#ImagineThis

You're now the tech guru on your campus!

Microsoft Student Partners

US Microsoft Student Partner (MSP) Program Job Description | January – June 2016

www.microsoftstudentpartners.com

What are we looking for?

If you're passionate about technology, love throwing fun events on campus, and aren't shy about sharing your enthusiasm for the latest technologies, you could be the perfect fit for the Microsoft Student Partner program!

What do you get from becoming an MSP?

As a student partner for Microsoft, you will learn new technical skills and improve them throughout the year, cultivate valuable social and professional relationships, and develop leadership and communication skills to boost your resume and increase your employability. You'll have the opportunity to learn from, and work alongside, industry professionals, you'll have access to a diverse Microsoft community of student developers, tools, and resources *and* you'll receive free software, hardware, and cool swag!

As a student partner for Microsoft, you will:

- Build your resume and increase your professional technology experience
- Represent Microsoft on your campus
- Host fun, informative workshops where you will teach other students new skills
- Work alongside Microsoft professionals, assisting them during hackathons and events
- Grow and nurture a community of students on-campus and online
- Promote Microsoft programs and competitions

Email usmsp@microsoft.com for any questions.

RICE®

Web Development

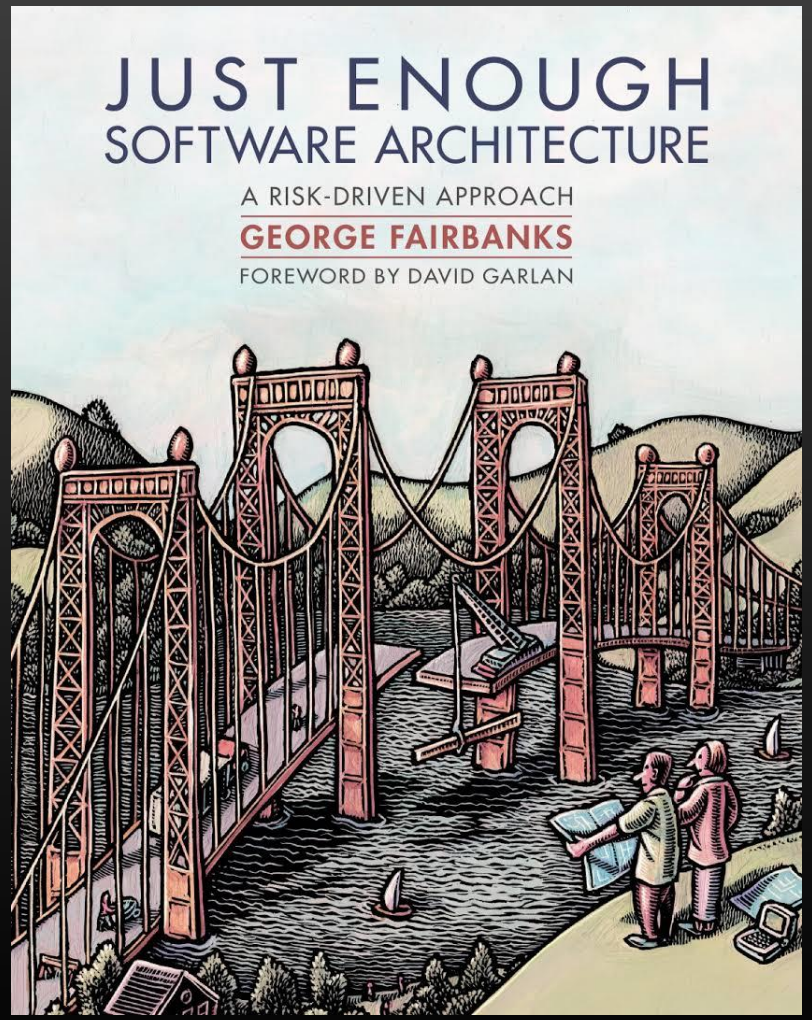
COMP 431 / COMP 531

Architecture and REST

Scott E Pollack, PhD

March 15, 2016

Part II – Back End Development



Part II

~~Web Servers~~

~~Backend~~

Architecture

Unit Testing

Web Hosting

Databases

Homework Assignment 6
(Draft Back-End)
Due Thursday 3/24

What was it called?

- HTML
- CSS
- JavaScript

each in their own file

- Controllers
- Views
- Models

each in their own file

- Each view has one Controller
- Every function performs one operation
- Each function is as simple as it can be

What was it called?

