APOLLO

Expedia Group Transforms Product Development with Apollo

expedia group™ | Case Study

Expedia Key Stats

Scaling the benefits of GraphQL with a cross-brand data graph

25 Travel brands

200+ Travel booking sites in 70+ countries

>1 in 3 transactions booked on mobile

As the world's travel platform, Expedia Group manages customer experiences across more than 200 booking sites and 25 brands including Brand Expedia, Orbitz, Travelocity, Vrbo, Hotels.com and Egencia. Continuously improving these experiences requires the creativity and careful orchestration of thousands of developers and dozens of backend service teams. And because today's travelers increasingly use multiple touchpoints when planning, shopping and traveling, ensuring a seamless experience across an ever-growing set of teams, geographies, and capabilities poses a real challenge for Expedia Group.

"As you'd expect with a company of our history and scale, we operate multiple technology stacks, most built in-house, some added during acquisitions, and a few dating back to the dot-com era," said Dan Boerner, Distinguished Product Manager at Expedia Group.

expedia group™

A travel platform to bring the world within reach.

CHALLENGE

Legacy REST infrastructure isn't suited for modern apps, leading to slow development and inconsistencies across clients.

SOLUTION

Apollo Data Graph Platform to modernize the application architecture with GraphQL and a single cross-brand data graph.

RESULTS

- Faster development of products.
- Faster resolution of time-to-root-cause and time-to-fix live issues.
- Avoid weeklong outages caused by breaking API changes
- Less time to maintain consistency between platforms.

"As time progressed, that architectural complexity began to leak into the customer experiences, and teams found themselves duplicating work across platforms and managing ever-evolving service APIs rather than creating next-generation travel solutions for our customers."

Something had to be done. So a team within Brand Expedia set out to combine an ambitious vision with an agile approach to completely transform the way they build and deliver customer experiences.

Adopting a common data graph while moving away from heavy clients with tight bindings to REST APIs has been transformational. Features ship faster and it takes far less effort to deliver great customer experiences across each of their client platforms.

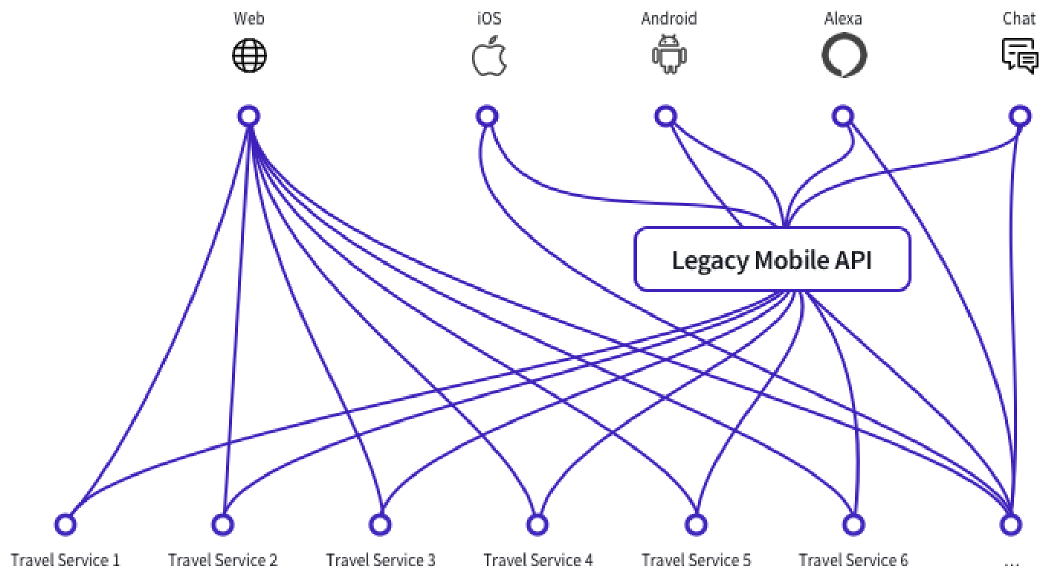
CHALLENGES OF A LARGE AND COMPLEX PLATFORM

Over the past decade, Brand Expedia group built a broad set of traditional REST services for its native and web clients. In this model, each client front-end was connected directly to a large number of underlying services.

