Practical API Design

Ronnie Mitra
ronnie.mitra@publicissapient.com
@mitraman

publicis
sapient

Publicis Sapient | Digital Business Transformation

As a digital business transformation partner of choice, we've spent nearly three decades utilising the disruptive power of technology and ingenuity to help digitally enable our clients' business in their pursuit of next

Our scale

20,000
passionate people

50+
offices globally connect

30
years of digital pioneering and customer innovation

Our clients

Nationwide Building Society, AVIVA, Lloyds Banking Group, HSBC, T-Mobile, Goldman Sachs, Marriott, SAMSUNG, Mercedes

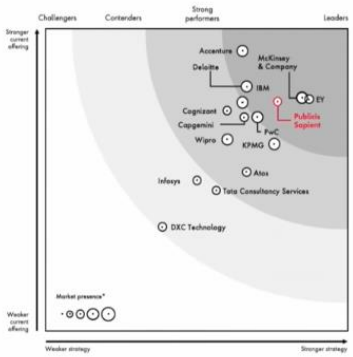
Serving you

A startup mindset and agile methods to unlock value in ways that delight your customers and improve their operational effectiveness

A transformation approach that is grounded in a view of both the company and the customer simultaneously

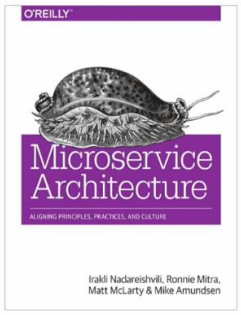
A unique fusing of strategy and consulting, experience and engineering with an enduring culture of problem-solving creativity

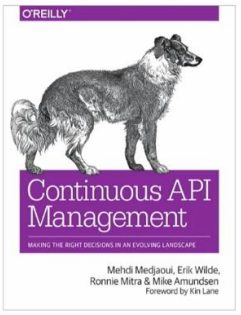
Industry Recognition



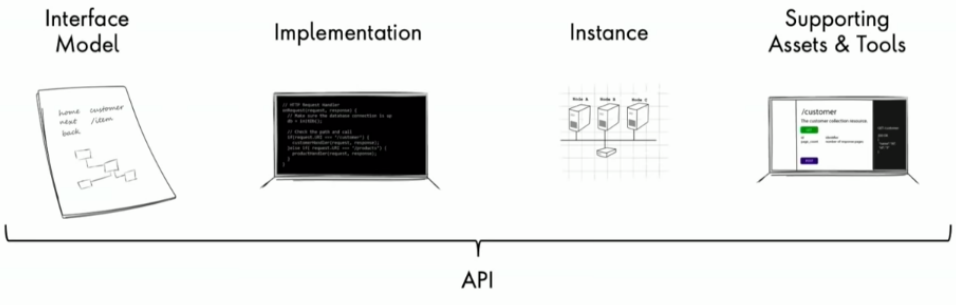
Forrester Global Digital Business Transformation Accelerators – Q1 2019