- HTML
- CSS
- JavaScript

each in their own file

- Controllers
- Views
- Models

each in their own file

- Each view has one Controller
- Every function performs one operation
- Each function is as simple as it can be

Separation of Concerns

Service Oriented Architecture (SOA)

- Separation of concerns within a business application
- Instead of a monolithic application, a collection of services
- Services are encapsulated behind interfaces
- Services
  - are logical representations of a single business activity
  - are self-contained
  - May be composed of other services
  - a “black box” to consumers
  - unassociated or loosely coupled to other services

SOA Framework

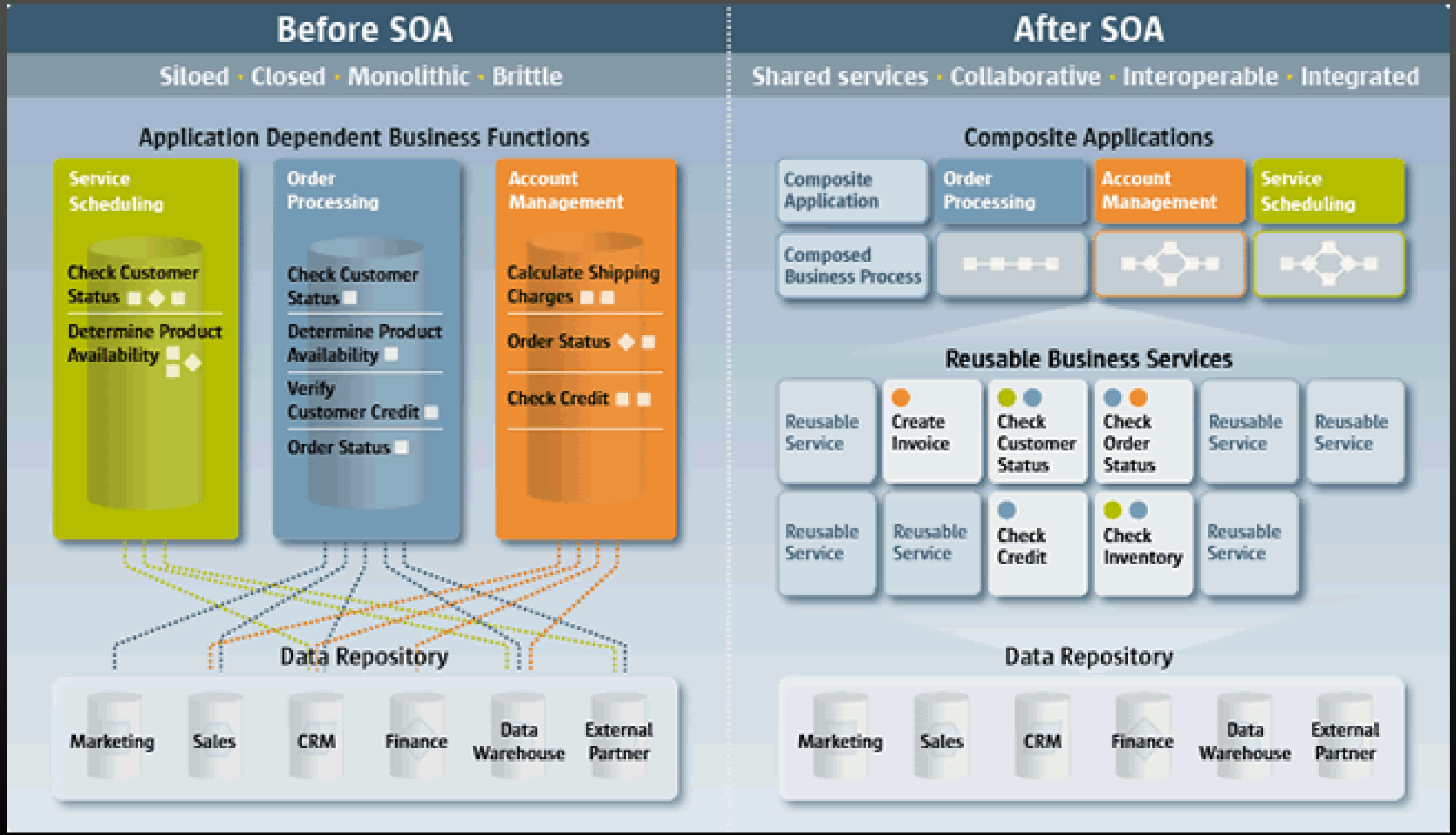
Horizontal Layers

- Consumer Interface Layer
  - GUI
  - Front end of web app
- Business Process Layer
  - Orchestration
- Services
  - Functionality
- Operational Systems
  - Storage and Persistence
    - e.g., database

