Designing and Structuring a RESTful Service

Maximilian Schwarzmüller
AUTHOR

@maxedapps www.mschwarzmueller.com

Overview

Decomposing your service

Data formats and when to use them

HTTP methods and API endpoints

Route protection and URL styles

A Finished Product

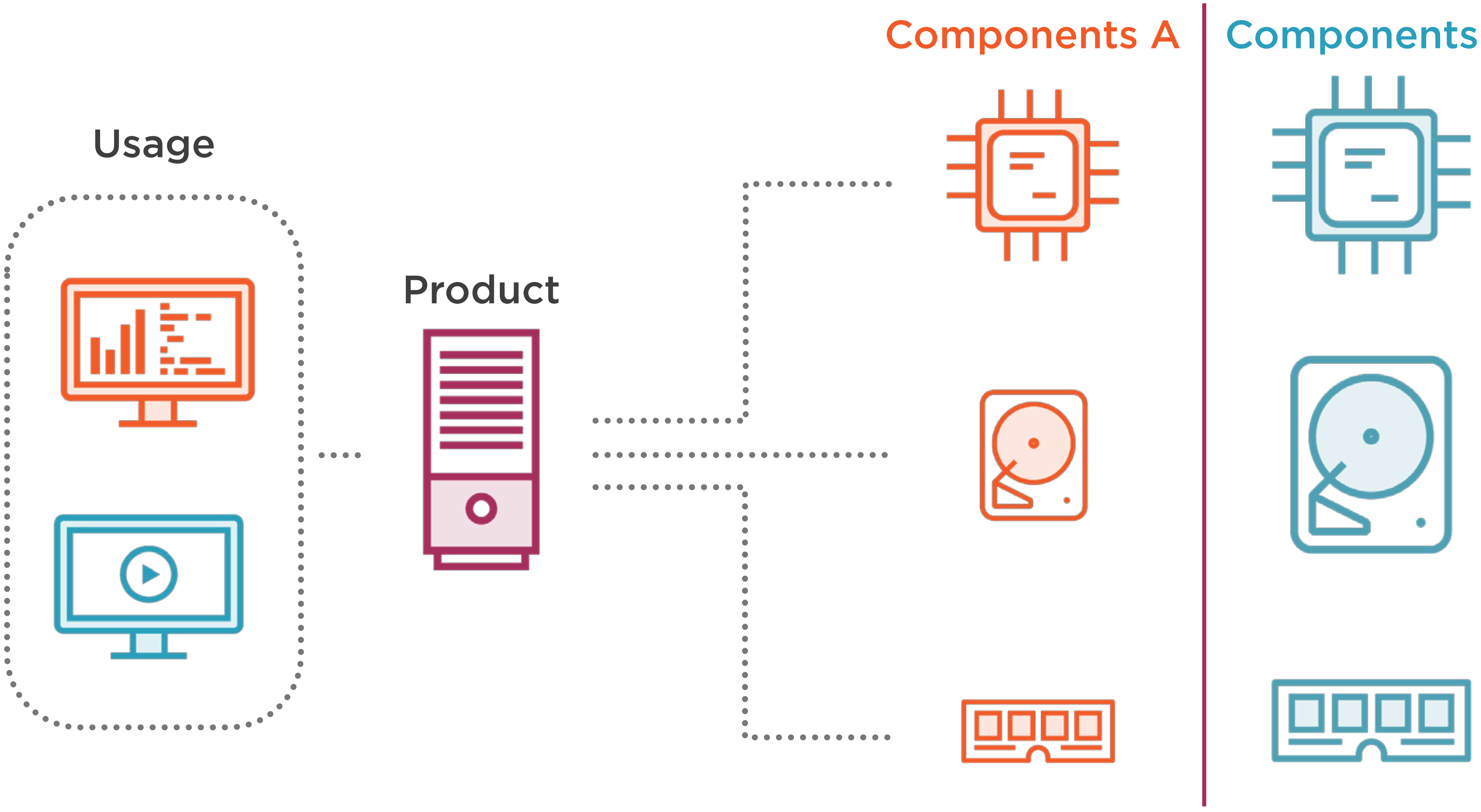
Computer

Allows users to handle tasks digitally

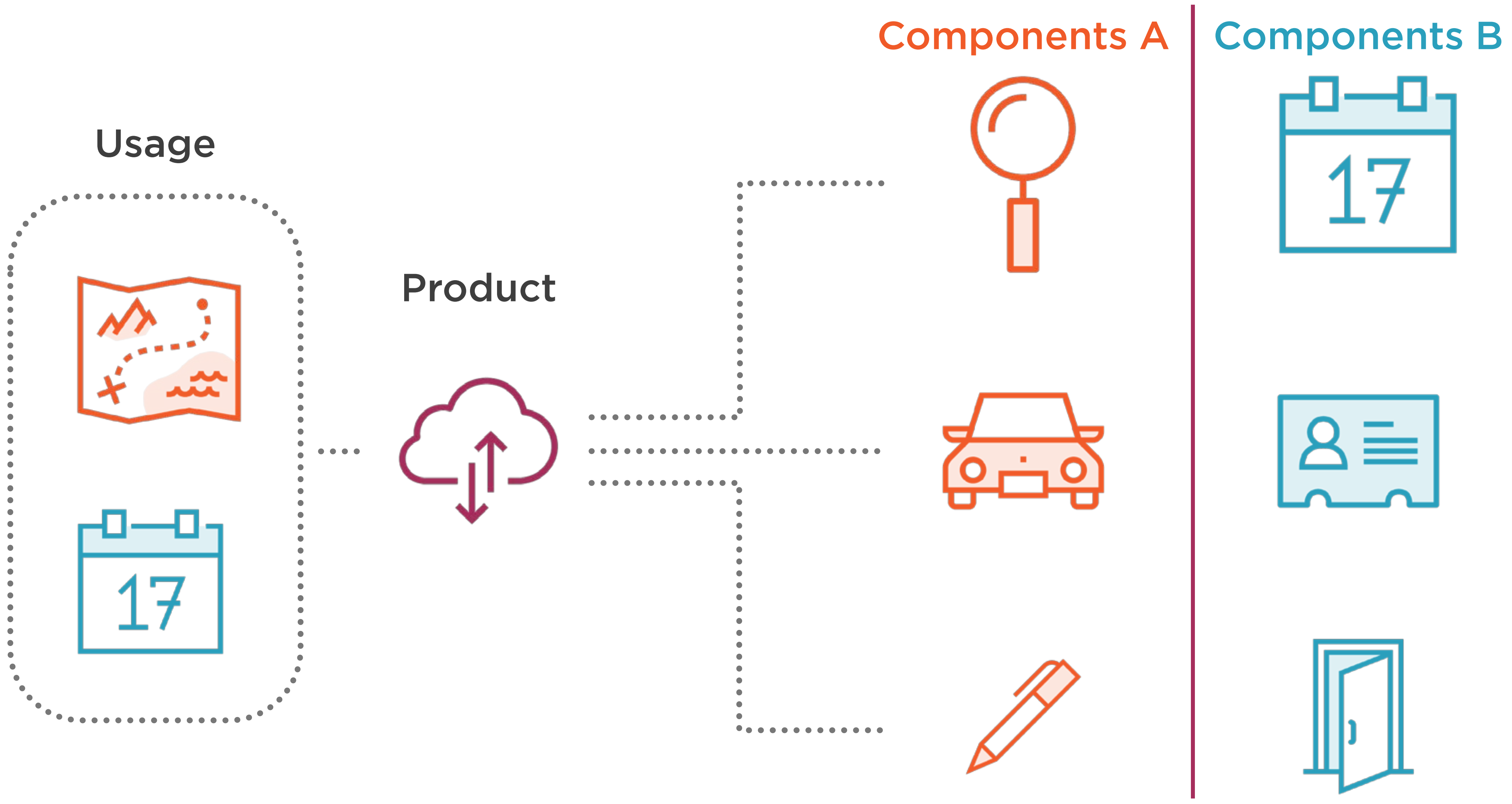
RESTful Service

Exposes various services via API

Decomposition of a Computer



Decomposition of a RESTful Service (API)

