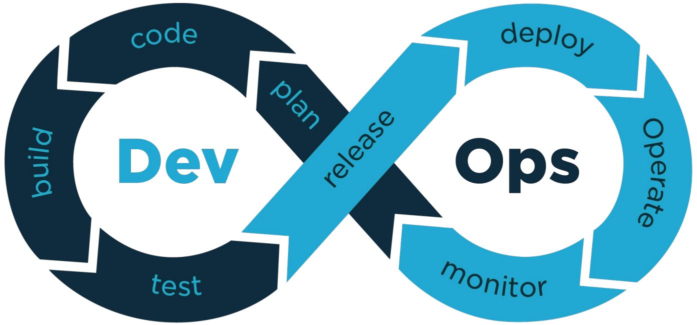
Monolithic vs Microservices Architecture

Diego Leon & George Rezkalla

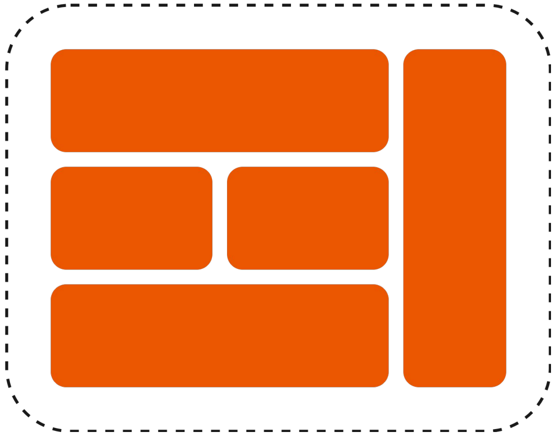
Agenda

Motivation

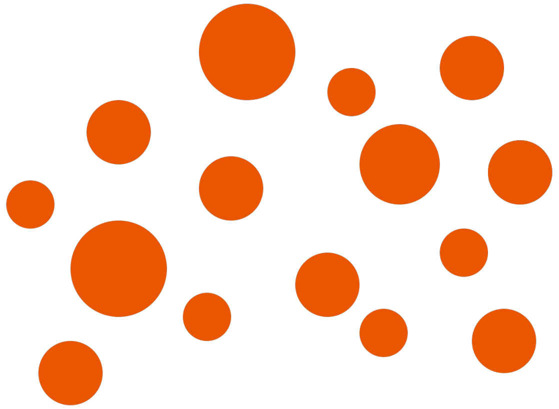
Microservices Architecture enables Continuous Deployment



Monolithic Architecture



Monolith



Microservices

Microservices Architecture

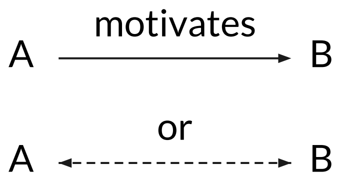
Decompose the application into smaller, interconnected units (services)

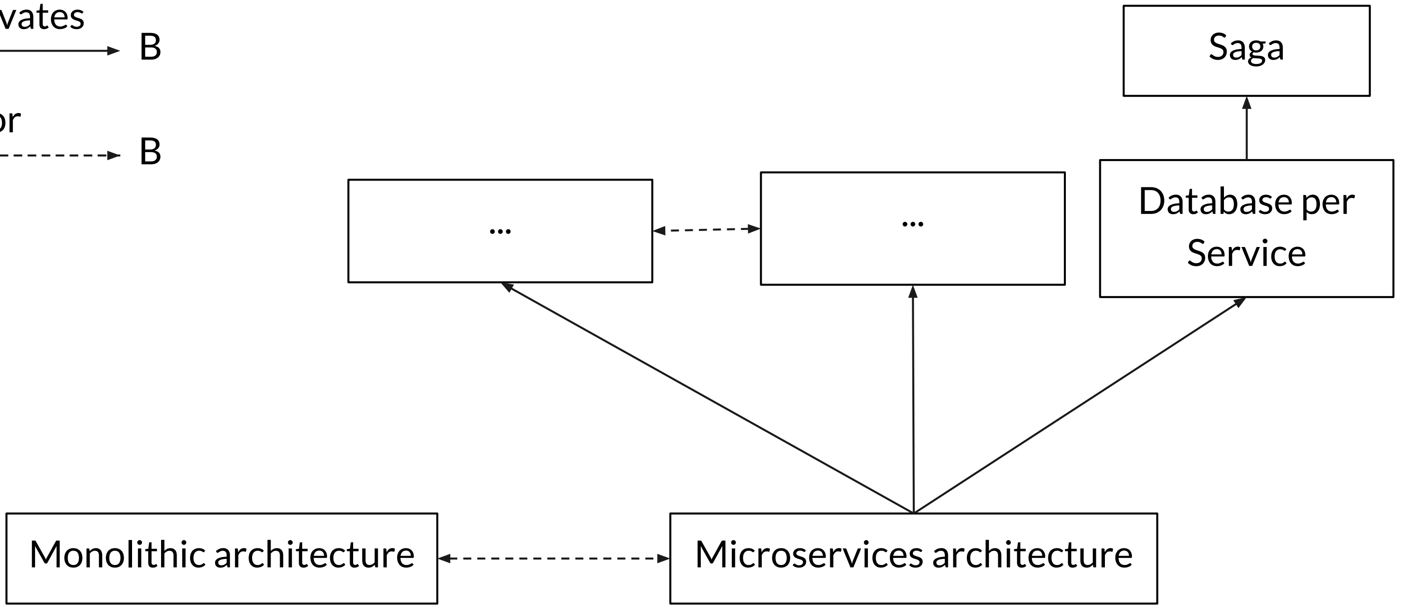
How to decompose applications?