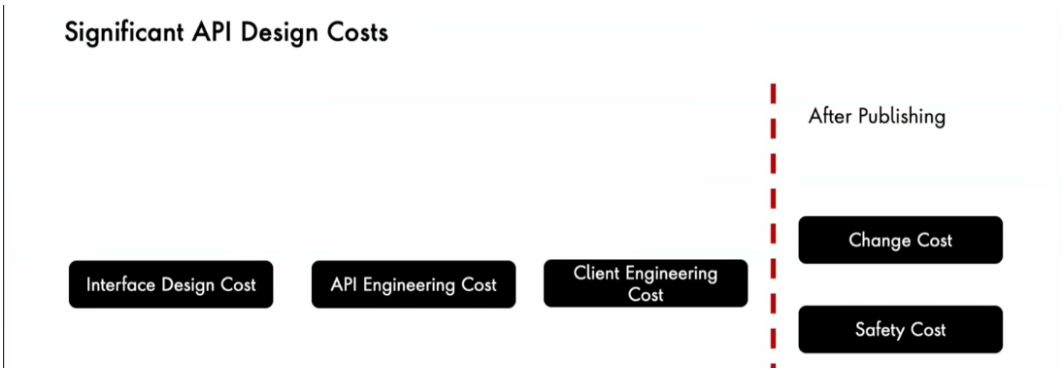
Ronnie Mitra





The Scope of API Design



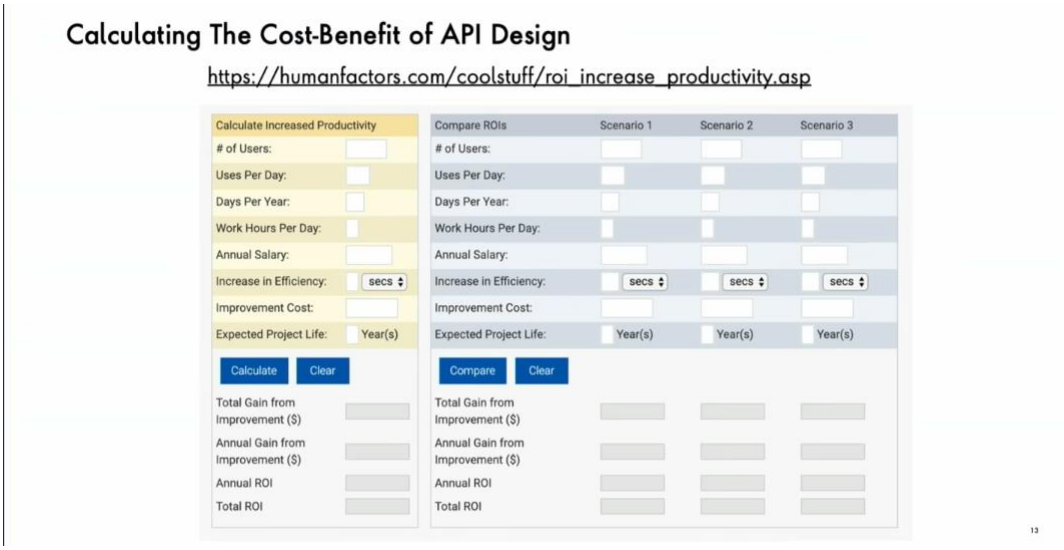


How much time and energy do you need to design and operationalize your API.