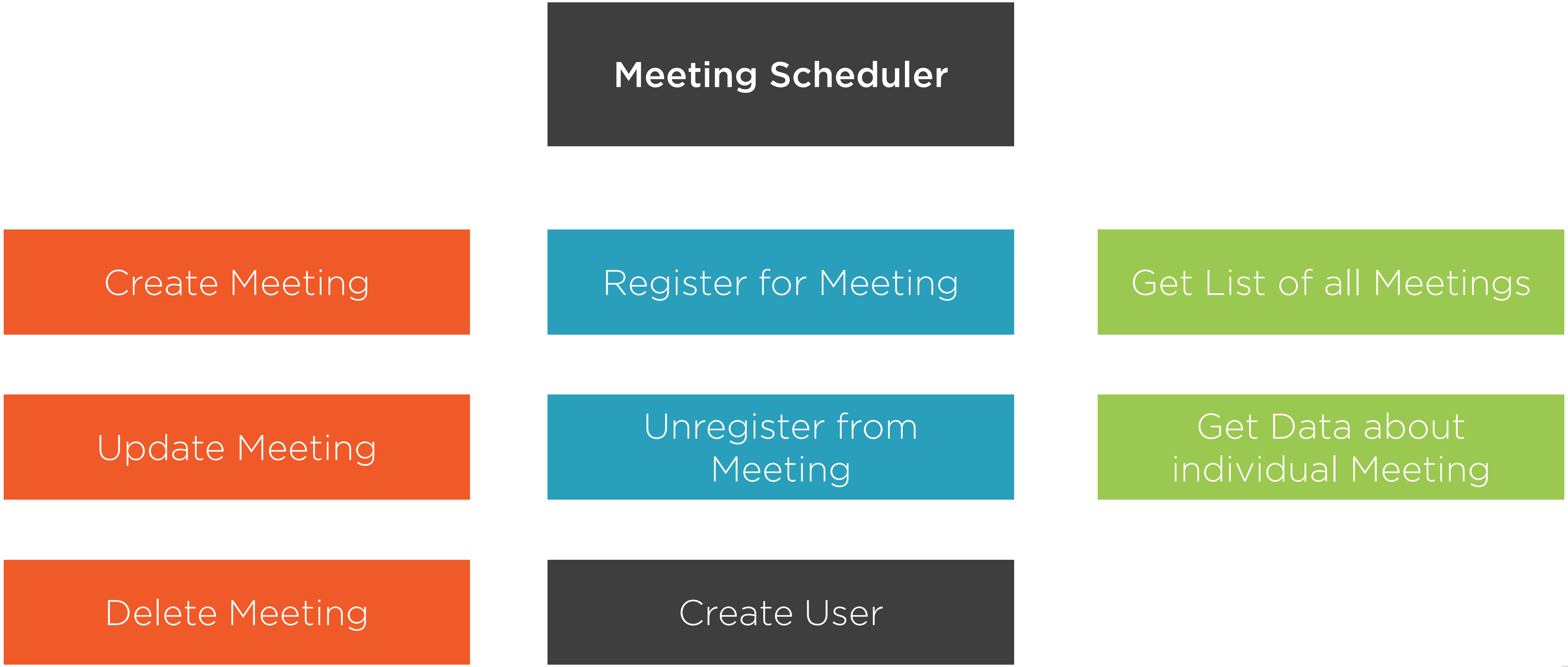


Demo: Course Project

Meeting Scheduler API

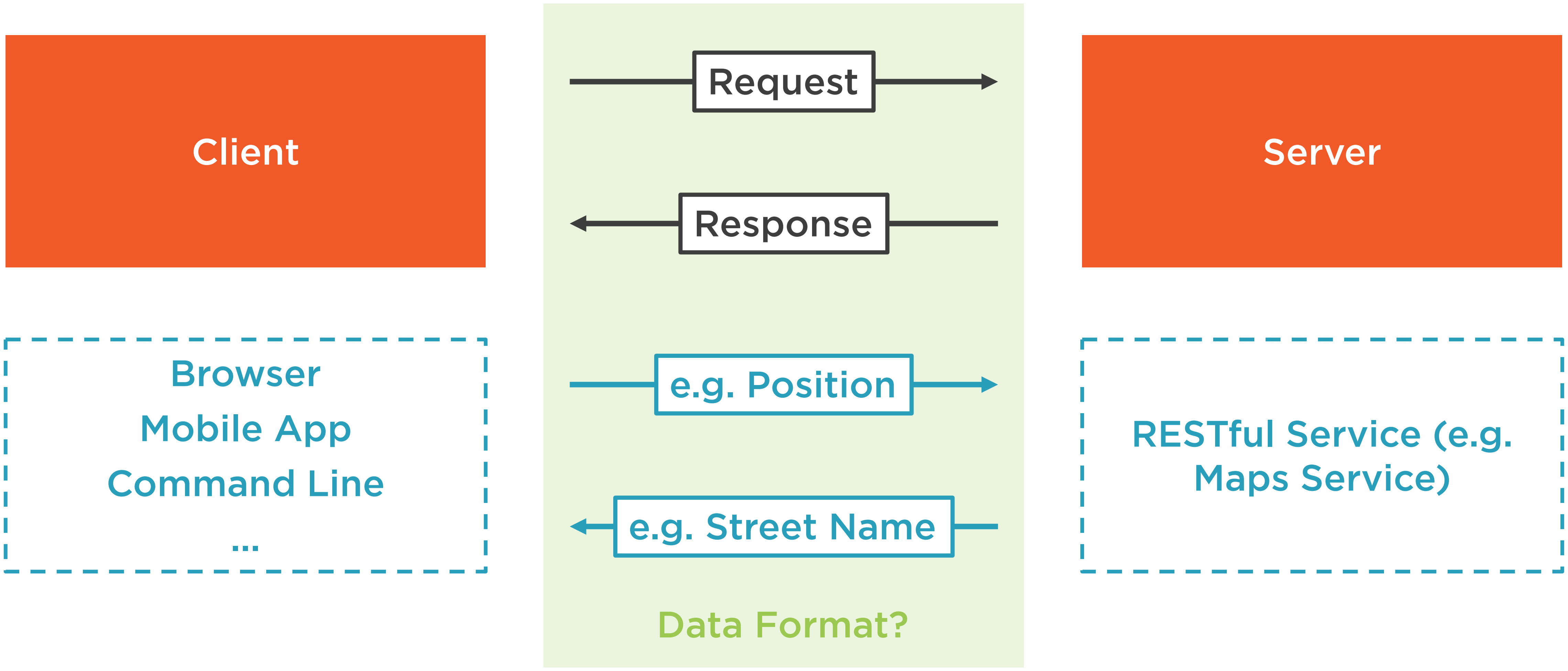
Users should be able to create, update, and delete meetings. Furthermore, other users should be able to register and unregister for any created meetings. Lastly, it should be possible to retrieve a list of all meetings or data about an individual meeting.