Strategies (patterns):

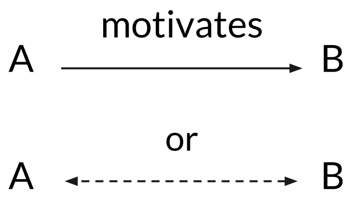
- Decompose by use case
- Decompose by resources
- And more

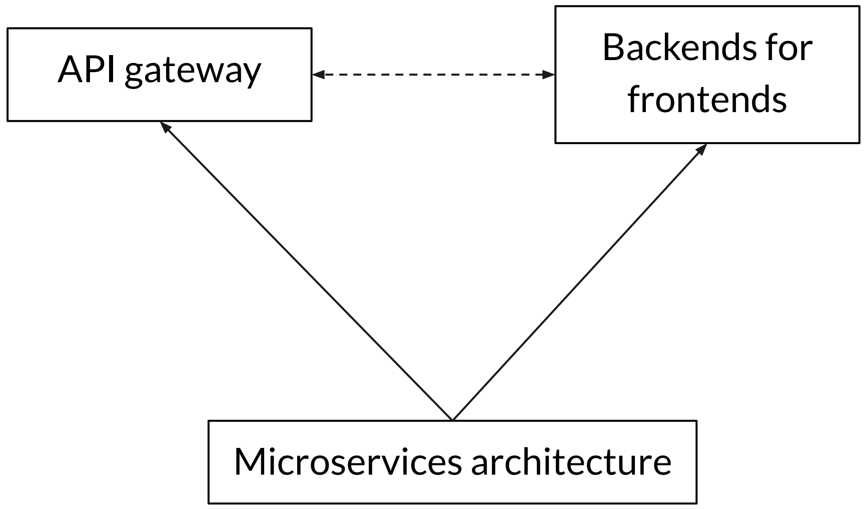
Patterns as a Graph



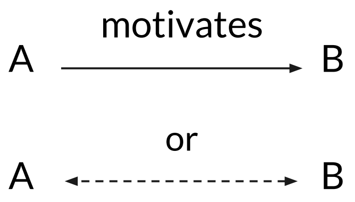


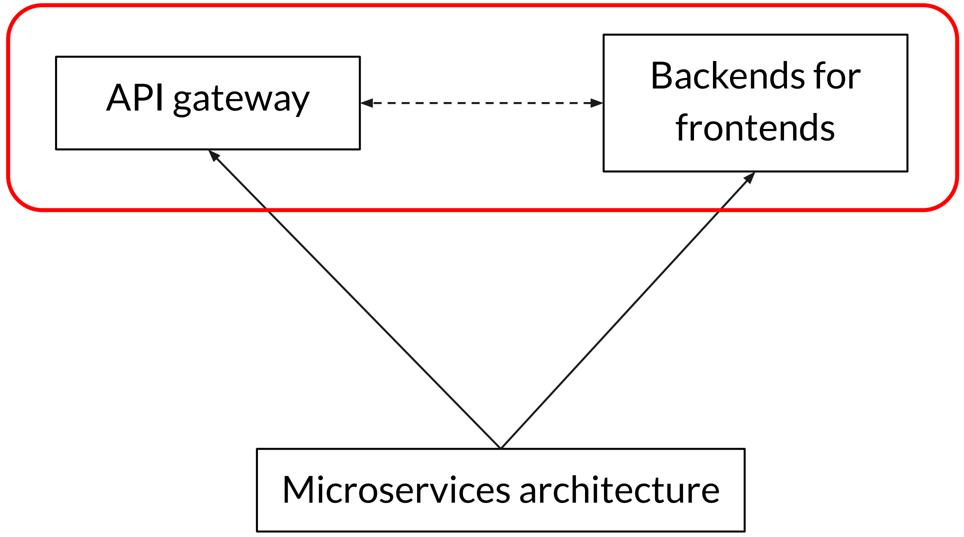
Patterns Comparison - Technical Example





Patterns Comparison - Technical Example



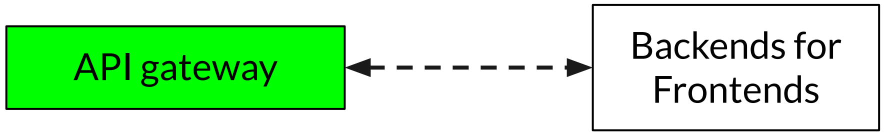


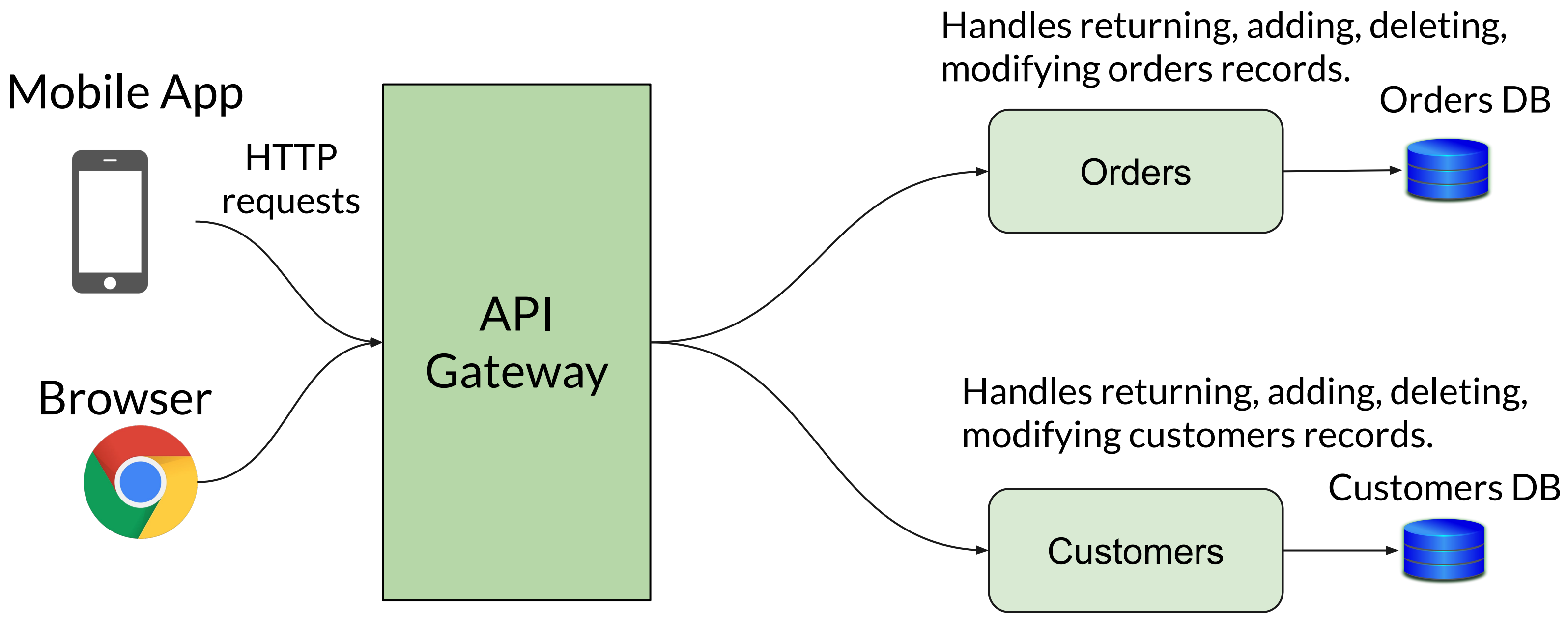
Tables

Orders DB		
Order_ID	Customer_ID	Total