Vertical Layers

- Integration Layer
  - Communication protocols
  - Enterprise service bus (ESB)
  - Business-to-Business (B2B)
  - Business-to-Consumer (B2C)
- Quality of Service
  - security, availability, performance
  - Service and operational level agreements (SLA and OLA)
- Informational
- Governance

Service consumer
Service provider
Consumers
Channel
B2B
Business process
Composition; choreography;
business state machines
Services
atomic and composite
Service components
Operational systems
Packaged
application
Custom
application
OO
application
Integration (enterprise service bus)
QoS layer (security, management &
monitoring infrastructure services)
Data architecture (meta-data) &
business intelligence
Governance



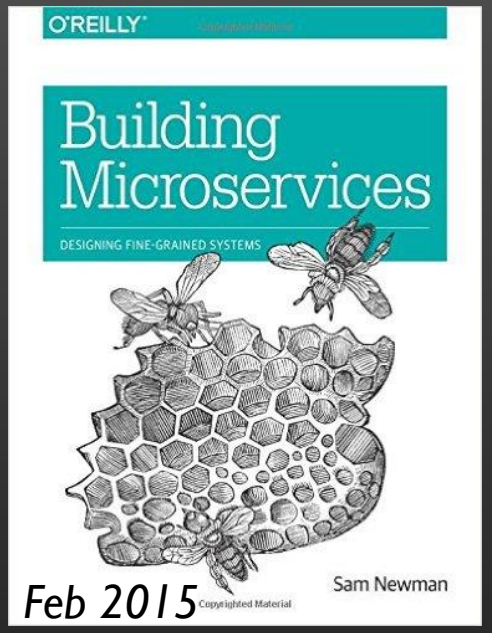
http://www.logimethods.com/enterprise-soa-soe.php

Microservices

- *Small independent processes communicating with each other using language-agnostic APIs*

