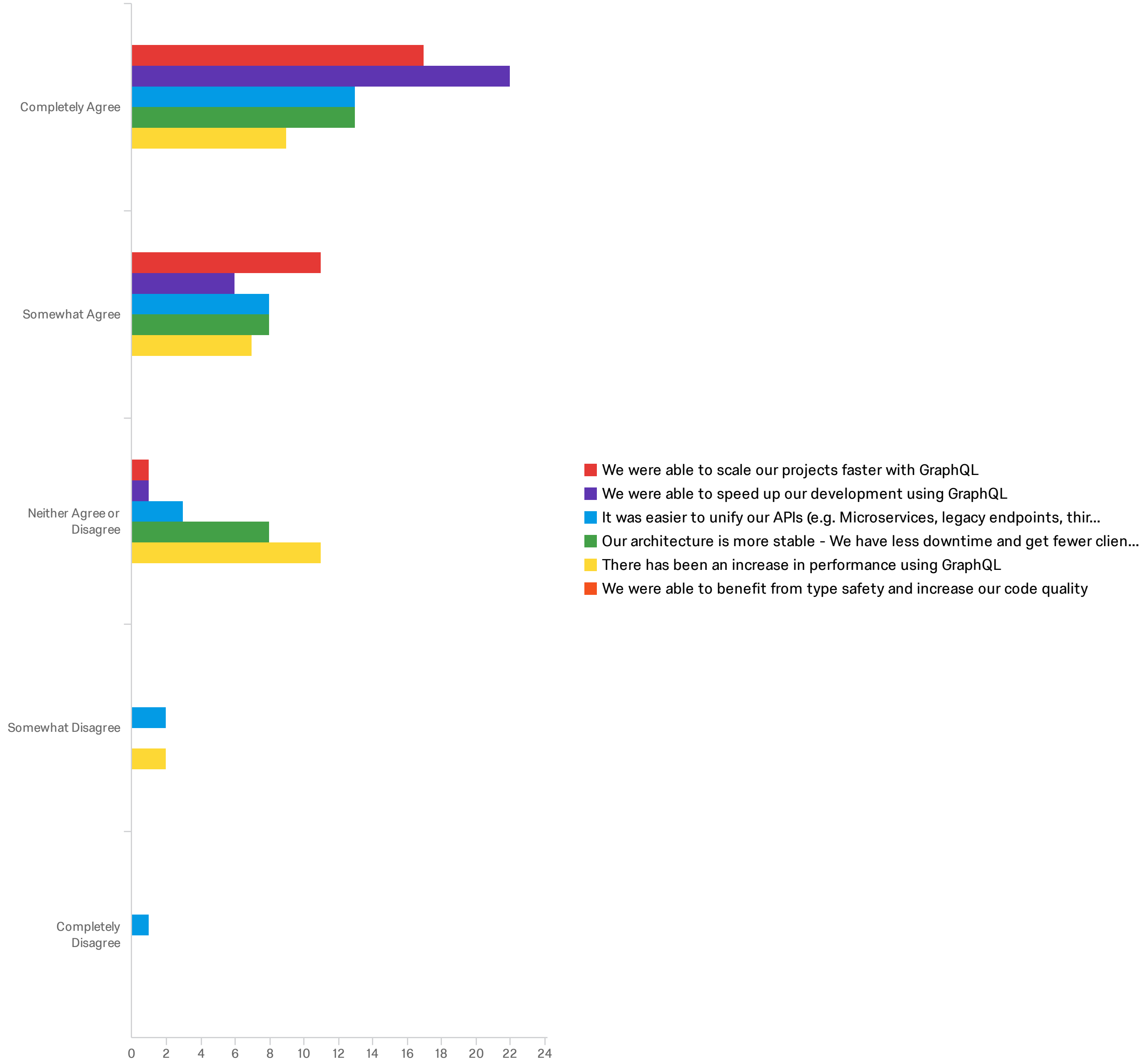
#	Field	Choice Count	
1	REST	82.76%	24
2	SOAP	0.00%	0
3	RPC	0.00%	0
4	Multiple (REST, SOAP, etc)	10.34%	3
5	Other	0.00%	0
6	None	6.90%	2
			29

Showing rows 1 - 7 of 7

Other

No Data

Q3 - Below are a list of statements on the impact of GraphQL on your product. Please indicate how strongly you agree or disagree with the statement.



