

Back-End Development

# Exploring

## #5 Loopback Module





# Loopback

**LoopBack** is a highly-extensible, open-source Node.js framework that enables you to create dynamic end-to-end REST APIs with little or no coding, access databases and also Incorporate model relationships and access controls for complex APIs (<http://loopback.io>).

## ■ Installation

```
$ npm install -g loopback-cli
```

# Create App

- On terminal, type:  
**\$ lb**

- *What's the name of your application?*  
**lin\_loopback**
- *Enter name of the directory to contain the project:*  
**lin\_loopback**
- *Which version of LoopBack would you like to use?*  
**3.x (current)**
- *What kind of application do you have in mind?*  
**api-server**

- Go to project directory:  
**\$ cd lin\_loopback**

- Run the application:  
**\$ node .**

- *Web server Listening at:* **<http://localhost:3000>**
- *Browse your REST API at* **<http://localhost:3000/explorer>**

# http://localhost:3000/explorer

**LoopBack API Explorer**

Token Set.  [Set Access Token](#)

**User**

Show/Hide | List Operations | Expand Operations

PATCH	/Users	Patch an existing model instance or insert a new one into the data source.
GET	/Users	Find all instances of the model matched by filter from the data source.
PUT	/Users	Replace an existing model instance or insert a new one into the data source.
POST	/Users	Create a new instance of the model and persist it into the data source.
PATCH	/Users/{id}	Patch attributes for a model instance and persist it into the data source.
GET	/Users/{id}	Find a model instance by {{id}} from the data source.
HEAD	/Users/{id}	Check whether a model instance exists in the data source.
PUT	/Users/{id}	Replace attributes for a model instance and persist it into the data source.
DELETE	/Users/{id}	Delete a model instance by {{id}} from the data source.
GET	/Users/{id}/accessTokens	Queries accessTokens of User.
POST	/Users/{id}/accessTokens	Creates a new instance in accessTokens of this model.
DELETE	/Users/{id}/accessTokens	Deletes all accessTokens of this model.
GET	/Users/{id}/accessTokens/{fk}	Find a related item by id for accessTokens.
PUT	/Users/{id}/accessTokens/{fk}	Update a related item by id for accessTokens.
DELETE	/Users/{id}/accessTokens/{fk}	Delete a related item by id for accessTokens.

# Set App ★

- By default, we must set token for accessing loopback API. To use its API without any authentication, do these steps:

Go to *projectDir/server/boot/authentication.js*  
Comment this line: *server.enableAuth();*

- Try to ***POST*** at ***POST/Users!***

```
{  
  "email" : "lintang@purwadhika.com",  
  "password" : "123456"  
}
```

- Try to post custom data, then ***GET*** at ***GET/Users!***

# POST

**LoopBack API Explorer**Token Not SetSet Access Token

**POST** /UsersCreate a new instance of the model and persist it into the data source.

### Parameters

Parameter	Value	Description	Parameter Type	Data Type				
data	<pre>{  "email": "lin@aku.id",  "password": "1234321"}  </pre> <div>Parameter content type: application/json</div>	Model instance data	body	<table><thead><tr><th>Model</th><th>Example Value</th></tr></thead><tbody><tr><td></td><td><pre>{  "realm": "string",  "username": "string",  "email": "string",  "emailVerified": true}</pre></td></tr></tbody></table>	Model	Example Value		<pre>{  "realm": "string",  "username": "string",  "email": "string",  "emailVerified": true}</pre>
Model	Example Value							
	<pre>{  "realm": "string",  "username": "string",  "email": "string",  "emailVerified": true}</pre>							

Try it out!