- Small
- Highly decoupled
- Focus on doing a small task

- Modularity
- Componentization



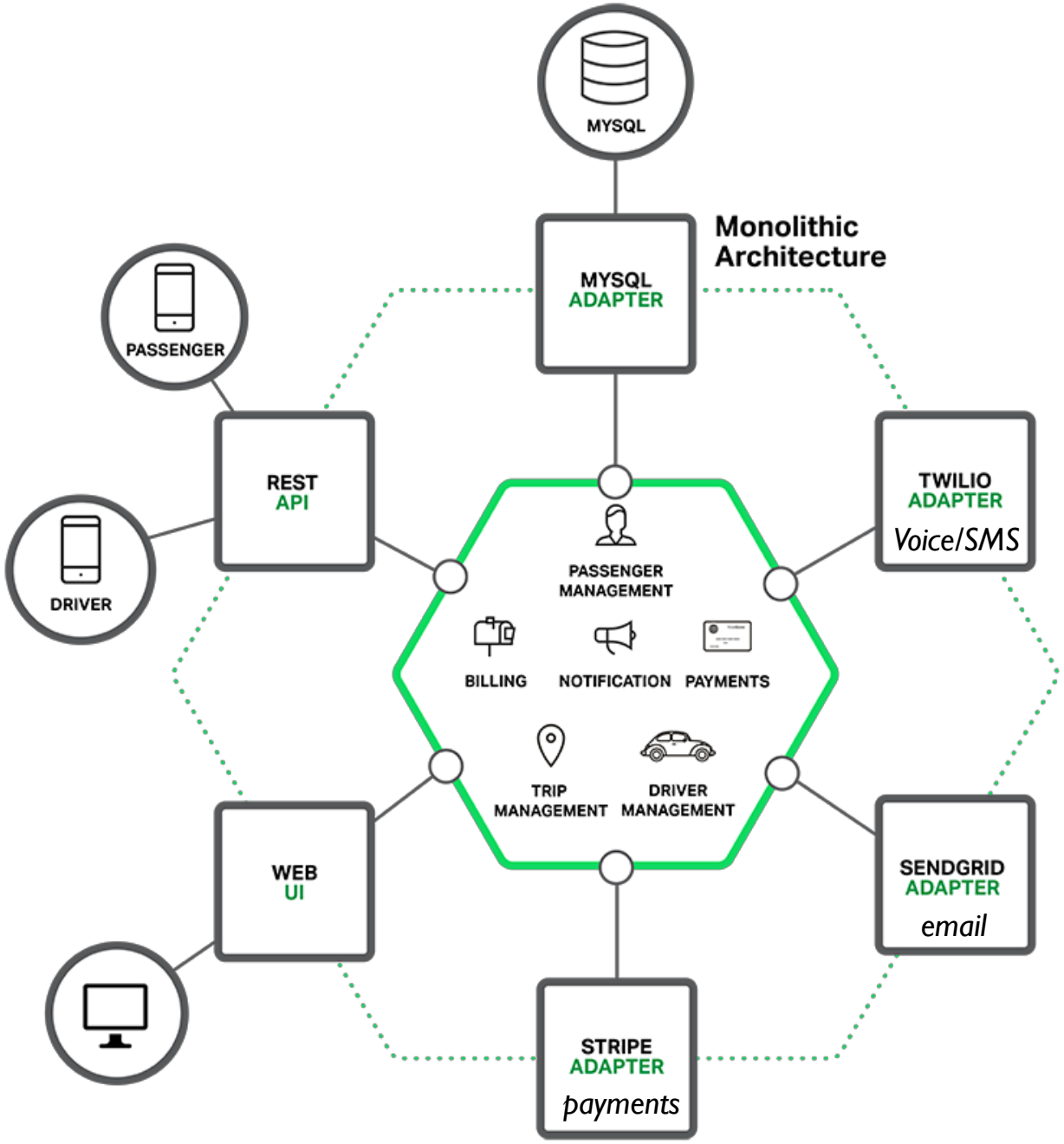
- Services are easy to replace

- Implementation agnostic
  - In relation to language
  - Database selection
  - Hardware

