API Workshop

API Recommendations and API Gateways

O'REILLY®
Software Architecture

Agenda

- Recommendations for Building APIs
- Handling Multiple APIs
- The role of gateways
- API Centric Architecture
- Role of API Management
- Demo Spring Cloud Gateway

Recommendations for Building APIs

- Use API guidelines document that fits your business
  - e.g. PayPal or Microsoft
  - Provides structure that all can follow
  - Build consistent APIs

Recommendations for Building APIs

- Use HTTP Verbs correctly
  - Follow the recommended RFC 7231
- Understand idempotency guidelines
  - If you send the same request multiple times what is consequence?

Recommendations for Building APIs

- Avoid this

Developer | Use cases | Products | Docs | More | Labs | Dashboard

Search all documentation...

Basics

Accounts and users

Tweets

- Post, retrieve and engage with Tweets
- Get Tweet timelines
- Curate a collection of Tweets
- Optimize Tweets with Cards
- Search Tweets

Post, retrieve and engage with Tweets

Overview | Guides | API Reference

The following API endpoints can be used to programmatically create, retrieve and delete Tweets, Retweets and Likes:

Tweets	Retweets	Likes (formerly favorites)
• POST statuses/update	• POST statuses/retweet/:id	• POST favorites/create/:id
• POST statuses/destroy/:id	• POST statuses/unretweet/:id	• POST favorites/destroy/:id
• GET statuses/show/:id	• GET statuses/retweets/:id	• GET favorites/list
• GET statuses/oembed	• GET statuses/retweets_of_me	
• GET statuses/lookup	• GET statuses/retweeters/ids	

Recommendations for Building APIs

- Version your APIs from the outset
  - Difficult to migrate users if a public API
  - Can cause big bang releases
- Standardise Error structure
  - Applications will know how to look at Errors
  - Clients see a readable message

Recommendations for Building APIs

- Pagination
  - Server pagination - Server return partial result with link to next
  - Client pagination - Client specifies result count to receive

Recommendations for Building APIs

- Filtering
  - Provide query language for client to refine results
  - Implement entire query language

Recommendations for Building APIs

- Filtering

Operator	Description	Example
Comparison Operators		
eq	Equal	city eq 'Redmond'
ne	Not equal	city ne 'London'
gt	Greater than	price gt 20
ge	Greater than or equal	price ge 10
lt	Less than	price lt 20
le	Less than or equal	price le 100
Logical Operators		
and	Logical and	price le 200 and price gt 3.5
or	Logical or	price le 3.5 or price gt 200
not	Logical negation	not price le 3.5
Grouping Operators		
( )	Precedence grouping	(priority eq 1 or city eq 'Redmond') and price gt 100

https://github.com/microsoft/api-guidelines/blob/vNext/Guidelines.md#971-filter-operations

Recommendations for Building APIs

- CORS (Cross Origin Resource Sharing)
  - Important to consider from server side. Where are requests coming from?
- Consider security from the outset
  - e.g. OAuth2, HTTPS
  - Public or Private Clients
  - Internal and external applications