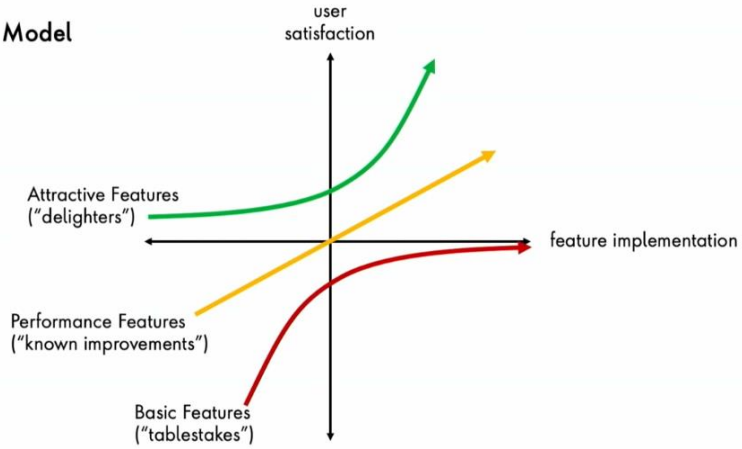
7 Practical Techniques For API Design

Technique #1 Set The Right Design Goals

The API goals you have will affect your ROI for designing the API

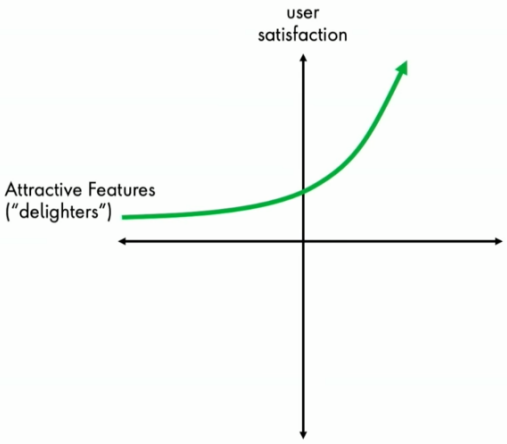


The Kano Model

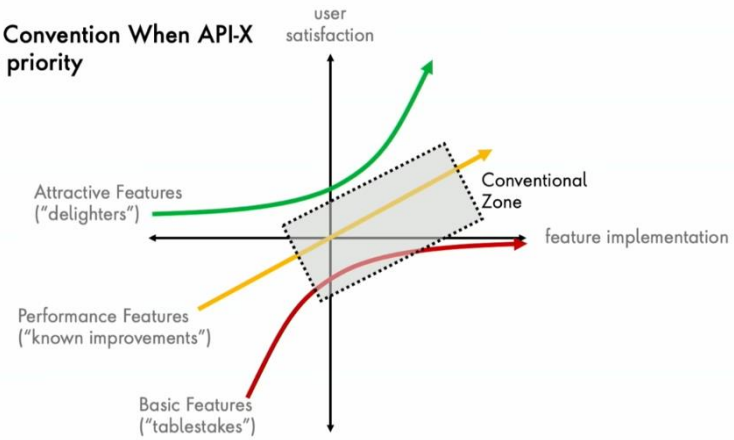


Invest in Attractive Features when API-X is a key success factor

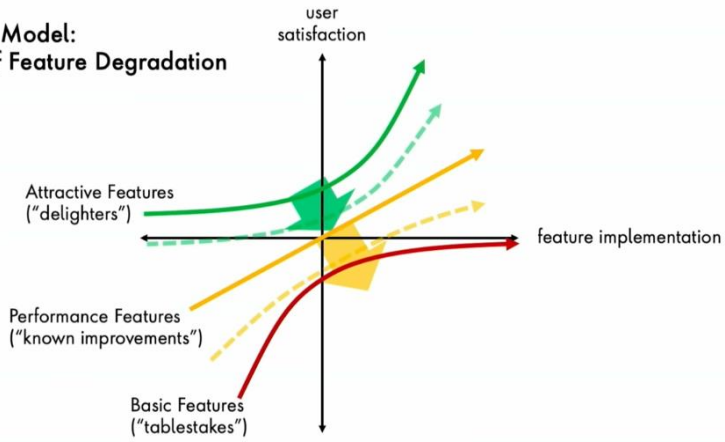
- Increased Conversion Rates
- Brand Marketability
- Talent Retention



Focus on Convention When API-X is not the priority



The Kano Model: Beware of Feature Degradation



Use Imitation as a Shortcut to a Conventional API

Save time by using another API design as inspiration

Considerations:

- Who are the API's users?
- What domain does it operate in?
- What is it like to use?

Changes

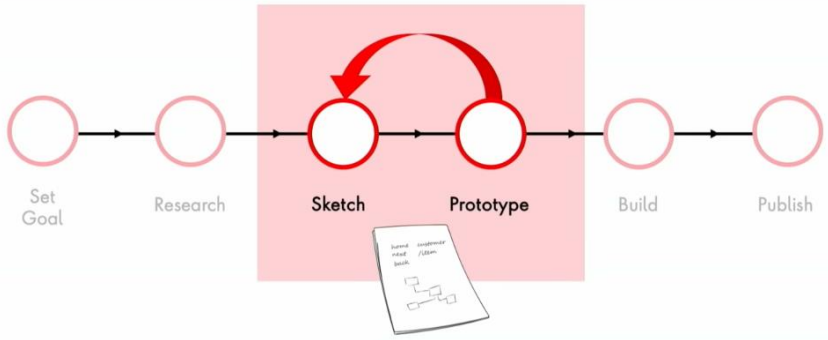
For Changes Resource details, see the resource representation page.

Method	HTTP request	Description
URIs relative to https://www.googleapis.com/drive/v3, unless otherwise noted		
getStartPageToken	GET /changes/startPageToken	Gets the starting pageToken for listing future changes.
list	GET /changes	Lists the changes for a user or shared drive. Required query parameters: pageToken
watch	POST /changes/watch	Subscribes to changes for a user. Required query parameters: pageToken

GET /changes
GET /changes/watch

Technique #2
Sketch & Prototype Iteratively

Sketch & Prototype



Sketch at first, don't start programming right away. You need a way to prototype or sketch your API, ask for feedback

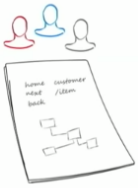
Technique #3
Heuristic Evaluation

API Design Reviews

Just like a code review, your API design can benefit from evaluation by other experts and your peers.