- Architecture is symmetrical
  - As opposed to producer-consumer

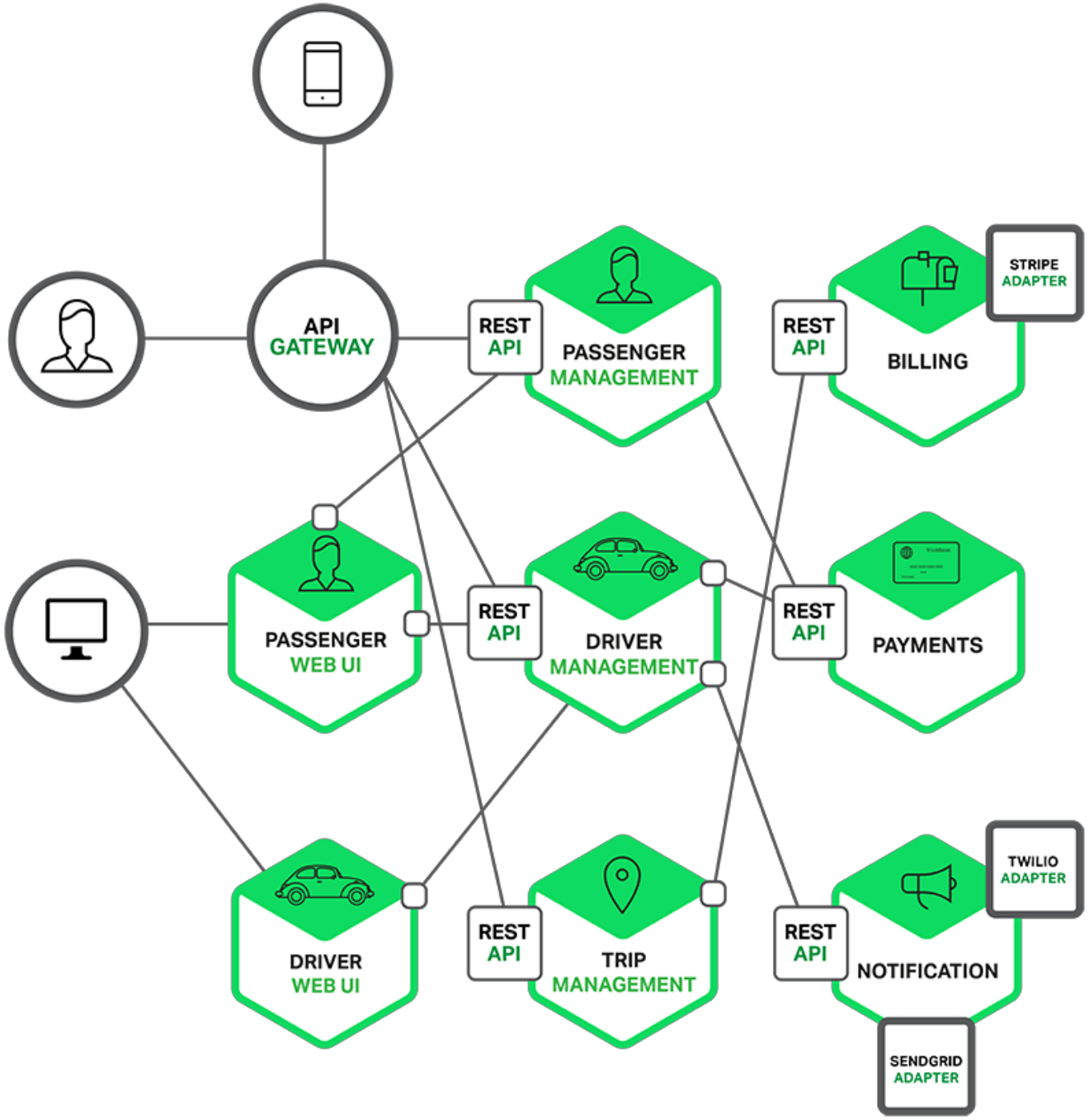
- Promotes continuous delivery

Monolithic



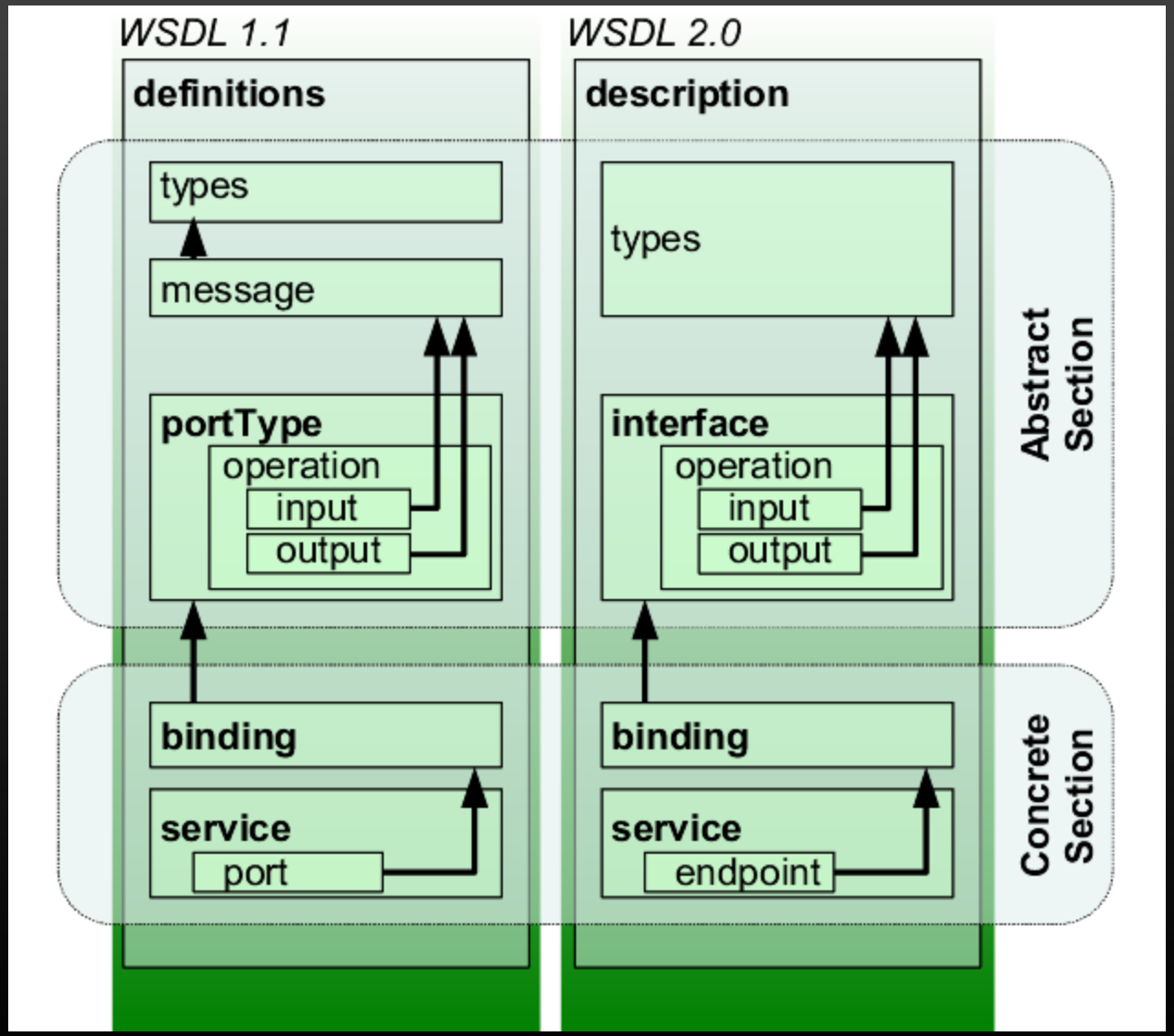
https://www.nginx.com/blog/introduction-to-microservices/