...	...	...
2	31	1500
3	32	400
4	32	500
...	...	...

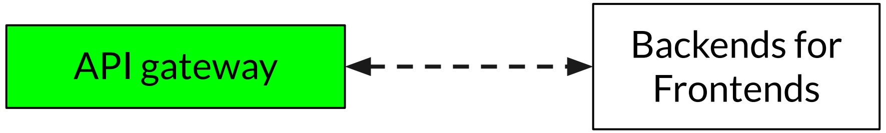
Customers DB	
Customer_ID	Customer_Name
...	...
31	Cust1
32	Cust2
...	...
...	...

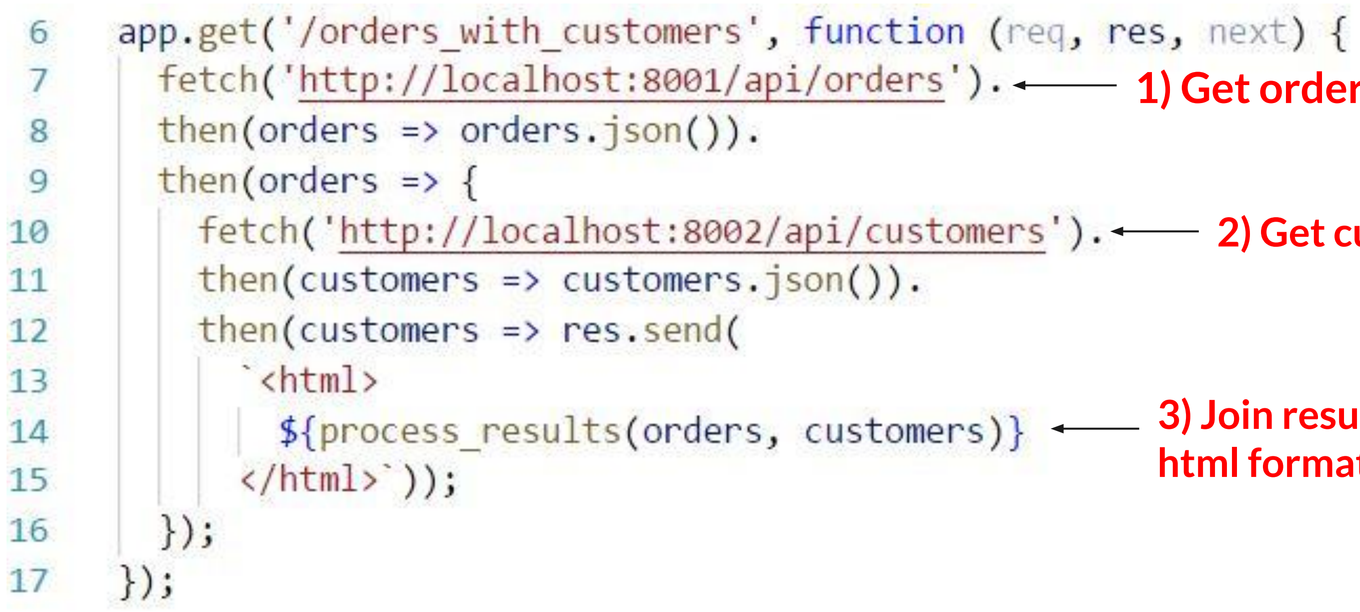
API Gateway


API gateway
Backends for Frontends


Mobile App
HTTP requests
Browser
API Gateway
Handles returning, adding, deleting, modifying orders records.
Orders DB
Orders
Handles returning, adding, deleting, modifying customers records.
Customers DB
Customers