Practical Challenges:

- Access to API design experts
- Getting comprehensive feedback
- Collating analysis from multiple experts



Jakob Neilsen and Rolf Molich: 10 Usability Heuristics for User Interface Design

1. Visibility of System Status
2. Match Between System and the Real World
3. User Control and Freedom
4. Consistency and Standards
5. Error Prevention
6. Recognition rather than recall
7. Flexibility and Efficiency of Use
8. Aesthetic and Minimalist Design
9. Help Users Recognize, Diagnose, and Recover from Errors
10. Help and Documentation

7 Usability Heuristics for API Design

1. **Visibility of System Status**
2. **Match Between System and the Real World**

   User Control and Freedom
3. **Consistency and Standards**
4. **Error Prevention**

   Recognition rather than recall
5. **Flexibility and Efficiency of Use**

   Aesthetic and Minimalist Design
6. **Help Users Recognize, Diagnose, and Recover from Errors**
7. **Help and Documentation**

5 Usability Heuristics for Machine Interface Design

5 Usability Heuristics for Machine Interface Design

5 Usability Heuristics for Machine Interface Design

5 Usability Heuristics for Machine Interface Design

1. Visibility of System Status
2. Consistency and Standards
3. Error Prevention
4. **Flexibility and Efficiency of Use**
5. Help Users Recognize, Diagnose, and Recover from Errors

Does the interface model support both beginner and advanced use cases?

Are their optimizations and accelerators available?

5 Usability Heuristics for Machine Interface Design

1. Visibility of System Status
2. Consistency and Standards
3. Error Prevention
4. Flexibility and Efficiency of Use
5. **Help Users Recognize, Diagnose, and Recover from Errors**

Is error information accurate and helpful?

Does it address both human and machine concerns?

Example of a Heuristic Analysis

REQUEST

```
POST /LongRunningJob
```

RESPONSE

```
HTTP 200 OK
{
    "status": "running"
}
```

Visibility:

- "Use 202 instead"
- "Provide a link where client can check job status and add some info about job length"

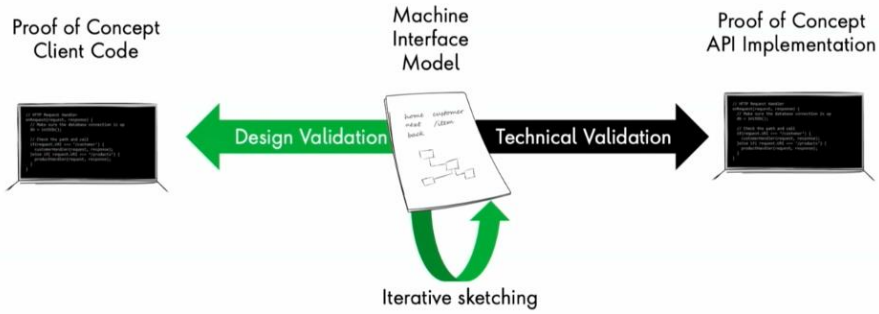
Consistency & Standards:

- "Use our standardized words for job status ("in-progress")"

Find Usability Problems by Combining Results

Technique #4
Write Code

Writing Code in the Design Phase



You need to write client code as PoCs as part of the design process

Technique: Write Code

"Code the use-cases against your API before you implement it, even before you specify it properly"

– Joshua Bloch

Technique: Write Client Code

Write code from the perspective of your users early in the API design cycle.

```
request('http://musiclibrary.api/songs/14', function (err, res, body) {
    let title = body.song.title;
    let artists = body.song.artists;
    let releaseDate = body.song.releaseDate;

    showCover(body.song.album.img);
});
```

Tips for Using Client Code Effectively

Be your user

Utilize languages, frameworks and techniques that you think your users would use.

