Module 2

Week 1 | Day 3 | **Bounded Contexts & REST APIs**

Sooo...what's on the agenda...

Sooo...what's on the agenda...

- Three Different Examples of Bounded Contexts

Sooo...what's on the agenda...

- Three Different Examples of Bounded Contexts
- Building a Simple Restful API

Question(s) of the Day.....

Bounded Contexts are about that business...

Bounded Contexts are about that business...

- What are the larger responsibilities of an application?
- Who need full access to a kind of data?
- What data is shared (through VOs)?

Every Bounded Context is a microservice.

Every microservice contains the databases, models and apis we need to support the Bounded Context.

Getting some **REST**

All data exchange happens through requests.

Each Request has access to:

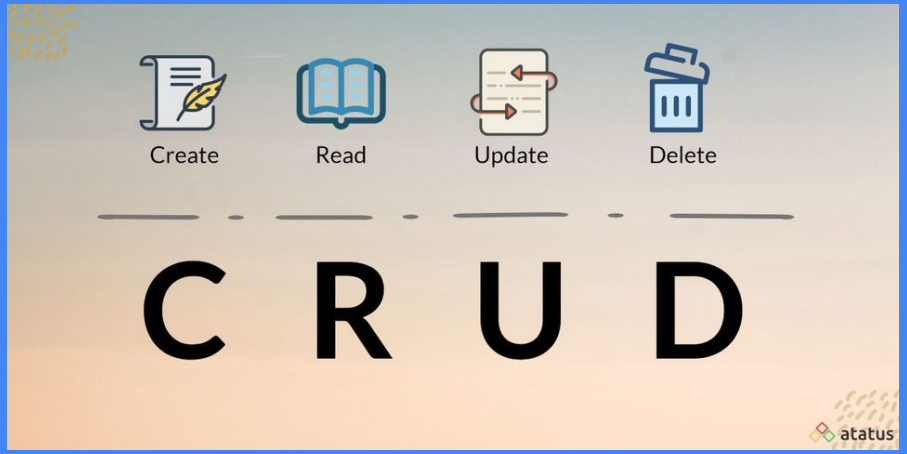
- A URL Path (with optional url parameter)
- A HTTP Method [POST, GET, PUT, DELETE]
- An optional Body

What we need is **CRUD:**

What we need is **CRUD:**

What we need is **CRUD:**

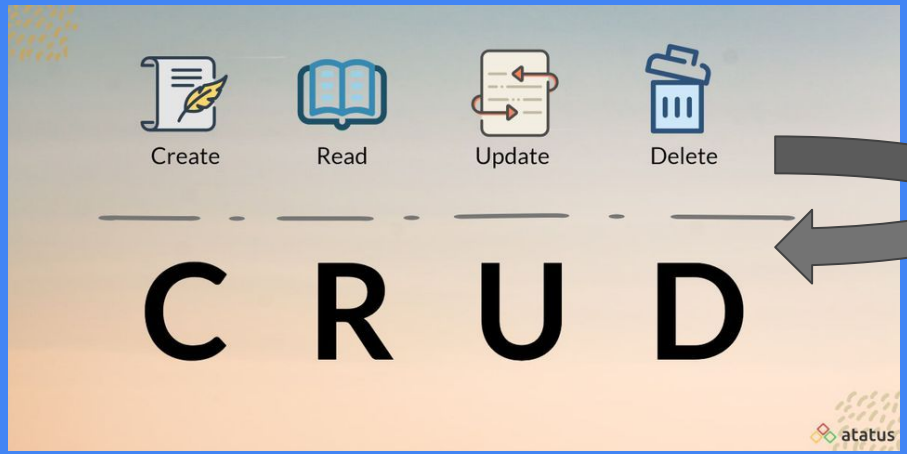
- Hotels
- Products
- Users
- **resource**



Each Request has access to:

- A URL Path (with optional url parameter)
- A HTTP Verb [POST, GET, PUT, DELETE]
- An optional Body

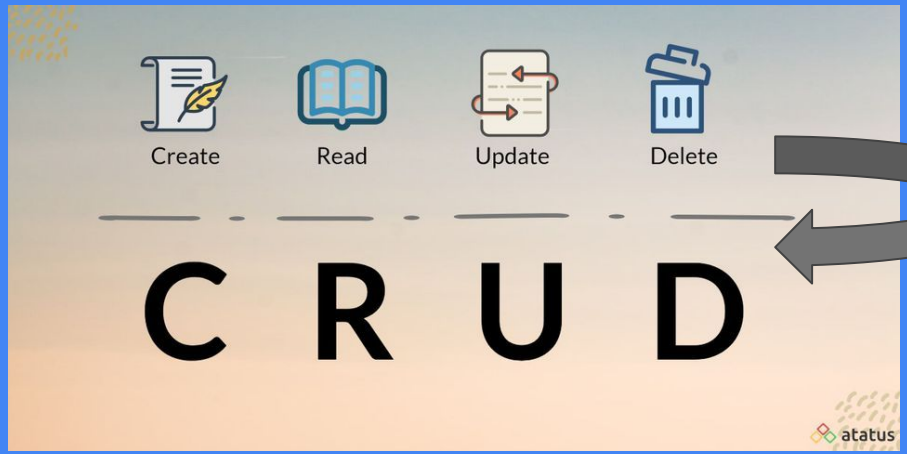
REST



Each Request has access to:

- A URL Path (with optional url parameter)
- A HTTP Verb [POST, GET, PUT, DELETE]
- An optional Body

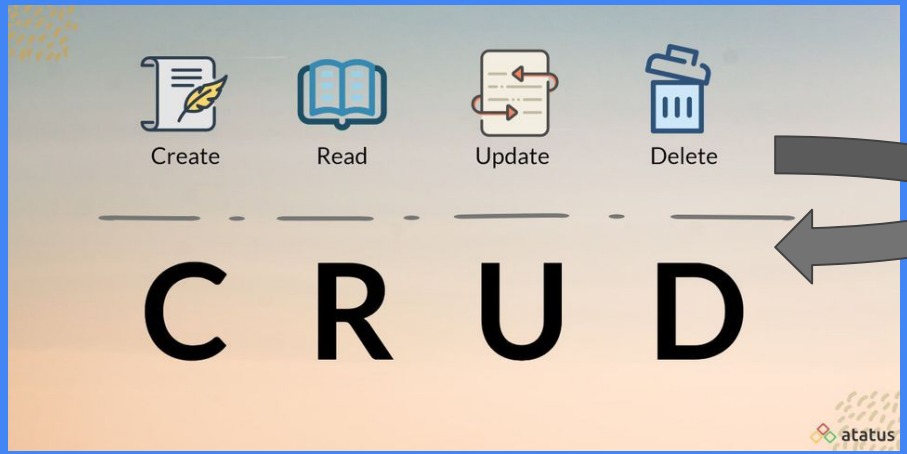
Representational **S**tate **T**ransfer



Each Request has access to:

- A URL Path (with optional url parameter)
- A HTTP Verb [POST, GET, PUT, DELETE]
- An optional Body

Representational **S**tate **T**ransfer
(a consistent url structure for CRUD on a resource)



Each Request has access to:

- A URL Path (with optional url parameter)
- A HTTP Verb [POST, GET, PUT, DELETE]
- An optional Body

Representational **S**tate **T**ransfer

(a consistent url structure for CRUD on a resource)

URL	HTTP Verb	POST Body	Result
/api/movies	GET	empty	Returns all movies
/api/movies	POST	JSON String	New movie Created
/api/movies/:id	GET	empty	Returns single movie
/api/movies/:id	PUT	JSON string	Updates an existing movie
/api/movies/:id	DELETE	empty	Deletes existing movie

Representational **S**tate **T**ransfer

(a consistent url structure for CRUD on a resource)

Action Name	HTTP Verb	Path	Description
index	GET	/songs	Display a list of all the songs
show	GET	/songs/:id	Display an individual song
create	POST	/songs	Manipulate the database to create a new song
edit	GET	/songs/:id/edit	Show an HTML form to edit an existing song
update	PUT/ PATCH	/songs/:id	Manipulate the database to update the song
destroy	DELETE	/songs/:id	Manipulate the database to delete the song

Representational **S**tate **T**ransfer
(a consistent url structure for CRUD on a resource)

Scenario	HTTP Method	URL Format
Add a customer	POST	/customers
Retrieve all customers	GET	/customers
Retrieve a specific customer	GET	/customers/{id}
Delete a customer	DELETE	/customers/{id}
Get all orders of a customer	GET	/customers/{id}/orders
Get specific order details of a customer	GET	/customers/{id}/orders/{id}
Get the address of a customer	GET	/customers/{id}/address
Edit details of a specific customer	PATCH	/customers/{id}

HOW TO MAKE AN API ENDPOINT!!!

HOW TO MAKE AN API ENDPOINT!!!

- Know what you are trying to do!!!! What is the feature?

HOW TO MAKE AN API ENDPOINT!!!

- Know what you are trying to do!!!! What is the feature?
- Make sure the model supports our feature.

HOW TO MAKE AN API ENDPOINT!!!

- Know what you are trying to do!!!! What is the feature?
- Make sure the model supports our feature.
- Make a url path for the endpoint

HOW TO MAKE AN API ENDPOINT!!!

- Know what you are trying to do!!!! What is the feature?
- Make sure the model supports our feature.
- Make a url path for the endpoint
- Make a view method for the endpoint

HOW TO MAKE AN API ENDPOINT!!!

- Know what you are trying to do!!!! What is the feature?
- Make sure the model supports our feature.
- Make a url path for the endpoint
- Make a view method for the endpoint
- Make sure endpoint returns a JSONRepsonse.

HOW TO MAKE AN API ENDPOINT!!!

- Know what you are trying to do!!!! What is the feature?
- Make sure the model supports our feature.
- Make a url path for the endpoint
- Make a view method for the endpoint
- Make sure endpoint returns a JSONRepsonse.
- Add code into the view method.

HOW TO MAKE AN API ENDPOINT!!!

- Know what you are trying to do!!!! What is the feature?
- Make sure the model supports our feature.
- Make a url path for the endpoint
- Make a view method for the endpoint
- Make sure endpoint returns a JSONRepsonse.
- Add code into the view method.
- TEST TEST TEST!!!