[Hide Response](#)

### Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \  "email": "lin%40aku.id", \  "password": "1234321" \}' 'http://localhost:3000/api/Users'
```

### Request URL

```
http://localhost:3000/api/Users
```

### Response Body

```
{  "email": "lin@aku.id",  "id": 3}
```

### Response Code

```
200
```

- By default, “email” & “password” have to be declared correctly!
- It will response an “id” automatically!
- You can type other JSON data after “email” & “password” too.

# GET

## LoopBack API Explorer

Token Not Set

Set Access Token

GET /Users

Find all instances of the model matched by filter from the data source.

Try it out!

[Hide Response](#)

### Curl

```
curl -X GET --header 'Accept: application/json' 'http://localhost:3000/api/Users'
```

### Request URL

```
http://localhost:3000/api/Users
```

### Response Body

```
[
  {
    "email": "sasas@kokok.mama",
    "id": 1
  },
  {
    "email": "sasaas@kokoak.mama",
    "id": 2
  },
  {
    "email": "lin@aku.id",
    "id": 3
  }
]
```

■ It will response an “id” & hide the “password” automatically!



- A powerful GUI platform to make API development faster & easier, from building API requests through testing, documentation & sharing.
- Postman has features for every API developer: request building, tests & pre-request scripts, variables, environments & request descriptions, designed to work seamlessly together.