API Gateway





```
app.get('/orders_with_customers', function (req, res, next) {
  fetch('http://localhost:8001/api/orders').
  then(orders => orders.json()).
  then(orders => {
    fetch('http://localhost:8002/api/customers').
    then(customers => customers.json()).
    then(customers => res.send(
      `<html>
        ${process_results(orders, customers)}
      </html>`));
  });
});
```

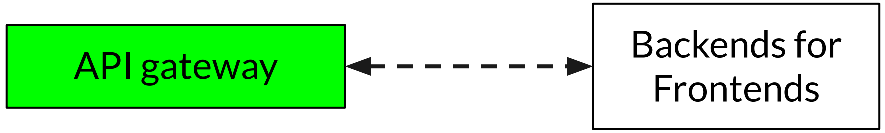
1) Get orders.

2) Get customers.

3) Join results and reply in html format.

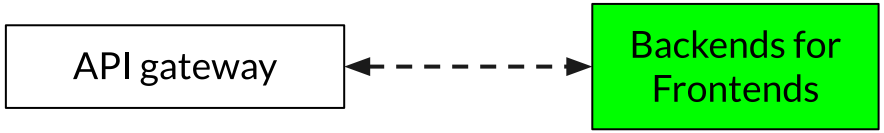
Code snippet for Shared API gateway

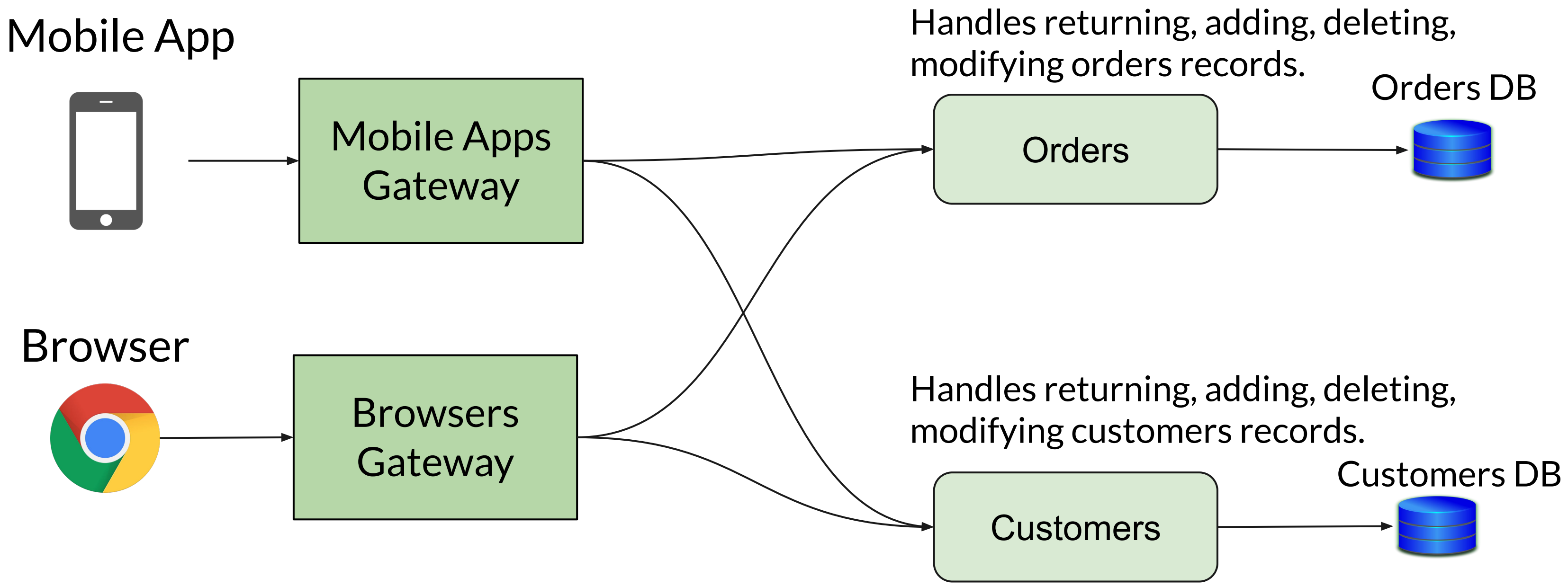
API Gateway - Notes



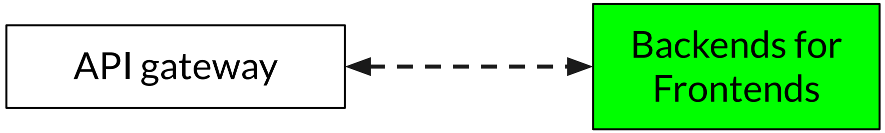
- This code is simplified.
- One API Gateway can return appropriate responses to calls from multiple user agents (e.g. browsers, mobile phones).