Unit tests aren't enough

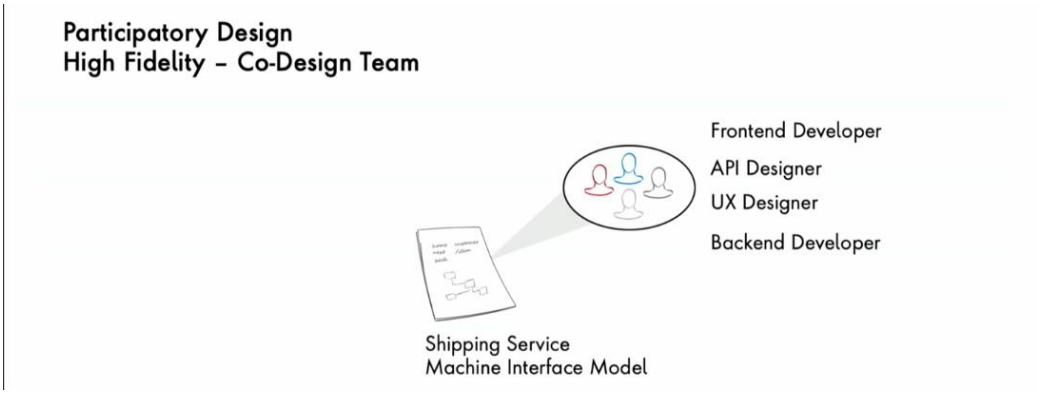
Write code that accomplishes a goal from a user perspective – not code that tests a spec.

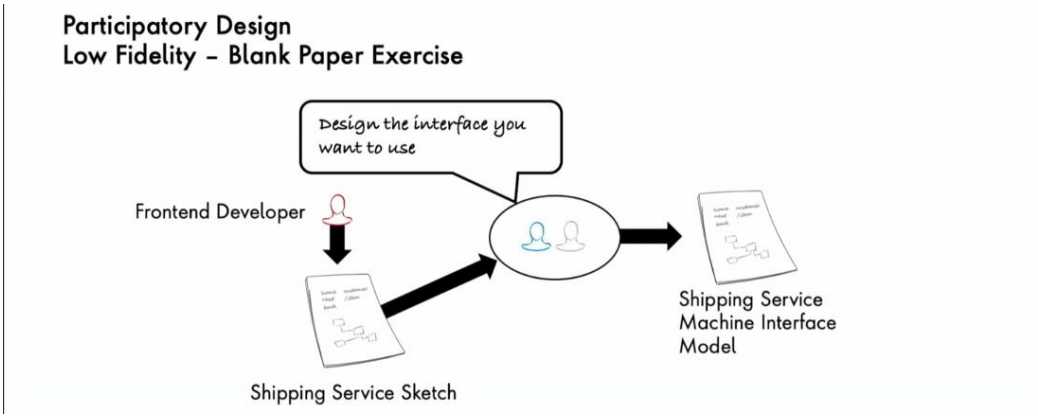
Focus on insight not syntax

Don't get caught investing too much time making code compile or worrying about code completeness.

Technique #5
Participatory Design

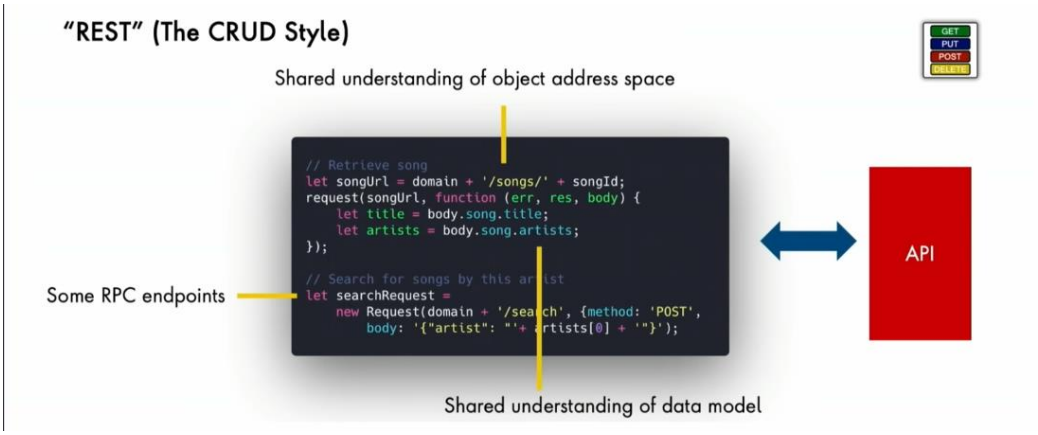
This is a way to bring our user to the table and ask for their needs and how they use the API's interface model