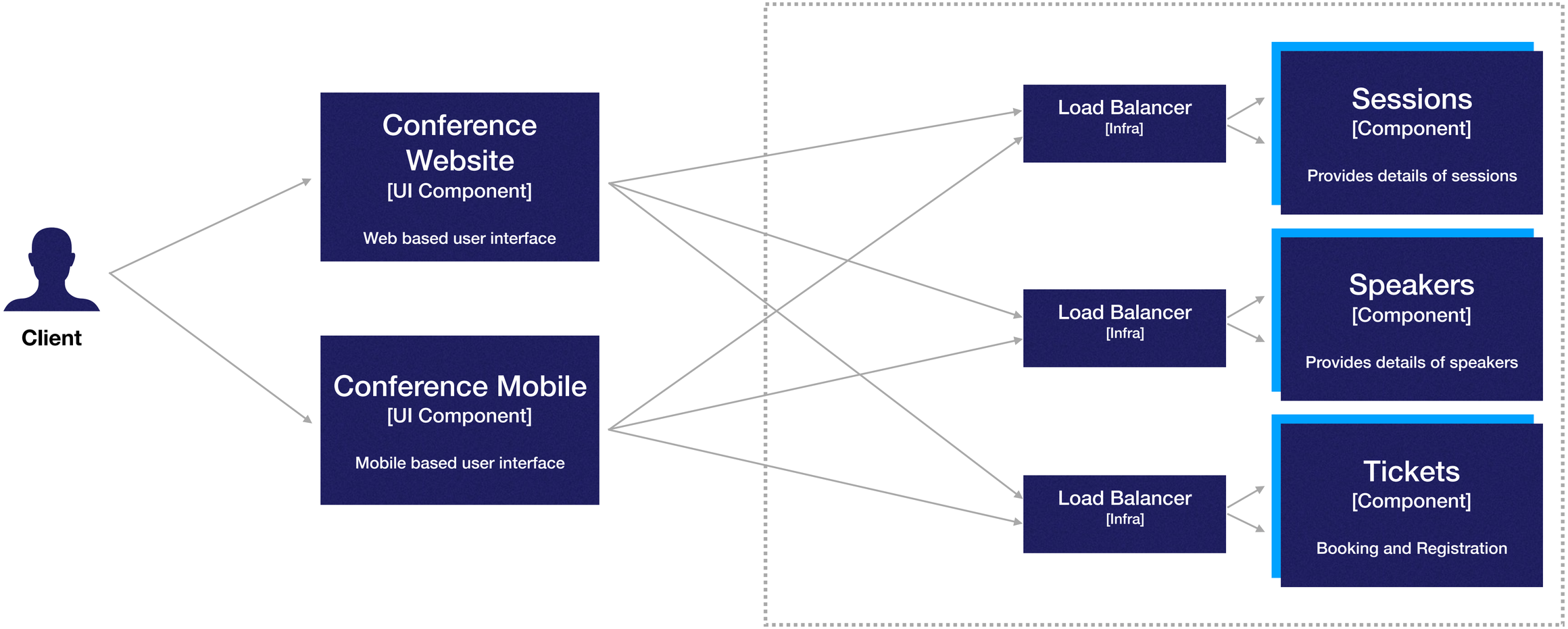
API Gateways

O'REILLY®
Software
Architecture

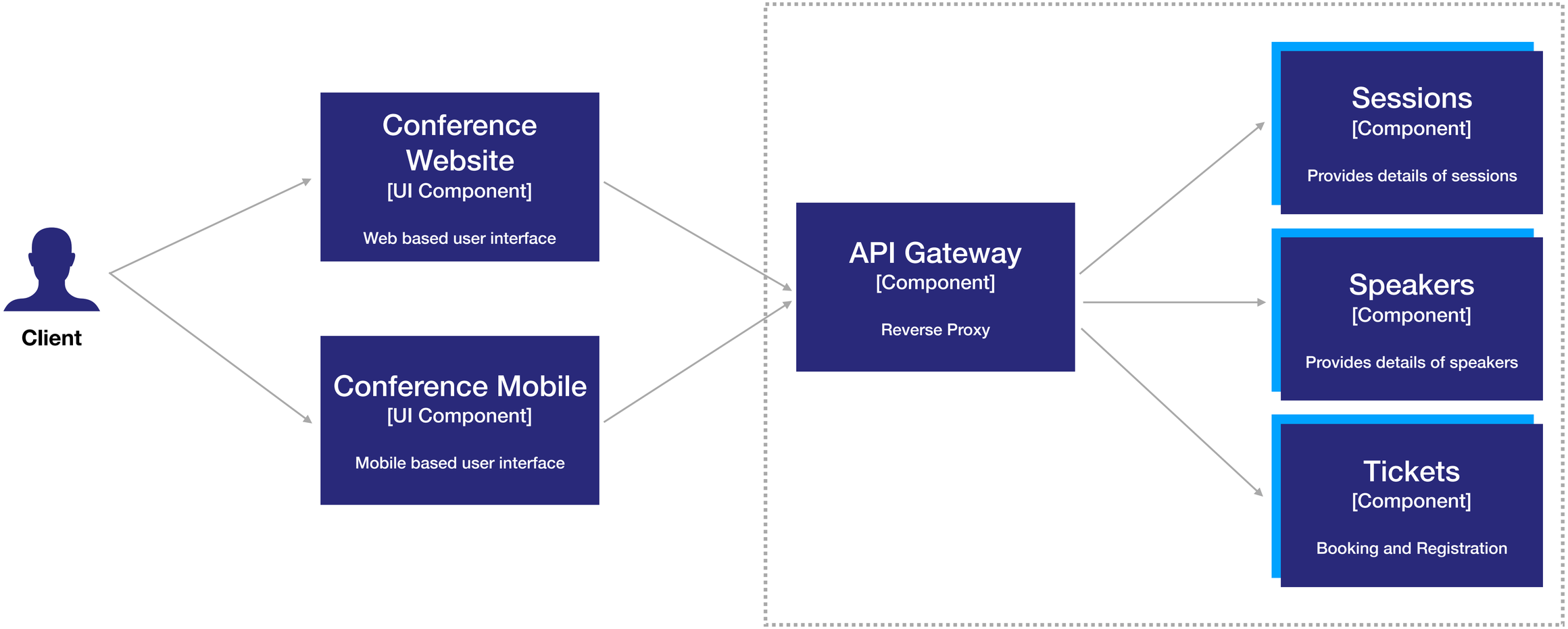
Handling Multiple APIs

- It is unlikely you will build a monolithic API system
- Routing traffic to services becomes an architectural point for consideration
- May wish to avoid each service implementing
  - Uniform Request Logging
  - Security Considerations (Entitlements/SSL Termination)
  - Circuit breaking/Load balancing