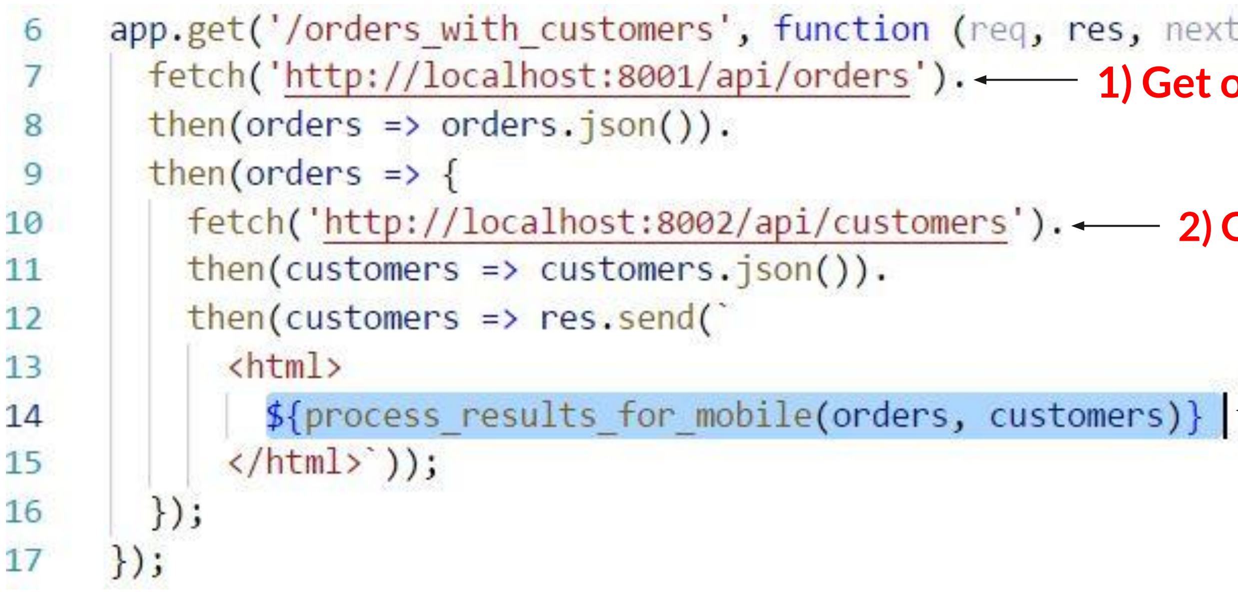
Backends for Frontends (BFF)




Mobile App
Mobile Apps Gateway
Browser
Browsers Gateway
Handles returning, adding, deleting, modifying orders records.
Orders DB
Orders
Handles returning, adding, deleting, modifying customers records.
Customers DB
Customers

Mobile App Gateway





```
app.get('/orders_with_customers', function (req, res, next) {
  fetch('http://localhost:8001/api/orders').
  then(orders => orders.json()).
  then(orders => {
    fetch('http://localhost:8002/api/customers').
    then(customers => customers.json()).
    then(customers => res.send(`
      <html>
        ${process_results_for_mobile(orders, customers)}
      </html>`));
  });
});
```

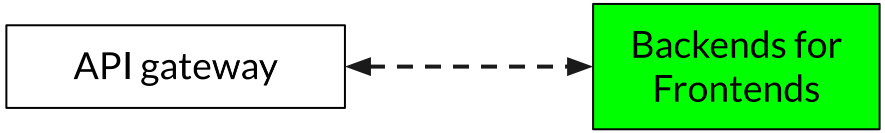
1) Get orders.

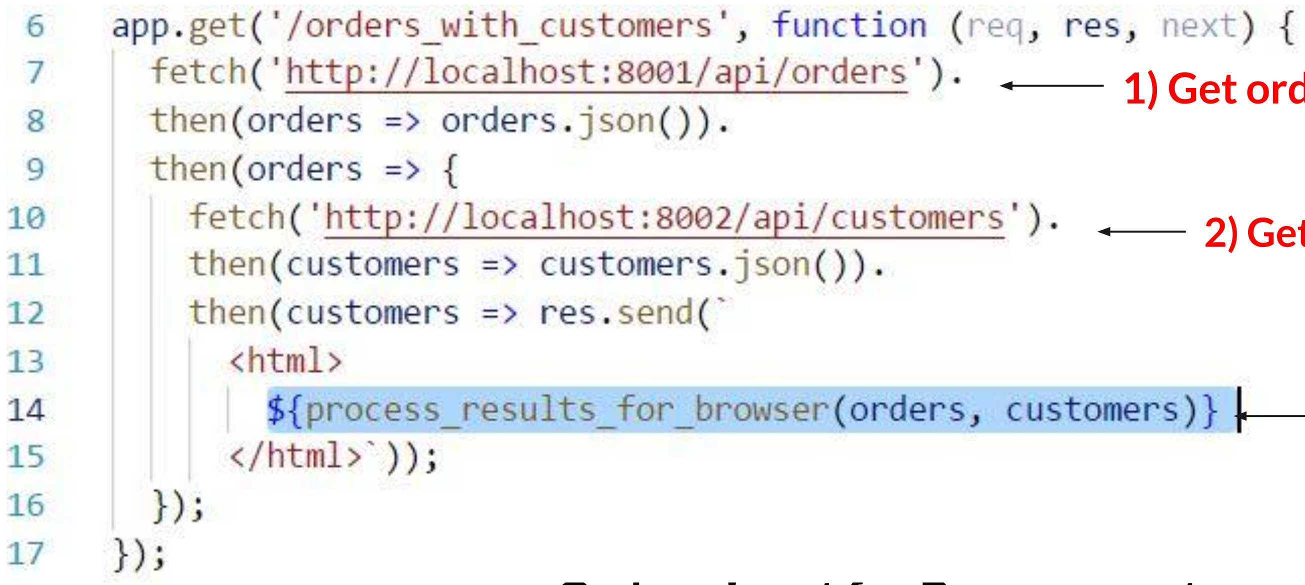
2) Get customers.