Microservices



https://www.nginx.com/blog/introduction-to-microservices/

Web Services Description Language (WSDL)




```
<?xml version="1.0" encoding=
<definitions name="AktienKurs
  targetNamespace="http://loc
  xmlns:xsd="http://schemas.xmlsoap.or
  xmlns="http://schemas.xmlsoap.org/wsd
  <service name="AktienKurs">
    <port name="AktienSoapPort" binding
      <soap:address location="http://loc
    </port>
    <message name="Aktie.HoleWert">
      <part name="body" element="xsd:Tra
    </message>
    …
  </service>
</definitions>
```

WSDL

Remote Procedure Call (RPC)

1. Client calls a *stub* with parameters
2. Stub *marshalls* parameters and performs system call to send message
3. Client OS sends message to server
4. …transport…
5. Server OS receives message and delivers to server *stub*
6. Stub *unmarshalls* parameters
7. Stub calls server procedure
8. Server replies with same steps in reverse direction

RPC is a Paradigm

Internet Protocol (IP)

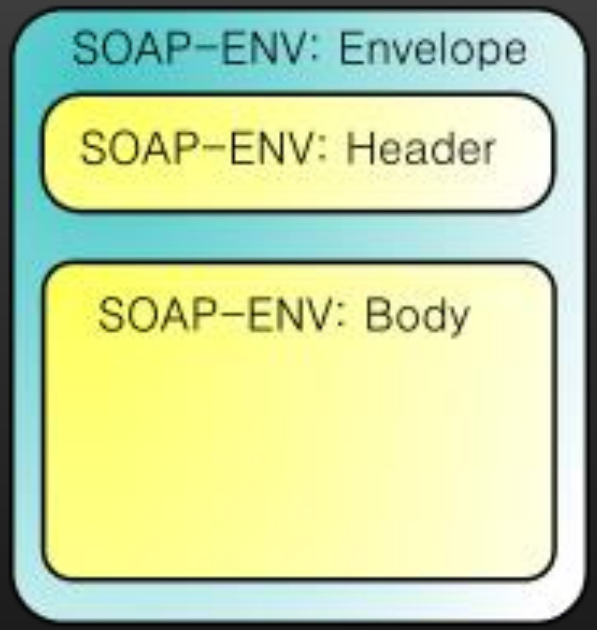
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)
- Internet Control Message Protocol (ICMP)
- Hypertext Transfer Protocol (HTTP)
- Post Office Protocol (POP)
- File Transfer Protocol (FTP)
- Internet Message Access Procol (IMAP)
- General Inter-ORB Protocol (GIOP)
- Java Remote Method Invocation (RMI)
- Distributed Component Object Model (DCOM)
- Dynamic Data Exchange (DDE)
- Simple object Access protocol (SOAP)

Simple Object Access Protocol (SOAP)

SOAP is a messaging protocol, typically XML-formatted

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 299
SOAPAction: "http://www.w3.org/2003/05/soap-envelope"

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <m:GetStockPrice xmlns:m="http://www.example.org/stock/Surya">
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```



Can be used with HTTP, SMTP, JMS, message queues

Representational State Transfer (ReST)

- First proposed by Roy Thomas Fielding in his 2000 PhD dissertation
- Client-server separation of concerns to simplify component implementation, reduction of complexity, increase scalability
- Standard HTTP method verbs
- Typically JSON messages
- Hypertext link to state
- Hypertext link to reference-related resources
- REST is stateless and cacheable
- Unlike SOAP, no official standard, REST is a style not a protocol

RESTful CRUD

- Data access and manipulation through CRUD operations

POST can be done more than once.

