UNAERP
Universidade de Ribeirão Preto
Campus Ribeirão Preto - Campus Guarujá

Programação para Web II

API / REST

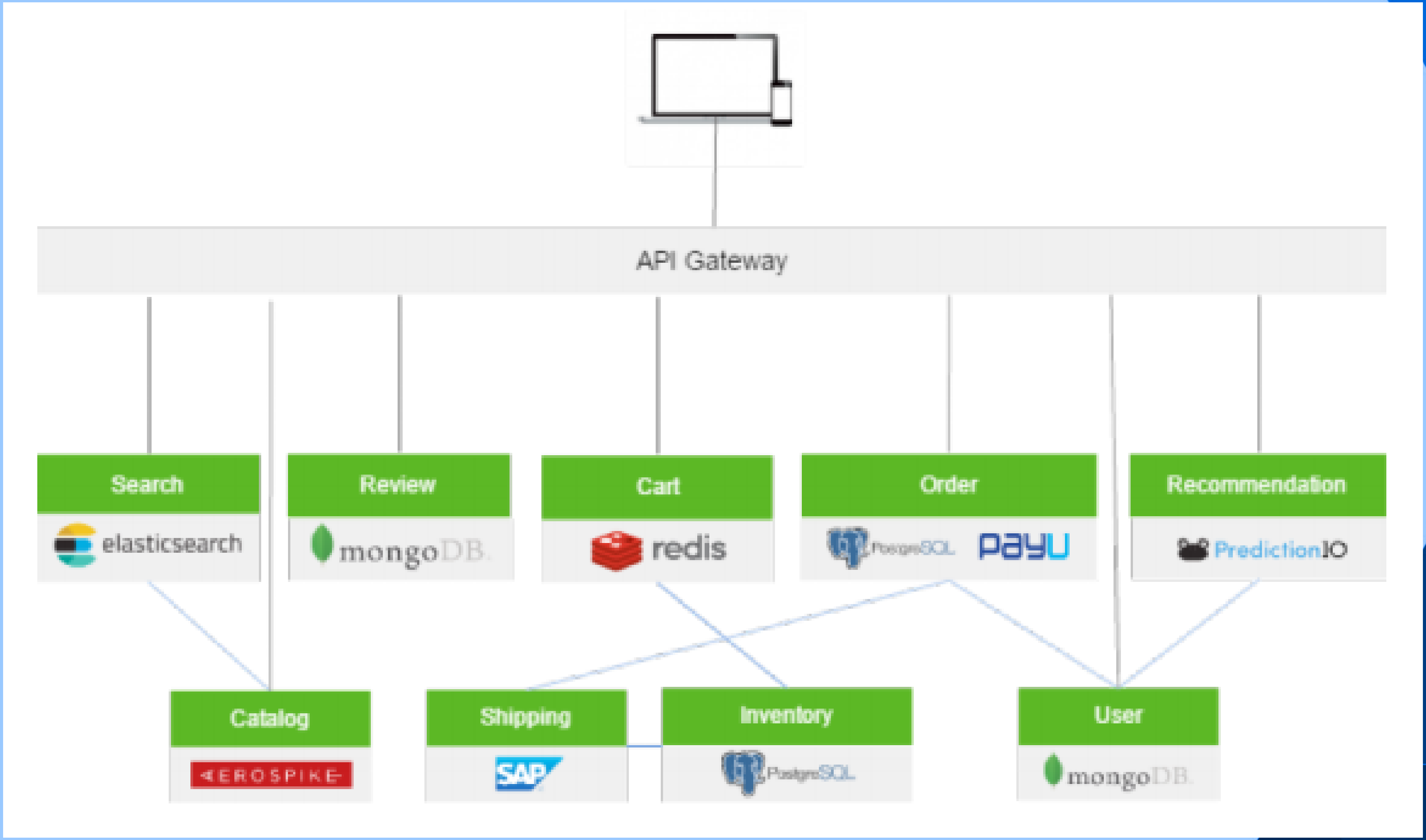
Prof. Eliézer Zarpelão

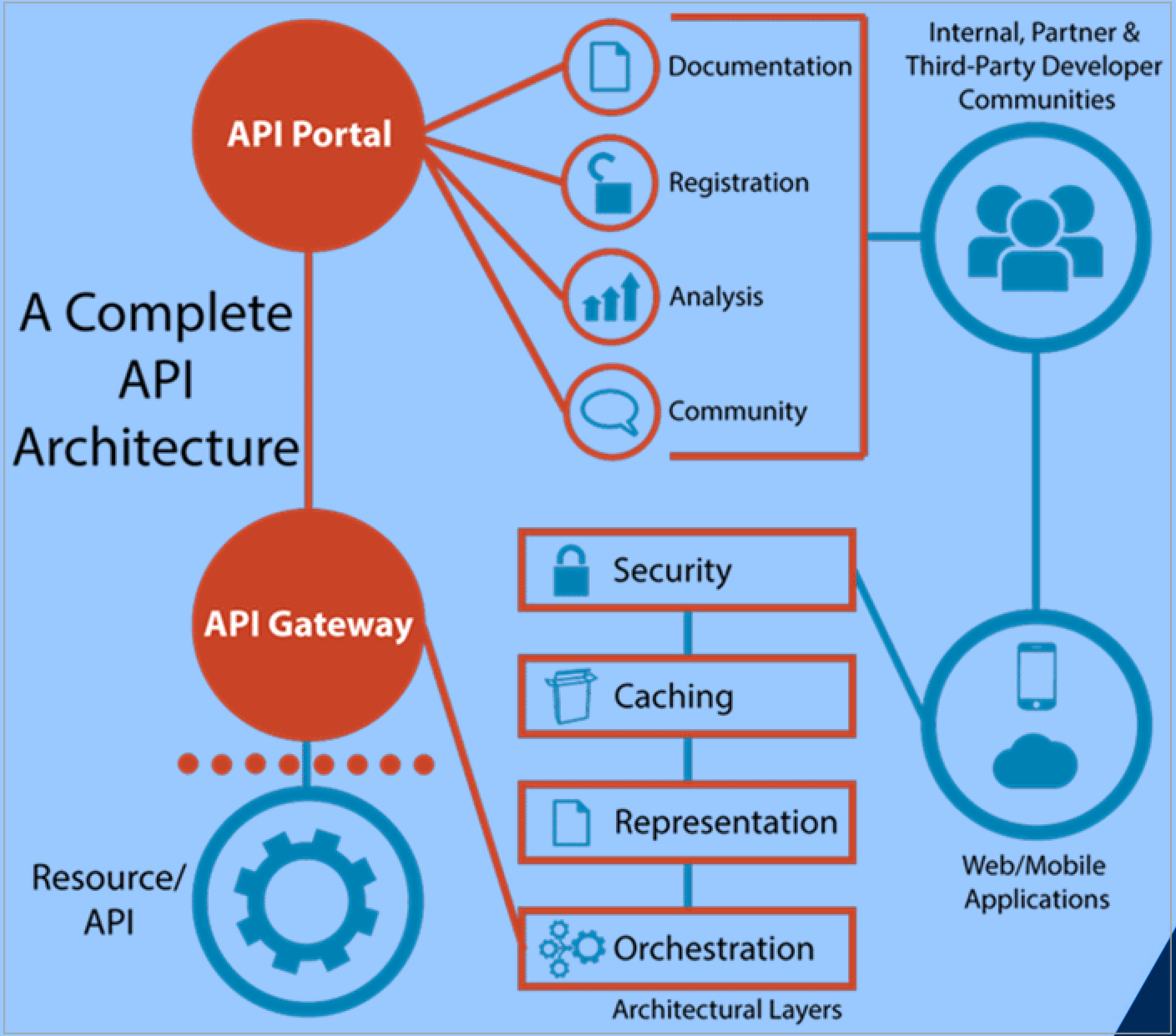
Schedule

- API
- REST
- Demo

API

- Application Programming Interface
- Software intermediary that allows two applications to talk to each other
- Using App like Instagram?
- Sending an instant message?
- Checking the weather on your phone?
- You’re using an API!!!