3) Join results in MOBILE App format and reply.

Code snippet for Mobile Apps gateway

Browsers Gateway





```
app.get('/orders_with_customers', function (req, res, next) {
  fetch('http://localhost:8001/api/orders').
  then(orders => orders.json()).
  then(orders => {
    fetch('http://localhost:8002/api/customers').
    then(customers => customers.json()).
    then(customers => res.send(`
      <html>
        ${process_results_for_browser(orders, customers)}
      </html>`));
  });
});
```

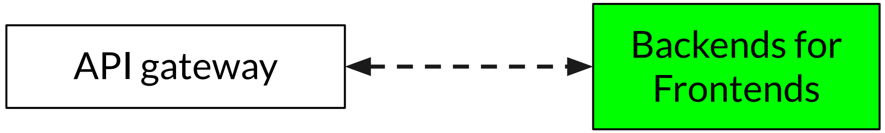
1) Get orders.

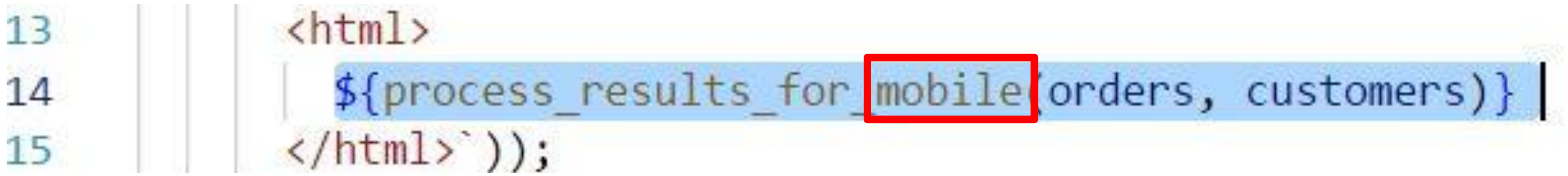
2) Get customers.

3) Join results in BROWSER format and reply.

Code snippet for Browsers gateway

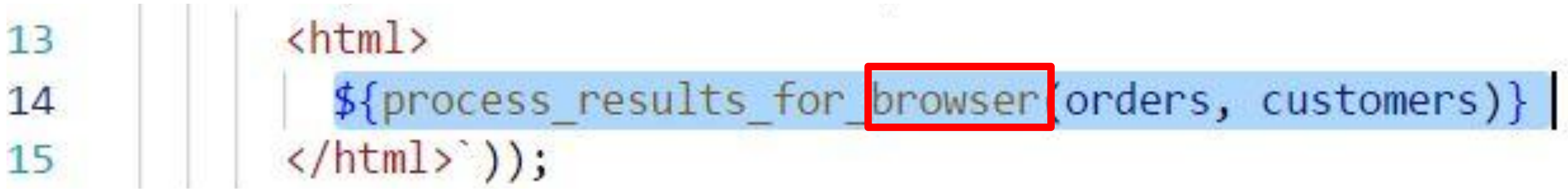
Mobile Apps vs Browsers





```
<html>
  ${process_results_for_mobile(orders, customers)}
</html>`));
```

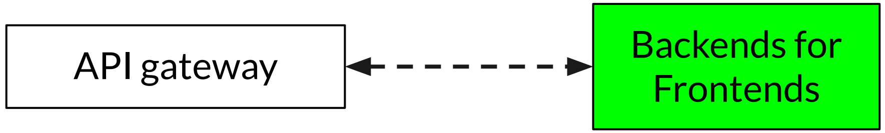
Code snippet for Mobile Apps gateway



```
<html>
  ${process_results_for_browser(orders, customers)}
</html>`));
```

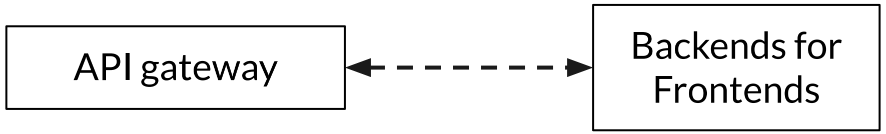
Code snippet for Browsers gateway

Mobile Apps vs Browsers



The rest of the code is the same.

Reflection - API Gateway vs BFF



	API Gateway	Backends for Frontends
Code duplication	Less	More
Codebase size	More	Less

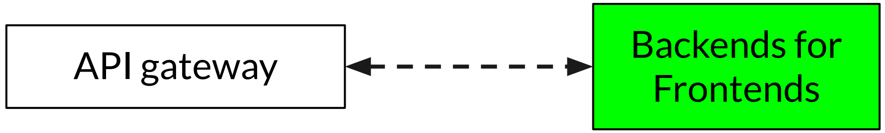
Either solve this:

Harder to solve.
Code becomes larger with time.

Or that:

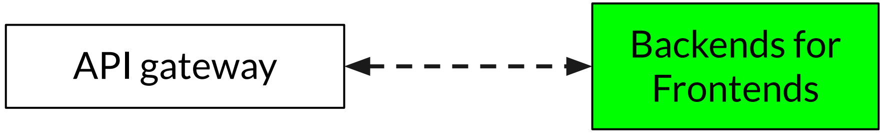
Easier to solve.
How?