While this approach worked, it slowed client teams down, as each team had to learn about every API, onboard, and then configure specific endpoints. With these APIs constantly evolving, improving, and moving to the cloud, it took more and more time for both client and service teams to make progress together.

The rise of native mobile clients added more complexity. Mobile-tuned APIs were created to serve those mobile applications, but that required an extra step each time a new capability was added to the core service. As a result, native apps ended up waiting for access to the latest features and the customer experiences began to diverge between clients. Also, because the service APIs weren't designed to deliver display-ready content, each client platform had to duplicate core business logic. Subtle differences in that logic, as well as client-specific experiments, all contributed to a diverging customer experience and a duplication of effort.

BEFORE ADOPTING A DATA GRAPH: CLIENT AND SERVICE COMPLEXITY



As Boerner explained, “Much of the mobile app team’s work had nothing to do with creating experiences for customers because they were constantly figuring out which API to call, onboarding those APIs, and dealing with version and endpoint changes. All of these distractions required additional resources and slowed our feature delivery."

Despite dedicated resources, the underlying problem persisted and continued to affect the customer experience. “The client-service complexity became a real bottleneck,” Boerner added. “Asking a native app team to keep track of tens of service teams so they can escalate production issues became a source of real frustration, and a real distraction for a team focused on creating great customer experiences."

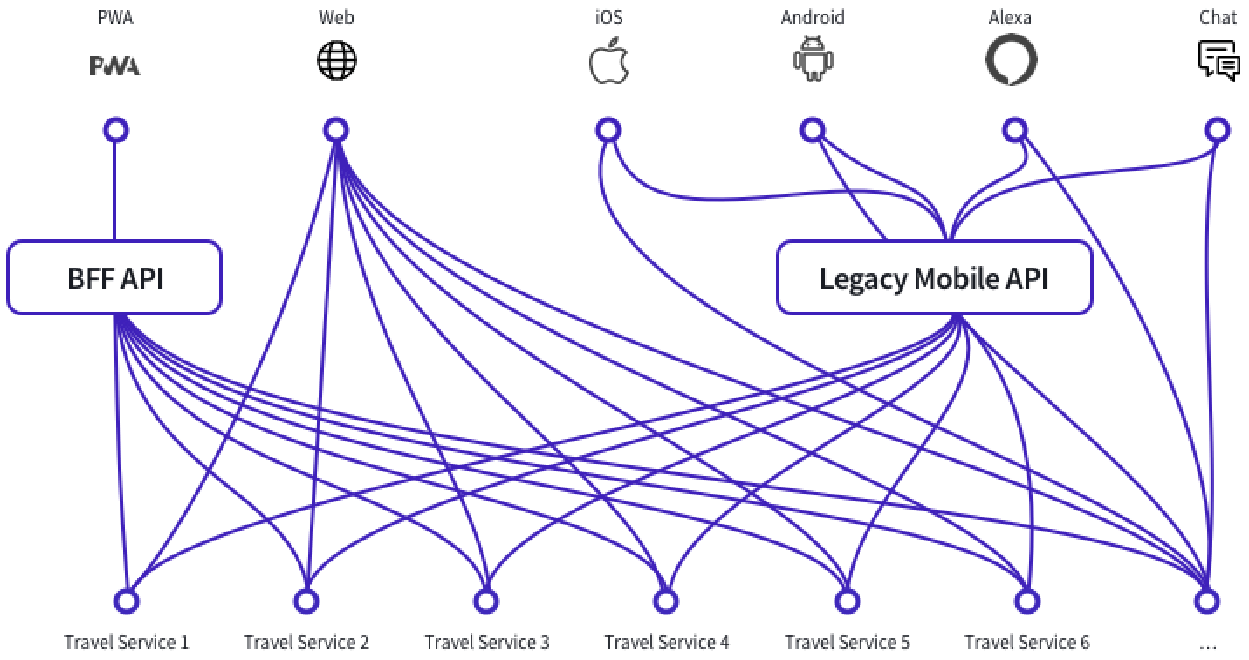
“Much of the mobile app team’s work had nothing to do with creating experiences for customers.”