#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	We were able to scale our projects faster with GraphQL	1.00	3.00	1.45	0.56	0.32	29
2	We were able to speed up our development using GraphQL	1.00	3.00	1.28	0.52	0.27	29

#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
3	It was easier to unify our APIs (e.g. Microservices, legacy endpoints, third party services etc.). Type NA if you didn't use GraphQL in this way.	1.00	5.00	1.89	1.10	1.21	27
4	Our architecture is more stable - We have less downtime and get fewer client errors.	1.00	3.00	1.83	0.83	0.69	29
5	There has been an increase in performance using GraphQL	1.00	4.00	2.21	0.96	0.92	29
6	We were able to benefit from type safety and increase our code quality	0.00	0.00	0.00	0.00	0.00	0

#	Field	Completely Agree		Somewhat Agree		Neither Agree or Disagree		Somewhat Disagree		Completely Disagree		Total
1	We were able to scale our projects faster with GraphQL	58.62%	17	37.93%	11	3.45%	1	0.00%	0	0.00%	0	29
2	We were able to speed up our development using GraphQL	75.86%	22	20.69%	6	3.45%	1	0.00%	0	0.00%	0	29
3	It was easier to unify our APIs (e.g. Microservices, legacy endpoints, third party services etc.). Type NA if you didn't use GraphQL in this way.	48.15%	13	29.63%	8	11.11%	3	7.41%	2	3.70%	1	27
4	Our architecture is more stable - We have less downtime and get fewer client errors.	44.83%	13	27.59%	8	27.59%	8	0.00%	0	0.00%	0	29
5	There has been an increase in performance using GraphQL	31.03%	9	24.14%	7	37.93%	11	6.90%	2	0.00%	0	29
6	We were able to benefit from type safety and increase our code quality	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0

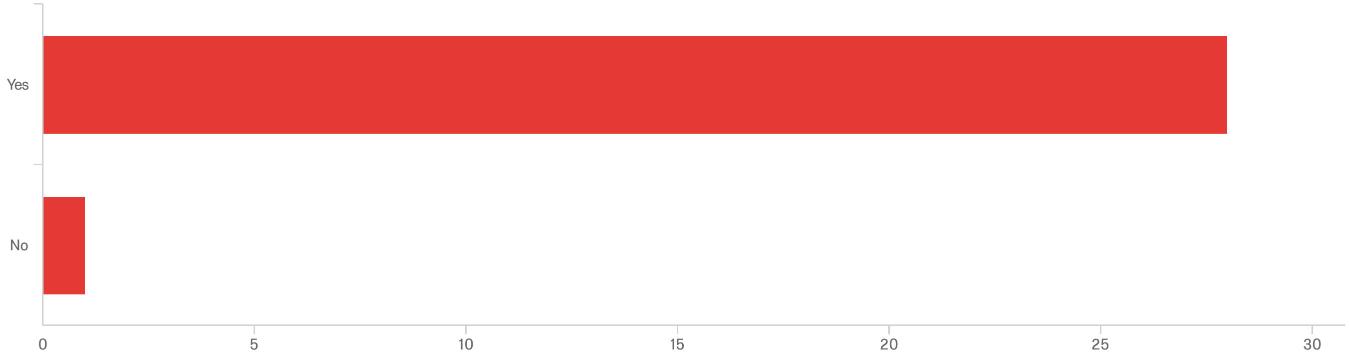
Showing rows 1 - 6 of 6

Q7 - Do you have any performance metrics on the following: (This is optional- but this will help others see validated results from adopting GraphQL)

API Latency Requests	# of Requests	# of Breaking Changes	Development Velocity
260ms		0	
Y	Y	N	N
NA	Na	Na	Na

Showing records 1 - 3 of 3

Q8 - Are you currently using GraphQL in production?

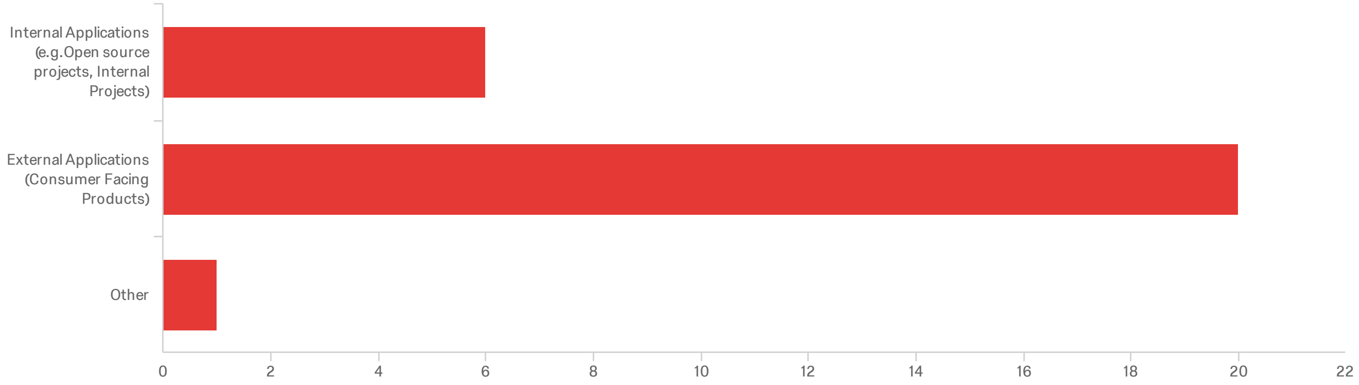


#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	Are you currently using GraphQL in production?	1.00	2.00	1.03	0.18	0.03	29