- Download from [getpostman.com](https://getpostman.com)

**Try to POST & GET  
with POSTMAN!**

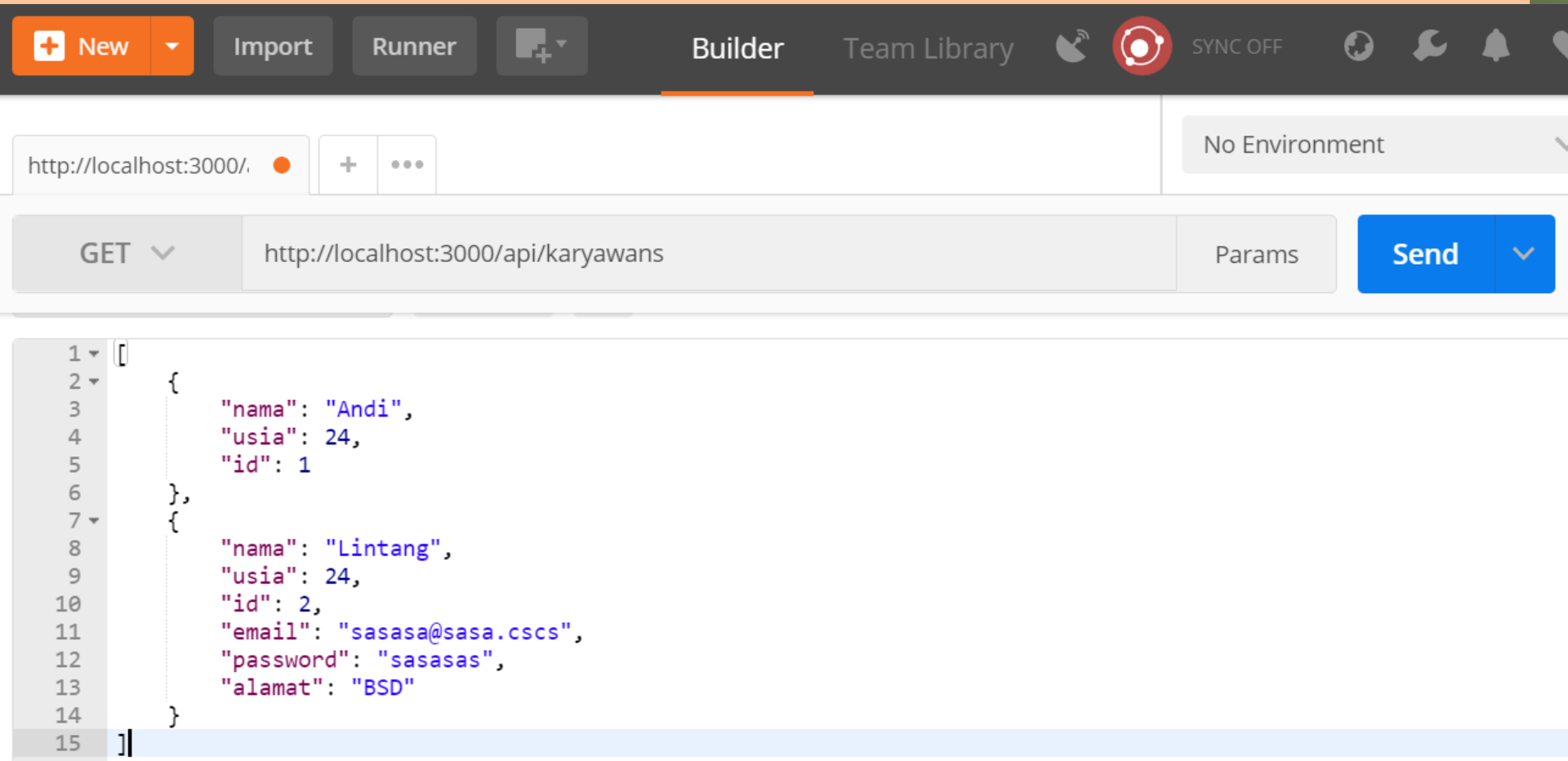


# POST with POSTMAN

The screenshot displays the Postman application interface. At the top, there is a dark navigation bar with buttons for 'New', 'Import', 'Runner', 'Builder' (which is highlighted), 'Team Library', and a 'SYNC OFF' status. To the right of these buttons are icons for a globe, a refresh symbol, a bell, and a heart. Below the navigation bar, the main workspace is divided into several sections. On the left, there is a URL bar showing 'http://localhost:3000/'. To its right is a dropdown menu labeled 'No Environment'. Below the URL bar, there is a 'POST' method selector and a URL field containing 'http://localhost:3000/api/karyawans'. To the right of the URL field is a 'Params' tab and a blue 'Send' button. Below these elements are tabs for 'Authorization', 'Headers (1)', 'Body' (which is selected and highlighted with a blue dot), 'Pre-request Script', and 'Tests'. Under the 'Body' tab, there are radio buttons for different body types: 'form-data', 'x-www-form-urlencoded', 'raw' (which is selected), and 'binary'. To the right of these radio buttons is a dropdown menu showing 'JSON (application/json)'. The main area of the 'Body' tab contains a JSON object being edited in a code editor. The JSON object is as follows:

```
1 {  
2   "nama": "Lintang",  
3   "usia": 24,  
4   "email": "sasasa@sasa.cscs",  
5   "password": "sasasas",  
6   "alamat": "BSD"  
7 }
```

# GET with POSTMAN



# Create Model

- On directory project, type:  
**\$ lb model**

- [?] Enter the model name: *karyawan*
  - [?] Select the data-source to attach person to: *db*
  - [?] Select model's base class (*PersistedModel*)
  - [?] Expose person via the REST API? *Yes*
  - [?] Custom plural form (used to build REST URL):
  - [?] Common model or server only? *Common*
- Let's add some person properties now.

Enter an empty property name when done.

- [?] Property name: *nama*
- [?] Property type: (Use arrow keys) *string*
- [?] Required? (y/N) *y*

- After done, there is model folder including js & json files. Try to run its app: **\$ node .**

# Create Model

LoopBack API Explorer

Token Not Set

accessToken

lin\_loopback

karyawan

Show/Hide | List Operations

User

Show/Hide | List Operations

[ BASE URL: /api , API VERSION: 1.0.0 ]

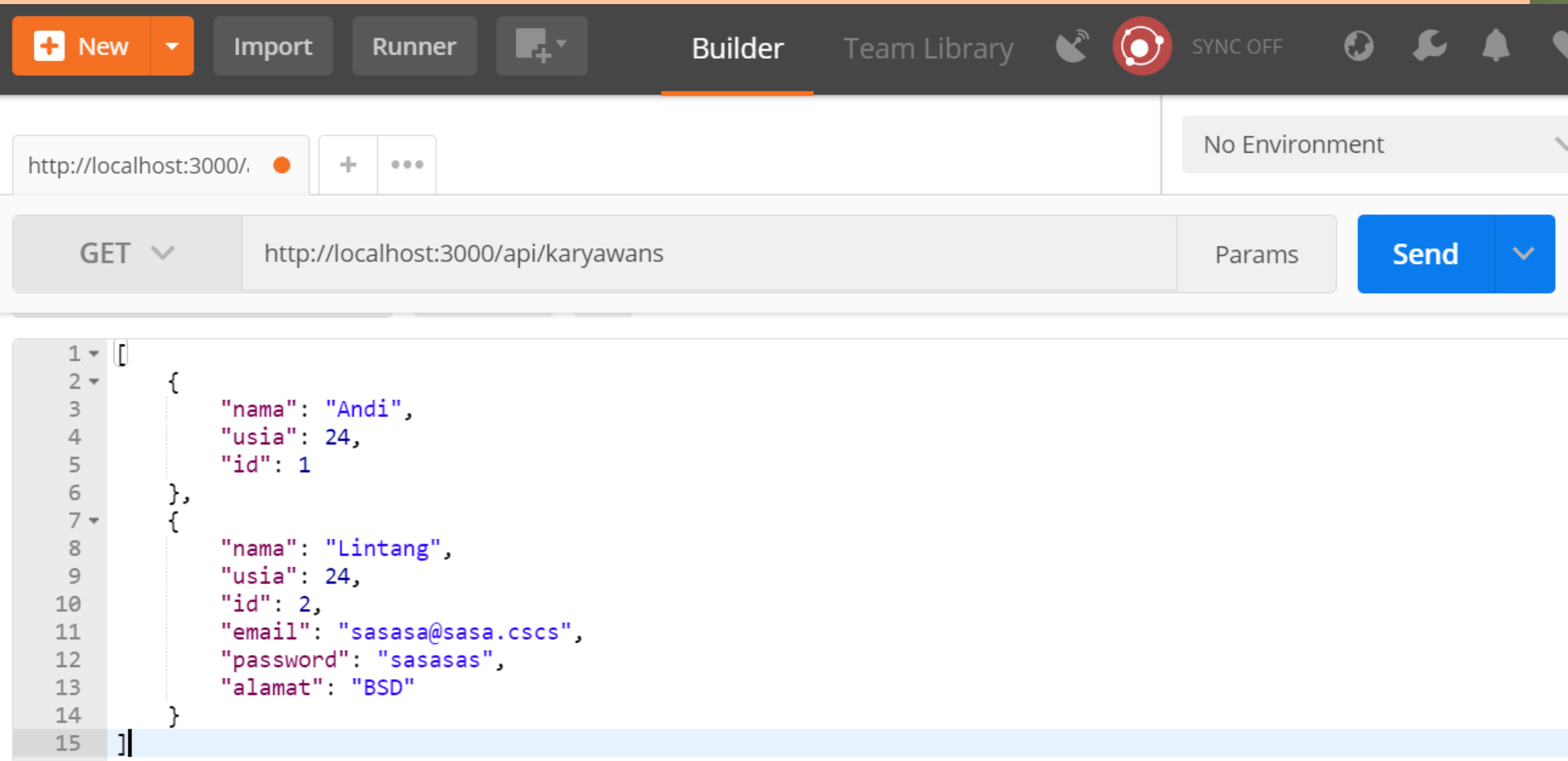
**Try to POST & GET  
with POSTMAN!**

# POST with POSTMAN

The screenshot displays the Postman application interface. At the top, there is a dark navigation bar with buttons for 'New', 'Import', 'Runner', 'Builder' (which is highlighted), 'Team Library', and a 'SYNC OFF' status. To the right of these buttons are icons for a globe, a refresh symbol, a bell, and a heart. Below the navigation bar, the main workspace is divided into several sections. On the left, there is a URL bar showing 'http://localhost:3000/'. To its right is a dropdown menu currently set to 'No Environment'. Below the URL bar, there is a row of tabs: 'POST' (selected), 'Params', 'Send' (a blue button), and a partially visible 'S'. Below these tabs, there is another row of tabs: 'Authorization', 'Headers (1)', 'Body' (selected), 'Pre-request Script', and 'Tests'. Below the 'Body' tab, there is a row of radio buttons for selecting the request body type: 'form-data', 'x-www-form-urlencoded', 'raw' (selected), and 'binary'. To the right of these radio buttons is a dropdown menu set to 'JSON (application/json)'. The main area of the 'Body' tab contains a JSON object being edited in a code editor. The JSON is as follows: 

```
1 {  
2   "nama": "Lintang",  
3   "usia": 24,  
4   "email": "sasasa@sasa.cscs",  
5   "password": "sasasas",  
6   "alamat": "BSD"  
7 }
```

# GET with POSTMAN



Back-End Development

# Exploring

## #5 Loopback Module