What is a Gateway?

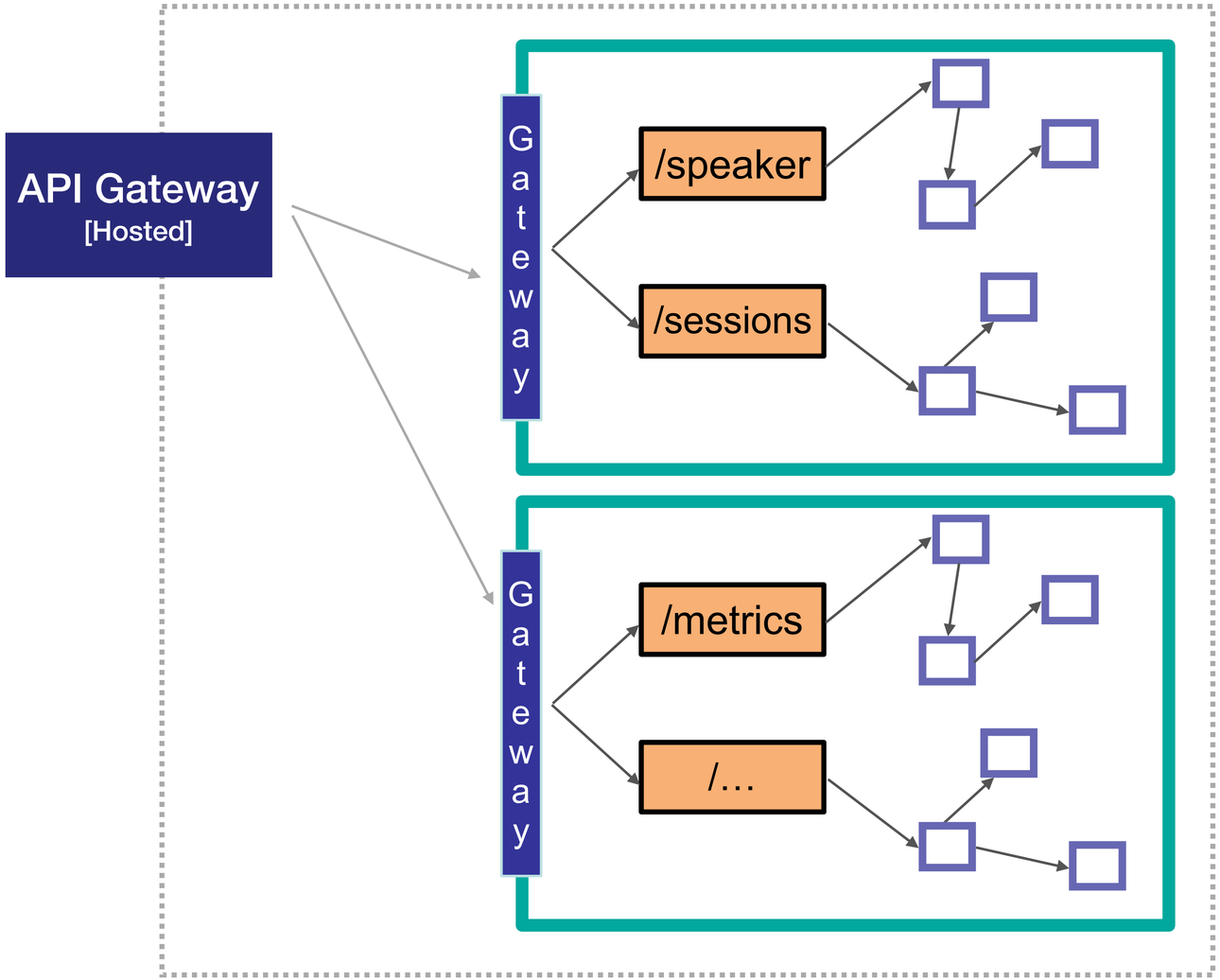


What is a Gateway?