#	Field	Choice Count	
1	Yes	96.55%	28
2	No	3.45%	1
			29

Showing rows 1 - 3 of 3

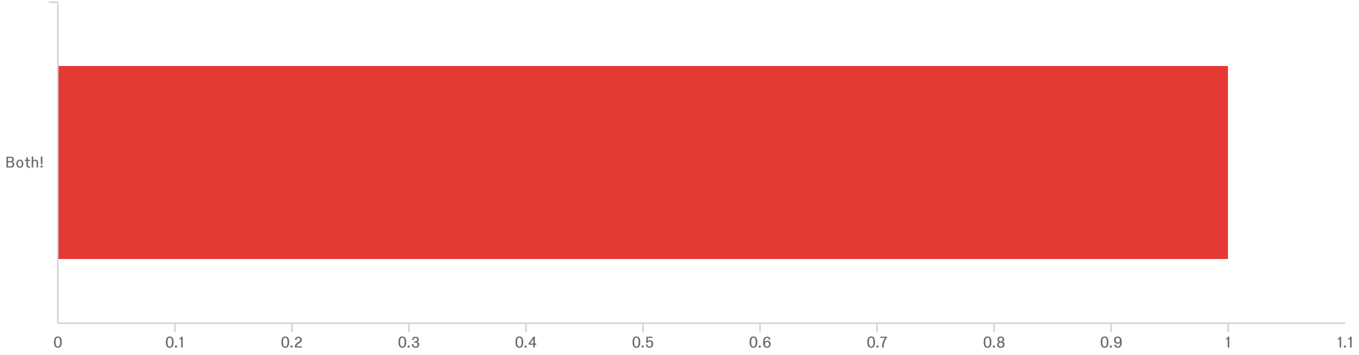
Q9 - If GraphQL is being used in production, is it being used for:



#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	If GraphQL is being used in production, is it being used for: - Selected Choice	1.00	3.00	1.81	0.47	0.22	27

#	Field		Choice Count
1	Internal Applications (e.g.Open source projects, Internal Projects)	22.22%	6
2	External Applications (Consumer Facing Products)	74.07%	20
3	Other	3.70%	1
			27

Showing rows 1 - 4 of 4



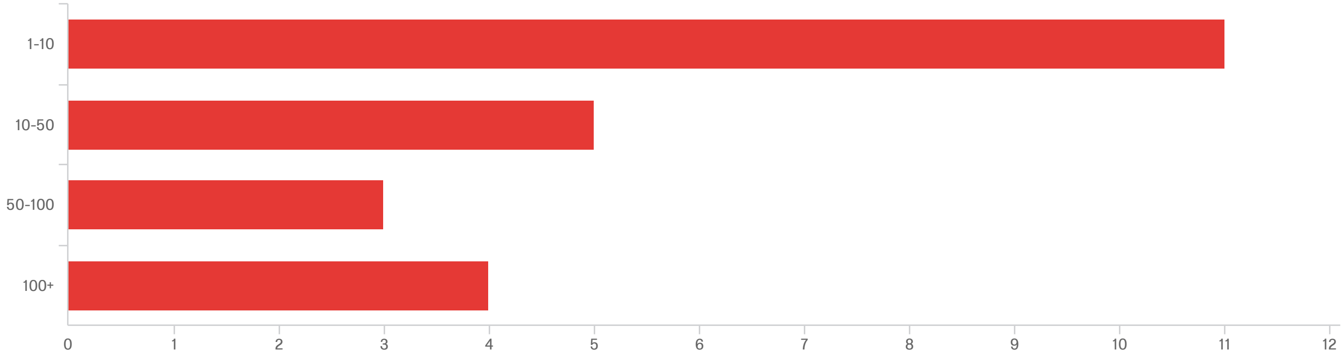
⚠

Data source misconfigured for this visualization.

#	Field	Choice Count	
1	Both!	100.00%	1

Showing rows 1 - 1 of 1

Q11 - What is the size of your engineering team?



#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	What is the size of your engineering team?	1.00	4.00	2.00	1.14	1.30	23

#	Field		Choice Count
1	1-10	47.83%	11
2	10-50	21.74%	5
3	50-100	13.04%	3
4	100+	17.39%	4
			23

Showing rows 1 - 5 of 5

Q10 - If you were to give another team advice on migrating from GraphQL, what would it be?