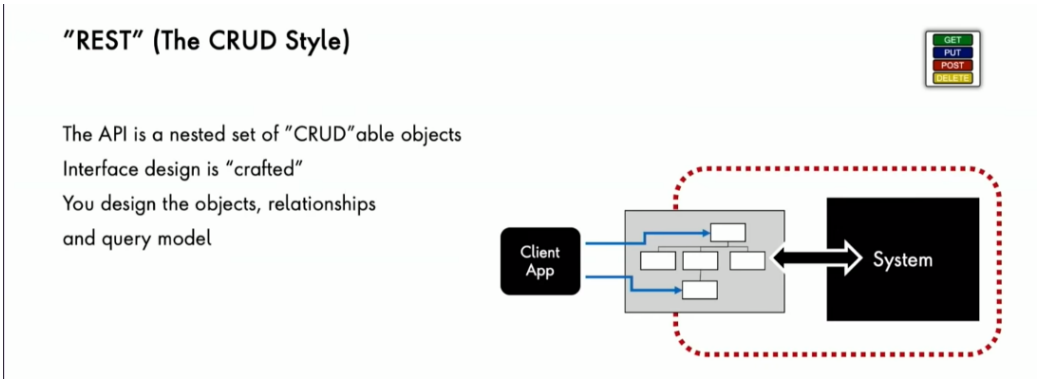




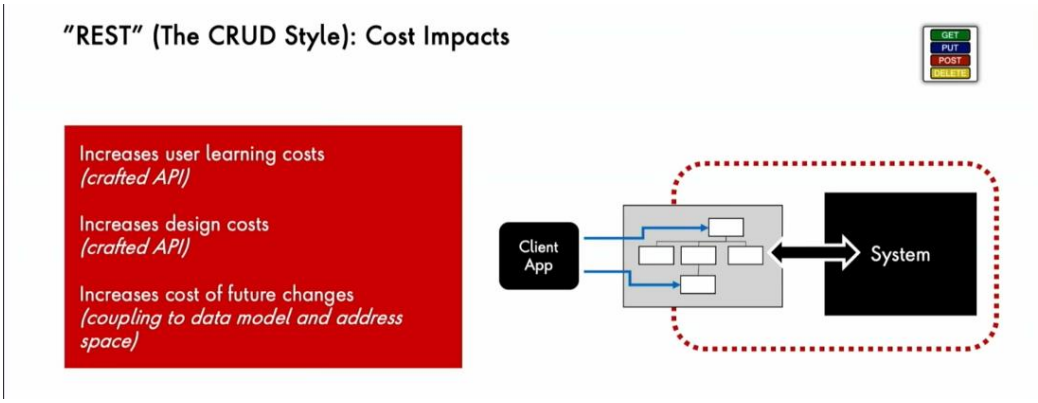
Technique #6
Choose a Style That Fits

How your API looks, the xtics it has and the constraints that it conforms to should be your own, like RPC style, GraphQL style, CRUD style APIs.