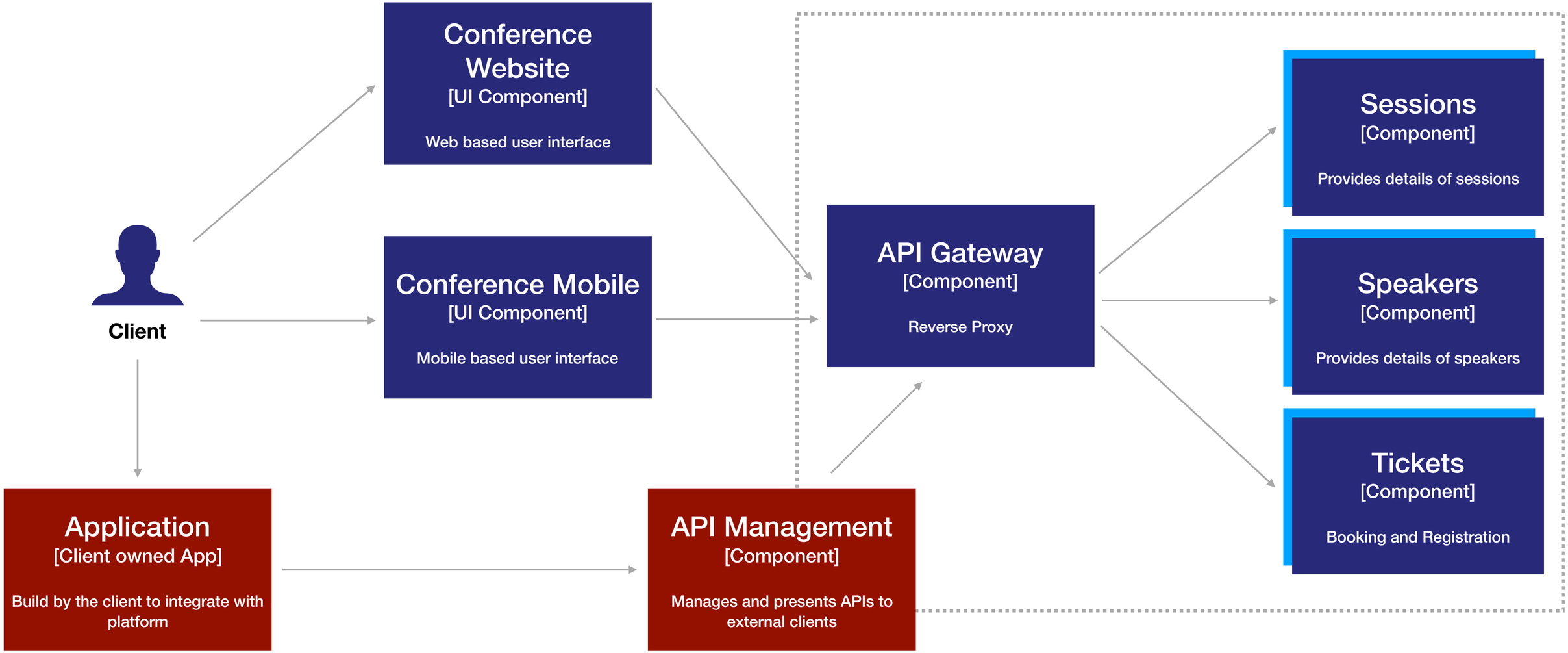


Building an API Centric Architecture

- Gateways allow for an API Centric strategy
- Build APIs as Microservices
- Delivery velocity at Microservices Gateway
- Repository of APIs at Enterprise Gateway
- Easy to extend and rapidly build
- API Governance and Curation Challenges



Role of API Management



Demo - Applying a Gateway

Creates a Network

Builds out an image

Pulls in a gateway

```yaml
version: '3'
services:
  todo-service:
    build:
      context: ./
      dockerfile: Dockerfile
    image: "task-service:latest"
    ports:
      - "8080:8080"
  api-gateway:
    image: jpgough/api-workshop-gateway
    ports:
      - "8081:8081"
```

Demo - Applying a Gateway

```
@SpringBootApplication
public class GatewayApplication {

    @Bean
    public RouteLocator customRouteLocator(RouteLocatorBuilder builder) {
        return builder.routes()
                .route("tasks", r -> r.path("/tasks/**")
                .filters(f -> f.rewritePath("/tasks/(?<segment>.*)", "/${segment}"))
                .uri("http://todo-service:8080"))
                .build();
    }

    public static void main(String[] args) {
        SpringApplication.run(GatewayApplication.class, args);
    }

}
```