If you were to give another team advice on migrating from GraphQL, what wou...
It have some performance downsides but stills worth to scalability on the client side. The C# graphql library is completely awesome, one of the best choices that we have made
Look into best practices and do some demo projects to have a nice insight into how graphql works.
Remember that GraphQL is a very client-focused protocol, and is built around high latency connections. It's built for frontends, not backends. Suggestions: ● Remember that the GraphQL schema doesn't have to directly align to your backend services—it can, and should, expose data in a way that is most convenient to clients. ● Strongly consider letting your frontend teams own the GraphQL schema. After all, they're the ones interacting with it the most, and have a better idea of how clients would prefer accessing data. ● If you're in an environment with many internal services, don't follow the temptation to convert them to GraphQL. Bandwidth is cheap, and the added cognitive load and technical complexity load of GraphQL for service <-> service communication isn't worth it.
Analyze and adopt based on better mitigation plan
get everyone involved invest in tooling
Your GraphQL schema should mirror your business logic, not your database schema.
do it
Build it for the product not for the data dource
It's a great alternative to REST, but if you're dealing with a large database or multiple microservices, be prepared to use caching or Facebook's data-loader (open source project) in order to resolve performance issues that GraphQL doesn't solve out of the box. Primarily the N+1 issue posed the greatest challenge for us.
Understand the technology choices you make when using GraphQL. If using something like Apollo, get to know it's in and outs.
Strategize a plan to move your infrastructure incrementally over to graphql
Figure out your type safety story, generate types for your components for improved type safety. Start with one endpoint and see how it goes. GraphQL is great!
Consider using something like Prisma to avoid needing to scaffold out all the base CRUD operations all of your base models always end up needing eventually later anyways. Having to build them all yourself is very time consuming. That's not just a GraphQL thing, but in general it has saved a massive amount of time not needing to implement every pagination and sorting parameter for every new type we add.
Learn GraphQL on a greenfield project and adapt your way of thinking. Do not try to apply previous patterns/models on graphql
Research and understand resolvers before moving to GraphQL
Schema design is extremely important. The data should be in a format which is easily consumed by the front end without much processing. It might be a good idea to co-locate queries with components to keep component and data responsibilities obvious. Every query and type which will be stored in the cache must have a unique ID. Mutations and optimistic updates might not work the way you expect if you don't have them. Be conscious of repeated transactions within the same request.

If you were to give another team advice on migrating from GraphQL, what wou...

Pay great attention to how you model your domain. If you do not want your API to grow out of control, I would recommend paying good attention to how the domain is expressed in terms of a graph. Choosing a good list pattern (relay-style connections, arrays, etc.) early on is also a good practice, so that it can be consistent throughout the API.

dfjjosdjf shd f sflsdf sdfasdf dfasf

Showing records 1 - 18 of 18

Q11 - What is your name? (Optional)

What is your name? (Optional)
Wendel Nascimento
Ian MacLeod
Fei He
Perumal Palani
jon wong
Alessio Dionisi
Aditi
Raj
Kevin Grandon
Alex
Priyank Patel
Tillman
Thomas Lilley
Bertrand DUBAUT

Showing records 1 - 14 of 14

Q13 - What is your title?

What is your title? (Optional)
CTO
CTO
Software Developer
Principle Software engineer
MTS 2
staff software engineer
Full stack developer
Software Engineer
Software Engineer
CEO & Cofounder
Senior Software Engineer
Head of Backend
Software Engineer
Lead Engineer - Foundation Medicine
Head of Technology
CTO

Showing records 1 - 16 of 16

Q14 - What is your company name? (Optional)

What is the name of your company? (Optional)
Shawee
appointer GmbH
Convoy
Houzz
Paypal
coursera
Snaplytics
Satispay
Trulia
Novvum
Uber

Showing records 1 - 11 of 11

Q15 - Where is your company HQ?

Where is your company HQ?
São Paulo, Brazil
Hamburg
Seattle
Palo Alto
San Jose
mountain view
Copenhagen

Showing records 1 - 7 of 7

Q13 - Do you have any feedback for the survey?

Do you have any feedback for the survey?
should talk about the value of types and discoverability
Good job!
Type safety and quality were one of the biggest reasons we started exploring GraphQL and one of the best benefits.

Showing records 1 - 3 of 3

Q12 - If you are interested in the results of our study, leave your email here and we'll send you the report once it's completed. (Don't worry we won't send you spam!)

If you are interested in the results of our study, leave your email here an...
wendel.nascimento@shawee.io
florian@appointer.co
ianm@convoy.com
fei.he@houzz.com
j@jnwng.com
hello@adns.io
richardgirges@gmail.com
adigarg94@gmail.com
tyler@enrollsy.com
alex@sourceapp.io
priyank.patel@stackspace.ca

Showing records 1 - 11 of 11

End of Report