Dan Boerner, Distinguished Program Manager at Expedia Group

EXPERIMENTING WITH A BACKEND-FOR-FRONTEND

When Brand Expedia group built their first Progressive Web App (PWA) for Expedia, Travelocity, and Orbitz brands, they followed the Backend-for-Frontend (BFF) pattern. The general approach held promise: let the BFF talk directly to the services and present a clean, non-service-specific API to the front-end. With this model, clients no longer had to include the business and orchestration logic and could enjoy an API built just for them.

EXPERIMENTING WITH BFF FOR A PROGRESSIVE WEB APP



It worked well initially, but the teams quickly learned that BFFs just aren't designed to scale. Since BFFs are tuned for a single client— in this case, the web—to make it work everywhere, Expedia Group would have had to write a BFF for each and every client platform, with duplicated logic and custom code in each. As Boerner explained, “when our native app team tried to replace the legacy mobile API with their own BFF, they found it difficult to keep up-to-date with all the test-and-learn experimentation going on in the web BFF. Ultimately, we decided it would be too time consuming to create a custom front-end service for each client and domain.”

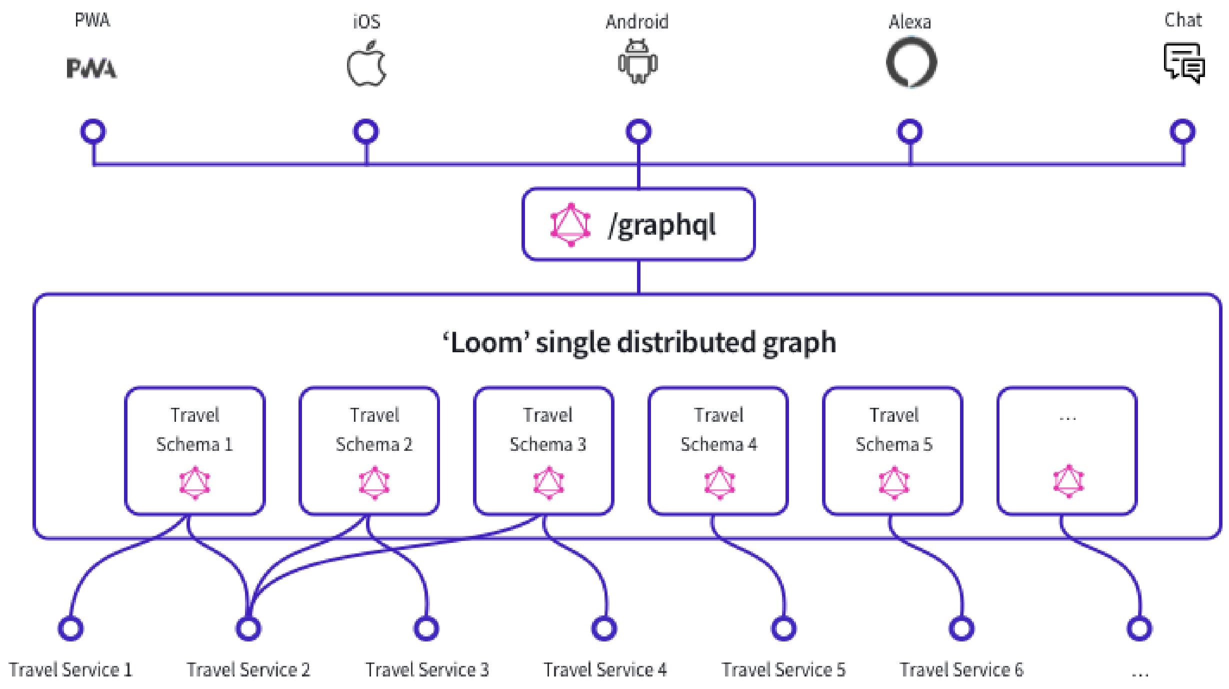
That’s when the team began to explore a data graph approach with GraphQL and Apollo.

THE MOVE TO ONE DATA GRAPH

With their BFF experience in hand, and the dual goals of moving the rest of the website pages to the new PWA platform and aligning capabilities and experiences between all client platforms, the team began to explore the use of GraphQL to connect the entire organization's app data and services into one central data graph. Their goal: a single graph, understood and discoverable by all teams, built to empower a new set of cohesive customer experiences.