Operation	SQL	HTTP	DDS
Create (Add)	INSERT	PUT / POST	write
Read (Retrieve)	SELECT	GET	read / take
Update (Modify)	UPDATE	PUT / PATCH	write
Delete (Destroy)	DELETE	DELETE	dispose

Do creation as "PUT" could avoid such as: double submission of forms and thus dupicate entries in server

nullipotent

idempotent

idempotent

single thing happened

Representational State Transfer Example

RESTful API HTTP methods

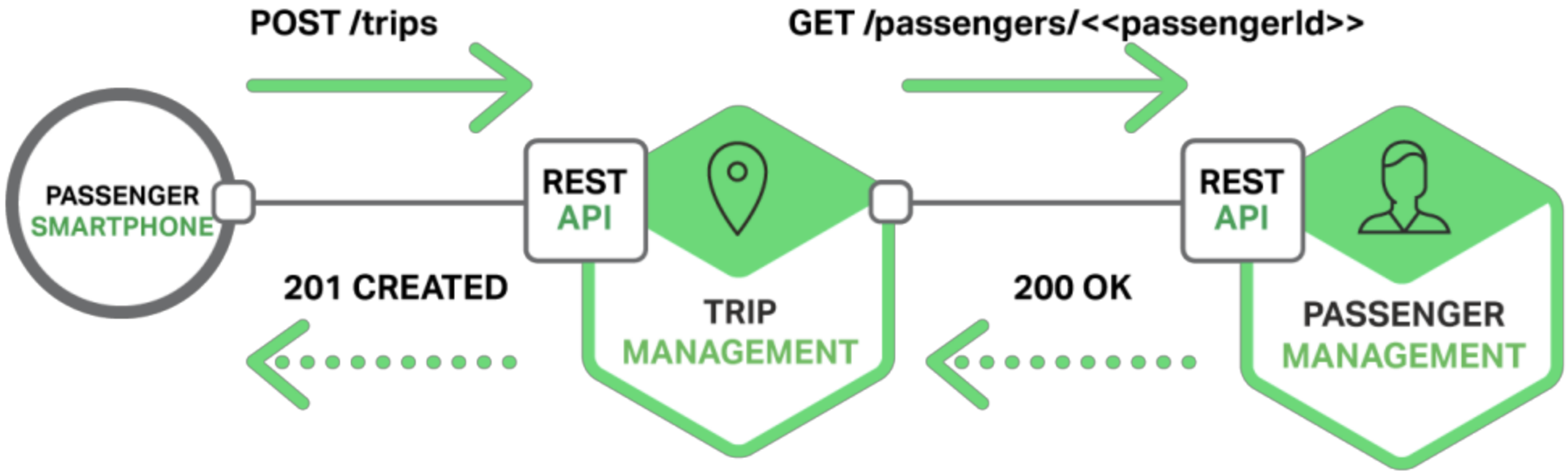
Resource	GET	PUT	POST	DELETE
Collection URI, such as `http://api.example.com/v1/resources/`	**List** the URIs and perhaps other details of the collection's members.	**Replace** the entire collection with another collection.	**Create** a new entry in the collection. The new entry's URI is assigned automatically and is usually returned by the operation.[10]	**Delete** the entire collection.
Element URI, such as `http://api.example.com/v1/resources/item17`	**Retrieve** a representation of the addressed member of the collection, expressed in an appropriate Internet media type.	**Replace** the addressed member of the collection, or if it does not exist, **create** it.	Not generally used. Treat the addressed member as a collection in its own right and **create** a new entry in it.[10]	**Delete** the addressed member of the collection.

"REST provides a set of architectural constraints that, when applied as a whole, emphasizes scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems."