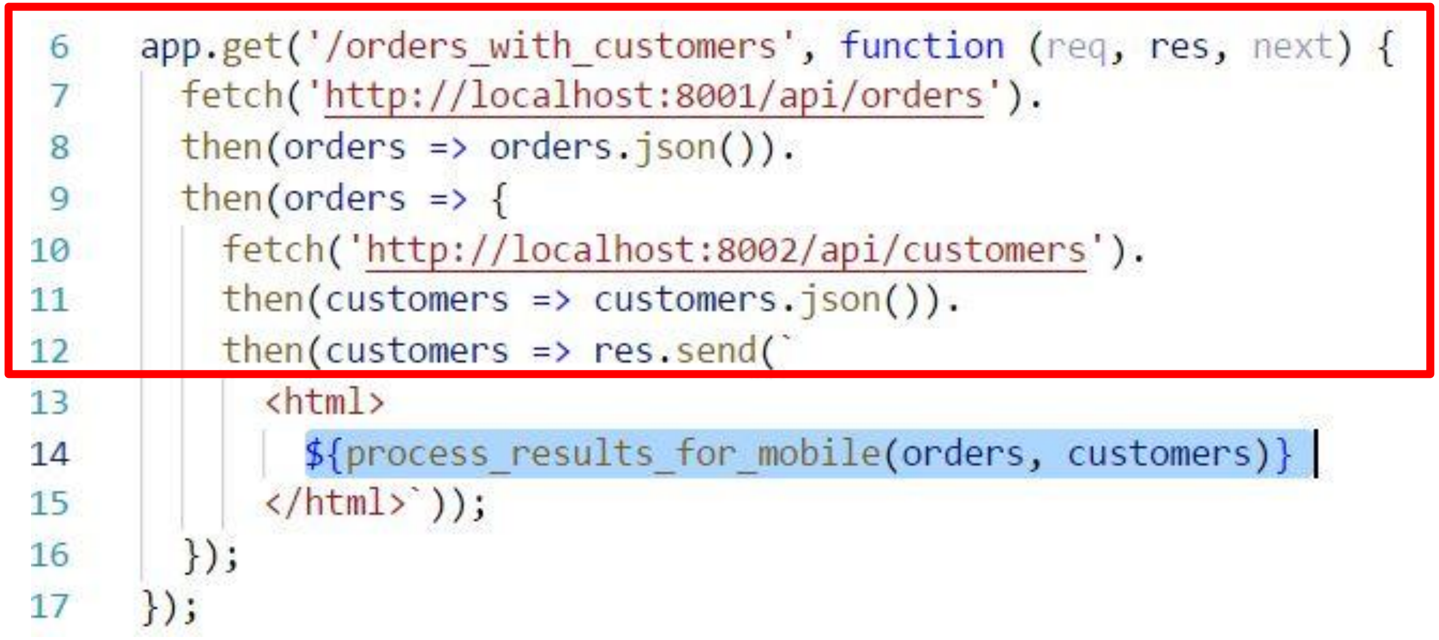
Solution to BFF Code Duplication



Create a shared library and replace duplicate code with it.

Mobile Apps Gateway

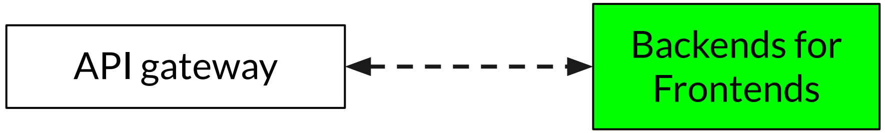


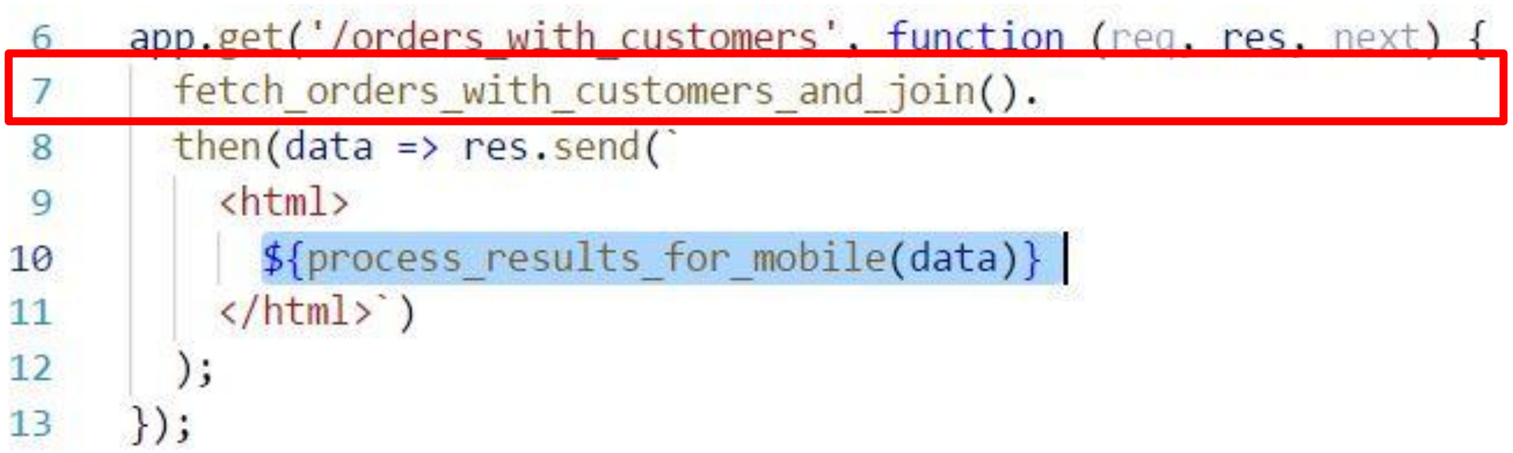


```
app.get('/orders_with_customers', function (req, res, next) {
  fetch('http://localhost:8001/api/orders').
  then(orders => orders.json()).
  then(orders => {
    fetch('http://localhost:8002/api/customers').
    then(customers => customers.json()).
    then(customers => res.send(`
      <html>
        ${process_results_for_mobile(orders, customers)}
      </html>`));
  });
});
```