Decomposition of the Meeting Scheduler API



Requests, Responses, & Data Formats

Requests & Responses



Possible Data Formats

Allows complex structures

Small file size

XML

```
<coordinates>
        <x>5.8</x>
        <y>-1.3</y>
</coordinates>
```

JSON

```
{
        ”x”: 5.8,
        ”y”: -1.3
}
```

CSV

```
5.8,-1.3
```

Plain Text

```
X = 5.8
Y = -1.3
```

HTML

```
<div id=“x”>5.8</div>
<div id=“y”>-1.3</div>
```

File

```
8asdff63fas6ha7lasd
```

Demo: Project Data Formats

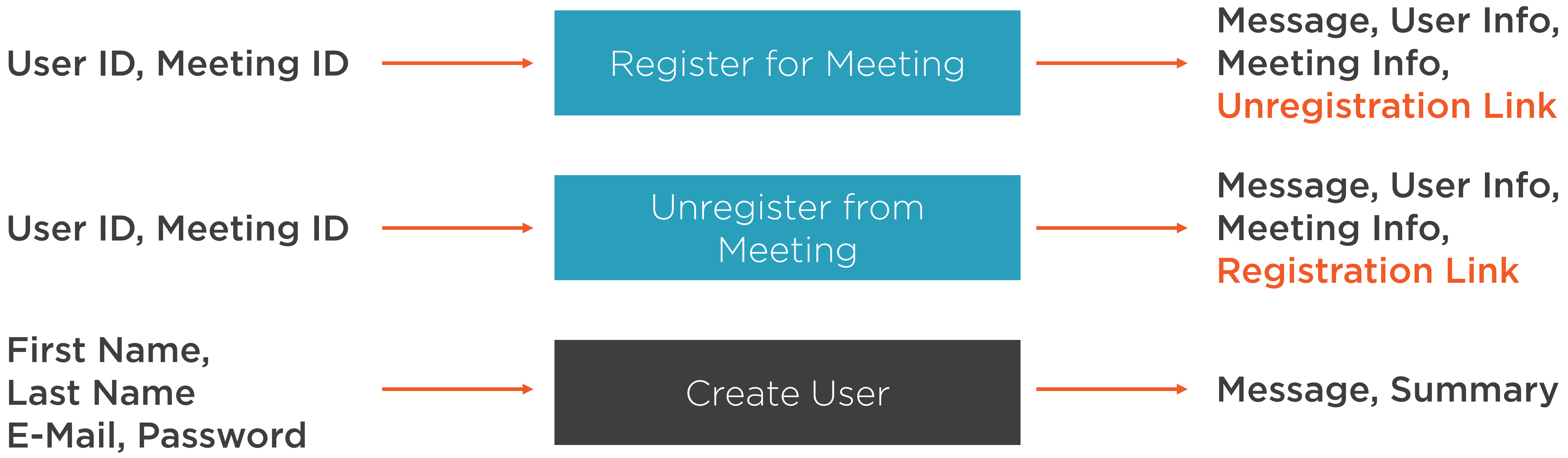
Meeting Scheduler API

- Create Meeting
- Update Meeting
- Delete Meeting
- Register for Meeting
- Unregister from Meeting
- Create User
- Get List of all Meetings
- Get Data about individual Meeting

Meeting Scheduler: Requests & Responses

Create Meeting	←	Title, Description, Time, User ID
	→	Message, Summary, Meeting URL, Participants
Update Meeting	←	Title, Description, Time, Meeting ID, User ID
	→	Message, Summary, Meeting URL, Participants
Delete Meeting	←	Meeting ID, User ID
	→	Message

Meeting Scheduler: Requests & Responses


User ID, Meeting ID
Register for Meeting
Message, User Info, Meeting Info, Unregistration Link
User ID, Meeting ID
Unregister from Meeting
Message, User Info, Meeting Info, Registration Link
First Name, Last Name E-Mail, Password
Create User
Message, Summary