—Roy Thomas Fielding

Architectural Styles and the Design of Network-based Software Architectures

(PhD Dissertation UC Irvine 2000)



Semantic URLs

Non-semantic URL	Semantic URL
http://example.com/index.php?page=name	http://example.com/name
http://example.com/index.php?page=consulting/marketing	http://example.com/consulting/marketing
http://example.com/products?category=2&pid=25	http://example.com/products/2/25
http://example.com/cgi-bin/feed.cgi?feed=news&frm=rss	http://example.com/news.rss
http://example.com/services/index.jsp?category=legal&id=patents	http://example.com/services/legal/patents
http://example.com/kb/index.php?cat=8&id=41	http://example.com/kb/8/41
http://example.com/index.php?mod=profiles&id=193	http://example.com/profiles/193

Limitations of REST

- Discoverability
- Extensibility
- Reverse Engineering (tight coupling)
- Modularity
- Versioning

```
/????

/customers/1234/somethingnew

/customers/1234
/customers/
/customers/1234/contacts/4455
/customers/1234/contacts/abc/4455
/v1/customers/1234
/v2/customers/1234
```

Don put version on theRUL

Hypermedia As The Engine Of Application State
HATEOAS

- Requests to service
- Service responds with state with possible actions
- Client acts (hypermedia)
- Service responds with state and possible actions
- Client acts (hypermedia)
- …

```
GET /account/12345
{
  accountNo: 12345,
  balance: 100.00,
  deposit: ‘/account/12345/deposit’,
  withdraw: ‘/account/12345/withdraw’,
  transfer: ‘/account/12345/transfer’
}
GET /account/54321
{
  accountNo: 54321,
  balance: -100.00,
  deposit: ‘/account/12345/deposit’,
}
```

HATEOAS in Action

```
#> curl https://api.github.com
{
  "current_user_url": "https://api.github.com/user",
  "current_user_authorizations_html_url": "https://github.com/settings/connections/applications{/client_id}",
  "authorizations_url": "https://api.github.com/authorizations",
  "code_search_url": "https://api.github.com/search/code?q={query}{&page,per_page,sort,order}",
  "emails_url": "https://api.github.com/user/emails",
  "emojis_url": "https://api.github.com/emojis",
  "events_url": "https://api.github.com/events",
  "feeds_url": "https://api.github.com/feeds",
  "followers_url": "https://api.github.com/user/followers",
  "following_url": "https://api.github.com/user/following{/target}",
  "gists_url": "https://api.github.com/gists{/gist_id}",
  "hub_url": "https://api.github.com/hub",
  "issue_search_url": "https://api.github.com/search/issues?q={query}{&page,per_page,sort,order}",
  "issues_url": "https://api.github.com/issues",
  "keys_url": "https://api.github.com/user/keys",
  "notifications_url": "https://api.github.com/notifications",
  "organization_repositories_url": "https://api.github.com/orgs/{org}/repos{?type,page,per_page,sort}",
  "organization_url": "https://api.github.com/orgs/{org}",
  "public_gists_url": "https://api.github.com/gists/public",
  "rate_limit_url": "https://api.github.com/rate_limit",
  "repository_url": "https://api.github.com/repos/{owner}/{repo}",
  "repository_search_url": "https://api.github.com/search/repositories?q={query}{&page,per_page,sort,order}",
```

HATEOAS in Action

```
#> curl https://api.github.com/repos/twosigma/ngrid/issues/5
{
  "url": "https://api.github.com/repos/twosigma/ngrid/issues/5",
  "labels_url": "https://api.github.com/repos/twosigma/ngrid/issues/5/labels{/name}",
  "comments_url": "https://api.github.com/repos/twosigma/ngrid/issues/5/comments",
  "events_url": "https://api.github.com/repos/twosigma/ngrid/issues/5/events",
  "html_url": "https://github.com/twosigma/ngrid/issues/5",
  "id": 68791265,
  "number": 5,
  "title": "Default codec should be UTF8, not ascii",
  "user": {
```

In-Class Exercise: Stubbing the Back-End

Using the API definition as your guide, and this example,
create profile.js and write stubs for

```
GET /status
PUT /status
GET /statues/:users
GET /email/:user
PUT /email
GET /zipcode/:user
PUT /zipcode
GET /pictures/:user
PUT /pictures
```

Validate your stubs by running your server locally and using curl or a browser plugin to make requests.

***Turnin* index.js *and* profile.js**
COMP431-S16:inclass-17