With a single entry point for clients to connect to, client teams were freed from directly connecting to multiple services or managing communications across multiple, extended teams. Meanwhile, service owners could easily expose their data via a schema that allowed clients to retrieve exactly the portion of data needed with a simple query.

AFTER ADOPTING A DATA GRAPH: A FEDERATED SERVICES ARCHITECTURE TO SCALE



Expedia Group turned this effort into a formal initiative, which now serves as the foundation of their new and modern application development stack, and supports their strategy to create seamless customer experiences across all their client platforms.

"When your client code doesn't need to know which underlying service provides the data, all kinds of good things start to happen, for both the client and service teams," said Boerner. "Client teams are freed from complexity and service teams have a single contract to maintain, which frees them up to evolve their back-ends faster and with less risk."

Though given a broad charter, the team knew from the start that they needed an iterative approach to building and operating this single data graph. It needed to have all the benefits of a single data graph, but without requiring that a single central team manage such a large schema. So, aligned with the "one graph, federated implementation" architecture described at principledgraphql.com, Brand Expedia group set out to distribute ownership of the data graph.

“If our small platform team had to build all the tech we use from Apollo Platform, we'd have much less time to help onboard new teams, evangelize our design principles, or ultimately solve our customer problems.”

Dan Boerner

The first effort by the team was to develop a schema stitcher leveraging Apollo Server called the “Loom,” which worked with their open-source graphql-kotlin to stitch together GraphQL services for each domain schema into a single graph. The Loom also allowed simple schema linking between common schema objects like Geography. While this initial stitching implementation worked, the schema linking required custom code for each link. When Apollo announced their Apollo Federation, a new architecture for composing multiple GraphQL services into a single graph, the team was immediately intrigued.

“Apollo Federation will allow us to implement our existing schema linking in a safe, declarative manner, while opening up a whole new level of broader federation opportunities across the wider enterprise,” explained Boerner.

MIGRATION TO APOLLO GRAPH MANAGEMENT

At scale, the team knew that expanding the data graph across and within teams would require powerful management capabilities to improve the development experience and protect the graph. “As our initial GraphQL rollouts began to show promise, the power of the technical approach was clear. Equally clear was the need to have all the powerful monitoring, management, and operational capabilities built into our platform to support tens of teams and hundreds of developers,” said Boerner.

“If our small platform team had to build all the tech we use from Apollo Platform, we'd have much less time to help onboard new teams, evangelize our design principles, or ultimately solve our customer problems.”

Dan Boerner

“When we learned about Apollo’s Data Graph Platform and the Apollo Graph Manager, it became a classic build-vs.-buy decision for us. Apollo gave us the tools we didn’t have, while allowing us to integrate with our existing operational infrastructure."

For Expedia Group, leveraging tools and tech from Apollo freed up the team for other priorities. As Boerner put it, "If our small platform team had to build all the tech we use from Apollo Platform, we'd have much less time to help onboard new teams, evangelize our design principles, or ultimately solve our customer problems.”

Apollo provided Expedia Group with a complete graph management solution, giving them a means to monitor, detect, and prevent schema breaks, as well as operationalize the data graph—all at scale. It also makes a unified data graph possible by sharing management and portable code, queries, and fixes across teams.

Since adopting Apollo, code complexity has decreased dramatically. “Our new PWA clients have a tiny fraction of the business logic of our older clients because we’ve moved it behind the graph where it’s shared by all clients,” said Boerner. “This also means that fixes made in the shared layer are picked up by all clients, so that’s a big win in terms of reducing technical debt and fulfilling our mission to create a seamless customer experience, regardless of the client platform.”

DEVELOPER EXPERIENCE AND PRODUCTIVITY IMPROVES

Almost immediately after adopting the Apollo platform, Expedia Group saw an improvement in the speed and efficiency of the developers working with the single graph and GraphQL. Ben Munge, lead developer on Expedia Group’s Native App iOS Team, saw how Apollo’s platform enabled developers to thread the graph through tools they were already using.

“The power of the Apollo tooling to reduce developer effort and ensure that the service code is always a true reflection of the schema made it a great option for us,” he said. “Using generated code has greatly reduced the number of bugs and minimized the development and testing effort compared to implementing this integration manually. We develop our queries in Xcode using the Apollo GraphQL plugin. We also leverage GraphQL Playgrounds to explore the schema and develop queries. Once a query is set, we use the Apollo iOS tools to update our schema and regenerate the API.”