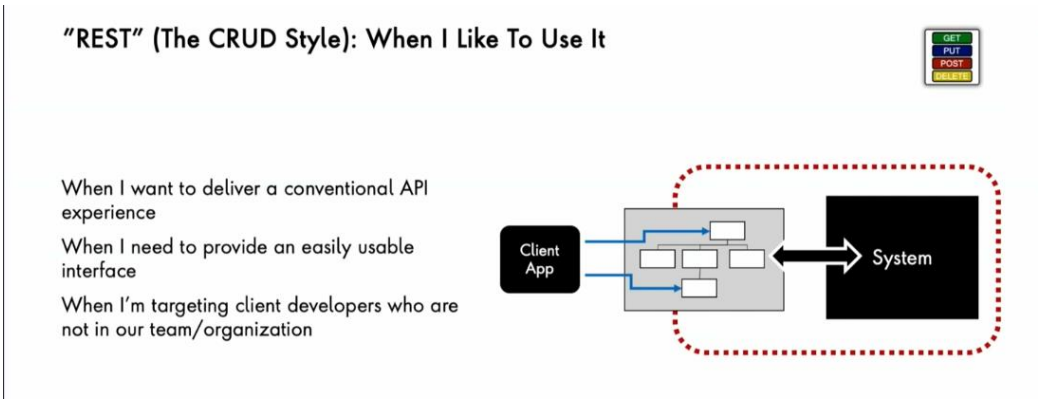




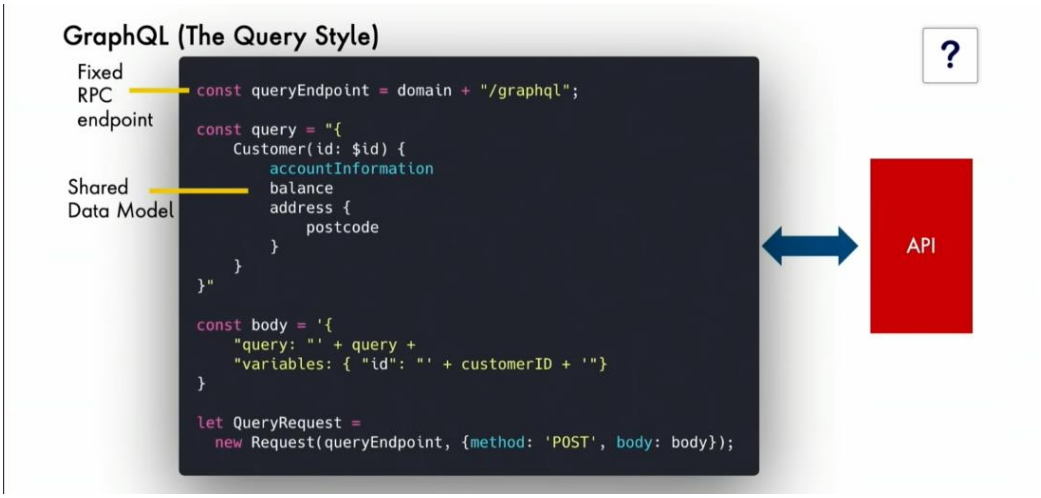
We need to make design decisions like what the object graph should look like, what the message format should look like, what the query parameters names should be and how to filter them, what to paginate, what the size of a data piece should be, what status codes to use for each API call, etc.



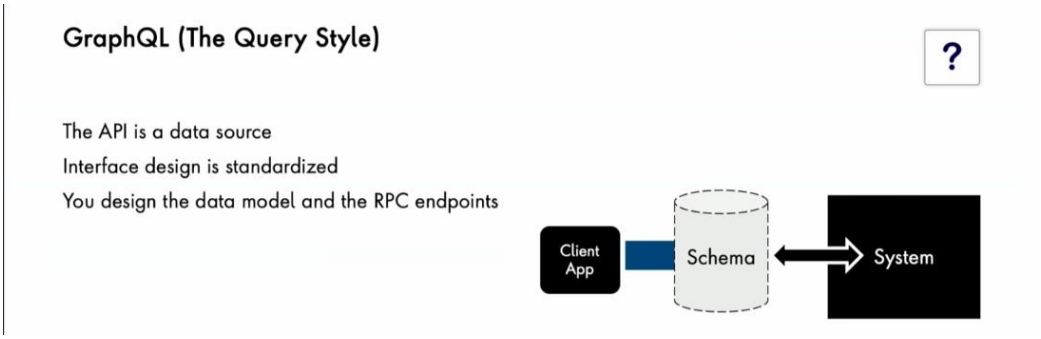
To reduce the learning cost for the users of our APIs, we need things like documentation and supporting assets.