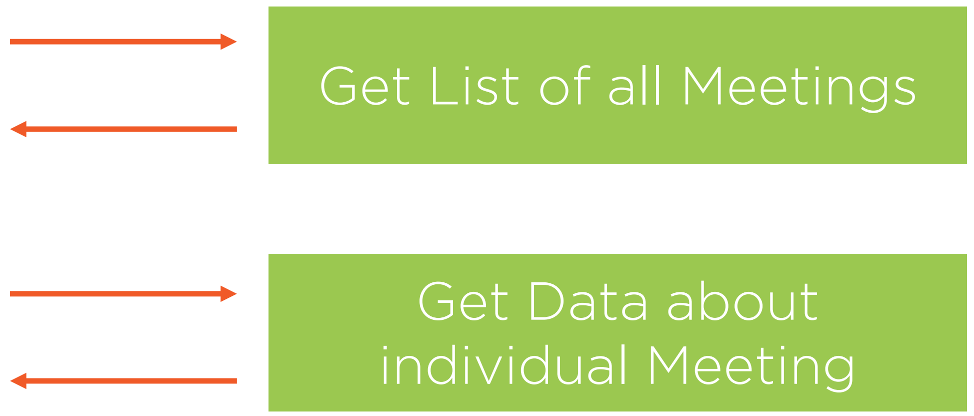
Meeting Scheduler: Requests & Responses

(null)

Meetings Info, Links to individual Meetings

Meeting ID

Meeting Info, Link to List of Meetings



HTTP Methods

Main HTTP Methods

GET

Retrieve a resource

POST

Add a resource

PUT

Replace a resource

PATCH

Update parts of a resource

DELETE

Remove a resource

HTTP Method Assignment

Database Method	HTTP Method
GET data from database	GET request
INSERT data into database	POST request
UPDATE data in database by overwriting old record	PUT request
UPDATE data in database by overwriting fields of old record	PATCH request
DELETE data in database	DELETE request

Demo: Assigning HTTP Methods

Meeting Scheduler API

POST
Create Meeting

PATCH **Replace Fields**
Update Meeting

DELETE
Delete Meeting

POST
Register for Meeting

DELETE
Unregister from Meeting

POST
Create User

GET
Get List of all Meetings

GET
Get Data about individual Meeting

Route Protection & URL Styles

Sensible API Endpoints

POST
Create Meeting

PUT
Update Meeting

DELETE
Delete Meeting

POST
Register for Meeting

DELETE
Unregister from Meeting

POST
Create User

GET
Get List of all Meetings

GET
Get Data about individual Meeting

POST
User Signin

Why URL Styling?

Expressive
Clear idea of accessed resource or action

Updatable
Easy usage of multiple API versions

Intuitive Navigation
Allows inference of other endpoints

Comparison of URL Styles

GET, POST, ...	📋	🔄	🗺
/api/meeting/get/5	✓	✗	✗
/api/meeting/5	✓	✗	✓
/api/v1/meeting/5	✓	✓	✓

Demo: Final Structure of Meeting Scheduler

POST

Create Meeting

↑ Title, Description, Time, User ID

↓ Message, Summary, URL, Users

/api/v1/meeting

PATCH

Update Meeting

↑ Meeting Data, User ID, Meeting ID

↓ Message, Summary, URL, Users

/api/v1/meeting

DELETE

Delete Meeting

↑ Meeting ID, User ID

↓ Message,

/api/v1/meeting

POST

Register

↑ User ID, Meeting ID

↓ Message, User, Meeting, URL

/api/v1/meeting/registration

DELETE

Unregister

↑ User ID, Meeting ID

↓ Message, User, Meeting, URL

/api/v1/meeting/registration

POST

Create User

↑ Name, E-Mail, Password

↓ Message, User, Meeting, URL

/api/v1/meeting/user

GET

Get List of all Meetings

↑ (null)

↓ List of Meetings, URL

/api/v1/meeting

GET

Get Meeting

↑ Meeting ID

↓ Meeting Info, URL

/api/v1/meeting

POST

User Signin

↑ E-Mail, Password

↓ ???

/api/v1/user/signin

🟥 Authentication required

🟦 No authentication required

Summary

Decomposition makes identification of API endpoints easy

Request and response data structure depends on the API endpoint

Data is best transmitted as JSON using appropriate HTTP methods

Route protection allows the implementation of user-dependent features

URL styling is directly correlated to the usability of the service