Munge also uses the platform to track down issues and monitor the health of services during debugging. “There were a lot of pluses to the Apollo platform,” he added. “Speed was one of them. By partnering with Apollo, we can greatly reduce the time to create new experiences and to resolve time-to-root-cause and time-to-fix live issues.”

KEEPING THE GRAPH HEALTHY AND EFFICIENT

Expedia Group recognized the data graph as an ideal central location to collaborate across product teams, so they took the vital step of creating a new group to create and manage their growing graph community. This became a key part of their agile API strategy.

Using Apollo’s data graph platform, developers across the organization can monitor the health of the graph and discover how back-end services are used.

“We can take the graph metrics from Apollo and plug them into our operational fabric, Grafana, Splunk, etc.” said Boerner. “But those tools are designed for general purpose, and we find that Apollo’s dashboard gives a better 360-degree GraphQL-native view of clients, operations, schema changes, and failures so that our developers get a much clearer view of how our graph is performing. We also see reductions in the time it takes to determine root cause during a service disruption, as Apollo Graph Manager allows us to quickly see exactly which clients and queries are affected.”

It also became important for managing changes to the underlying services and schemas, so that any individual change from a team wouldn’t break existing queries—or the customer experience. “Since customers run a variety of app versions, we have to ensure that even older clients are still compatible with our graph,” said Boerner. “To protect these clients, we use Apollo’s Github integration coupled with Schema Validation to ensure backward compatibility at all times by validating all schema changes against the client queries from the previous week of client requests.”

“Schema validation at PR commit is a game changer. We've already detected and prevented a number of schema changes that would have led to weeklong service outages for our native app customers.”

Dan Boerner

In the past, teams would make an API change on behalf of web clients and inadvertently break native apps. As Boerner explained, "Schema validation at PR commit is a game changer. We've already detected and prevented a number of schema changes that would have led to weeklong service outages for our native app customers. We modeled loss reductions from schema validation and were surprised at how quickly the technology would pay for itself."

It’s these types of central operations that ensure the data graph is a reliable resource even as it evolves.

WHAT’S NEXT

“GraphQL is becoming the standard front-end API for all our customer-facing applications, and teams are working on external partner APIs that leverage GraphQL as we strive to leverage technologies that create consistency and simplicity.”

Today, Expedia Group is expanding the use of the data graph across more teams, which is a great opportunity for everyone in the organization to think in terms of a truly unified customer experience.

"As we work to leverage and share more capabilities across Expedia Group, GraphQL and Federation are strategically important tools." said Boerner.

GraphQL is also enabling opportunities to align across different product types. “Across all lines of business, we are identifying the common capabilities that allow a user to execute a search, browse and filter results, view details, and make a purchase,” said Ben Munge, Senior iOS Software Engineer. “The service architecture needs to be designed with this unified experience in mind. Why should the response structure differ based on the product we’re selling? Imagine if Amazon had separate services for household goods, electronics, and books.”

Looking ahead, the core team is exploring advanced capabilities like server-driven UI, powered by a graph that dynamically adjusts the user experience across all clients based on machine learning models.

"We’re excited about the opportunities to connect our data graph with our Unified Design System to power a set of more scalable user experiences that allow server-side experimentation,” said Tyler Fleck, Director of User Experience. “Trying out new ideas across all our clients without the need to make any client-specific changes speeds up the design and learning process, and ultimately leads to better customer experiences and outcomes.”

“GraphQL is helping us bring the world within reach for millions of travelers," said Boerner. We believe we are just scratching the surface of what this transformation will unlock for our customers.”

Dan Boerner

What started as a bottom-up project with a small team focused on building a proof-of-concept multi-tenant graph has grown both intentionally and organically as teams throughout the company demonstrated the value of GraphQL and the Apollo Platform. Expedia Group is now focused on building on the groundswell of adoption across mobile web, iOS, and Android; onboarding service teams to publish their capabilities to the graph; and demonstrating the value across the company and across the customer experience. “GraphQL is helping us bring the world within reach for millions of travelers," said Boerner. We believe we are just scratching the surface of what this transformation will unlock for our customers.”