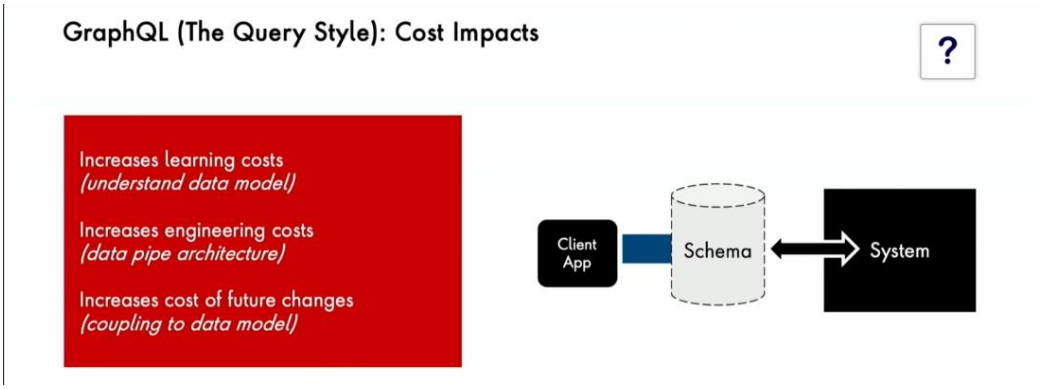
We need to think about what the API is for, is it about branding, productivity gains, usability, or access? We then aim for the right sweet spot like the CRUD style API.

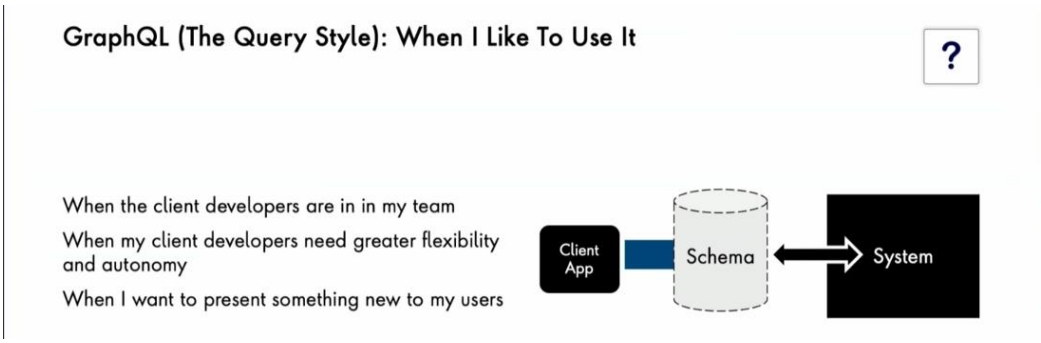


The client code needed for GraphQL looks really different, it has RPC endpoints and the user needs to know a lot about the data model to manipulate.



The API is now more of a data source.





API Styles – User Metaphors

Tunnel-RPC Style — The API is a local library

CRUD Style — The API is a set of data objects

Hypermedia Style — The API is a website

Query Style — The API is a database

Event Driven Style — The API is a notification message

Technique #7
Make Practical Design Decisions

Example

"What should we return when `GET /songs?genre=classical` doesn't produce a match?"

Resolving API Design Decisions

1. How reversible is this design decision?

If its easy to reverse we can afford to make a less optimal decision and improve it later.
This is debt that is easy to pay back.

"Once we decide on this, it's going to be difficult to change. We'd have to release a new version."

Resolving API Design Decisions

2. What do the specifications and standards say?

If there are clear rules, endeavor to follow them.

"We've read RFC 7231, now we are starting to think 404 is the way to go."

Resolving API Design Decisions

3. What would the client code look like?

Write client code to test your hypothesis and gain insight

"Actually, now it seems like a 200 with an empty collection makes the most sense!"

Seven API Design Techniques

1. Manage Your Debt
2. Build a Conventional Product (when it makes sense)
3. Perform Heuristic Evaluations
4. Write Code
5. Use Participatory Design
6. Choose a style that fits
7. Make Practical Design Decisions

Bill Moggride on Design

"If there's a simple, easy design principle that binds everything together, it's probably about starting with the people"