Code snippet for Mobile Apps gateway - with code duplication

Mobile Apps Gateway

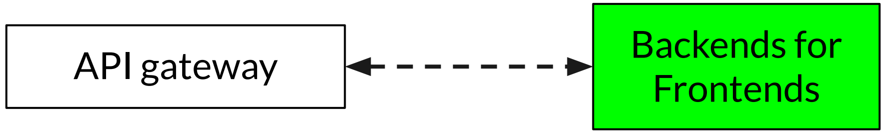


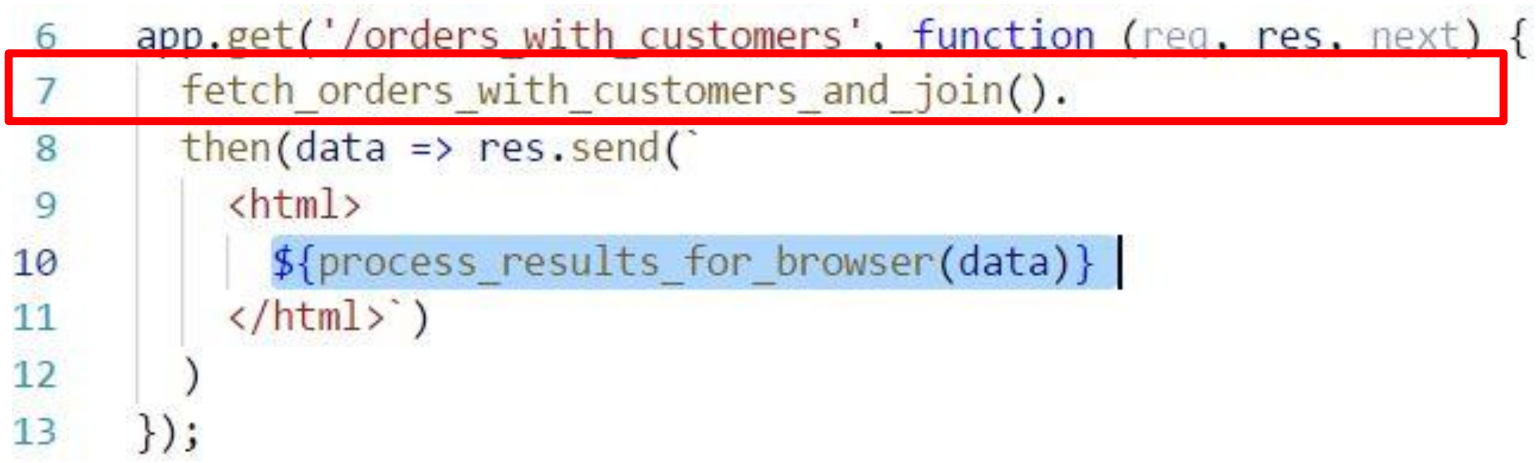


```
app.get('/orders_with_customers', function (req, res, next) {
  fetch_orders_with_customers_and_join().
  then(data => res.send(`
    <html>
      ${process_results_for_mobile(data)}
    </html>`)
  );
});
```

Code snippet for Mobile Apps gateway - with NO code duplication

Browsers Gateway

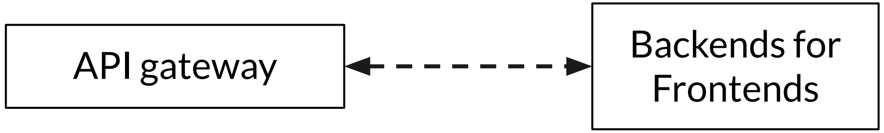




```
app.get('/orders_with_customers', function (req, res, next) {
  fetch_orders_with_customers_and_join().
  then(data => res.send(`
    <html>
      ${process_results_for_browser(data)}
    </html>`)
  )
});
```

Code snippet for Browsers gateway - with NO code duplication

Reflection - API Gateway vs BFF

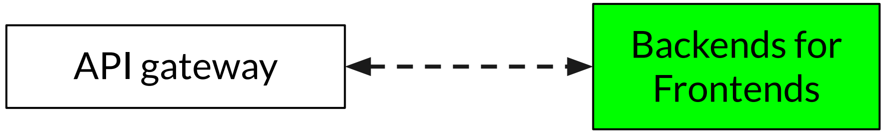


	API Gateway	Backends for Frontends
Code duplication	Less	More
Codebase size	More	Less

Either solve this:
Harder to solve

Or that:
Easier to solve

Reflection - API Gateway vs BFF



	API Gateway	Backends for Frontends
Code duplication	Less	Less
Codebase size	More	Less

Either solve this:

Harder to solve

Solved

What you’re probably thinking

Take-home message

- Microservices architecture is a design pattern implemented using other patterns.
- There is no one-fits-all solution/pattern.

Thank you for listening!
Any questions?