API Gateway
Search
elasticsearch
Review
mongoDB
Cart
redis
Order
PostgreSQL
payu
Recommendation
PredictionIO
Catalog
AEROSPIKE
Shipping
SAP
Inventory
PostgreSQL
User
mongoDB


API Portal
Documentation
Registration
Analysis
Community
Internal, Partner & Third-Party Developer Communities
A Complete API Architecture
API Gateway
Security
Caching
Representation
Orchestration
Architectural Layers
Web/Mobile Applications
Resource/ API

API FIRST!
MAKE BACKEND GREAT AGAIN!
makeameme.org

Representation XML

```
<endereco>
  <rua>
    Rua Jatobá
  </rua>
  <cidade>
    Ribeirão Preto
  </cidade>
</endereco>
```

Representation JSON

```
{
  endereco:
  {
  rua: Rua Jatobá,
  cidade: Ribeirão Preto
  }
}
```

Representation YAML

```
endereco:
 rua: rua Jatobá
 cidade: Ribeirão Preto
```

REST

- Roy Fielding
- Representational State Transfer
- Architectural style
- REST-compliant systems, often call **RESTful** systems
  - how they are stateless and separate the concerns of client and server

REST

- Communication between the client and server
  - Making requests
  - Sending responses

Making Requests

- REST requires that a client make a request to the server in order to retrieve or modify data on the server.
- A request consists of:
  - HTTP verb, which defines what kind of operation to perform
  - header, which allows the client to pass along information about the request
  - path to a resource
  - [optional] message body containing data

HTTP Verbs

- GET
  - retrieve a specific resource (by id) or a collection of resources
- POST
  - create a new resource
- PUT
  - update a specific resource (by id)
- DELETE
  - remove a specific resource by id

Headers and accept parameters

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types
- type/subtype
  - text/plain
  - application/json

Path

- Should contain the information necessary to locate a resource with the degree of specificity needed
- Conventionally, the first part of the path should be the plural form of the resource
  myecommerce.com/customers/223/orders/12
- To access a single resource, we would need to append an **id** to the path.
  - GET *myecommerce*.com/customers/:id
    - retrieves the item in the customers resource with the id specified.
  - DELETE *myecommerce*.com/customers/:id
    - deletes the item in the customers resource with the id specified.

Sending Responses

- Content Type
- Response Codes

Response Codes

- https://www.restapitutorial.com/httpstatuscodes.html
- For each HTTP verb, there are expected status codes a server should return upon success:
- GET - return 200 (OK)
- POST - return 201 (CREATED)
- PUT - return 200 (OK)
- DELETE - return 204 (NO CONTENT)
  - If the operation fails, return the most specific status code possible corresponding to the problem that was encountered.

Example - Request

```
POST https://myecommerce.com/customers
Body:
{
  “customer”: {
    “name” = “Eliézer Zarpelão”,
    “email” = “ezarpelao@unaerp.br”
  }
}
```

Example - Response

- Header:
  - 201 (CREATED)
  - Content-type: application/json
- Body:

  { id = 42 }

Example - Request

GET https://myecommerce.com/customers/42

Example - Response

- Header:
  - 200 (OK)
  - Content-type: application/json
- Body:

```
{
 “customer”: {
     “id” = 42,
     “name” = “Eliézer Zarpelão”,
     “email” = “ezarpelao@unaerp.br”
 }
}
```

Doubts

[Maior dúvida] – send by e-mail to ezarpelao@unaerp.br subject "Maior dúvida – 22/11/2019 - "+RA
Deadline: 25/11/2019