

📌 Design REST API for booking system – part 1 (<https://bestin-it.com/building-simple-laravel-5-4-api-for-books-management-with-books-readers-and-librarians/>)

👤 Artur Poniedziałek (<https://Bestin-It.Com/Author/Bestfuturox/>) 📅 30 Nov, 2018 📁 PHP Laravel (<https://Bestin-It.Com/Category/Php-Laravel/>)

💬 🗨 No Comment (<https://Bestin-It.Com/Building-Simple-Laravel-5-4-API-For-Books-Management-With-Books-Readers-And-Librarians/#Respond>)

Share me please

General scope for REST API in Laravel 5.4

This series of articles should help you easily design REST API in PHP Laravel framework using minimum effort and saving your development time. Skills for building quickly any kind of API are especially now very important when everything is decentralized.

Data Management

Ad Compare courses
search from multiple sources

Coursary

Open

At the beginning will focus on list of requirements needed for normal use of system for books management. Main roles and responsibilities will be assigned. All these requirements will be created independently from any physical system implementation.

After that we will define all endpoints in REST API that will enable most of functionality. It is also important to define users and their access to every endpoint.

Later we will map all endpoints into database structure with all fields and tables definitions. All tables relations should also be defined.

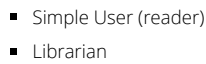
At the end we will create Laravel 5.4 project and start working on implementation of all the previous steps.

Main requirements for Rest API

I tried to find most valuable requirements that API should passed and below you can see all of them. This is of course some subset off all possible requirements and feel free to add more of them in your use of system when needed.

- we should be able to create books
- we should be able to create users
- we should have two types of users: simple user and let's name it reader and more sophisticated user let's name it librarian
- simple user should be able to make reservations and borrow books
- librarian will be responsible for books and physically booking books for simple user
- simple user can cancel any reservation at any moment
- simple user can make more than one reservation for more books
- simple user can book a book only for a month
- librarian can be able to define all fields for a book
- if simple user will not receive the book on time he cannot book other books
- it should be able possible to search through author, title, isbn, year
- it should be possible to search through username or email

Now when all main requirements are defined we can try to analyze what actors are in the system. I selected some of them:



They can interact each other by making some actions. We can name this actions as use cases.

Use cases diagram show us more clearly main interactions that we can observe in the system. It's definitely easier to understand what is going on.

You can modify designed by me uml diagram using beatfull system www.draw.io under [the link \(https://www.draw.io/](https://www.draw.io/)
lightbox=1&highlight=0000ff&edit= blank&layers=1&nav=1&title=Untitled%20Diagram.drawio#R5VpNb6MwEP01Oa4EGPJxbNj2e2ilqtGqe6scmAarBiNj
kr5%2Ffg0YAnagtmxLopBLzMMe2%2B2%2FZM2OLEVpEu98c%2BEfC4COHCvYjDlyHfCsQ3I4X68lojnTEpgzUIQOtYeWjI3qFogNCMBpAoricEYFSRpgz6LY
%2FBFC8Qcs2272jQjOQtI8BoMYOJlaqKPjBBHlU49a4%2FfAfMhVc%2B2pd5UeKqsgDTEAds2IH01QgvOmChL0W4BNCEvczv1O2%2FrgXGlxWcajOePNHN
n6dP1ww16e%2Fg7e1v4v5VDaaZmvAD4AC4Grj4rXiOo0%2EyYhbcR18wPkLzDXBBjFO3eAX0ngVEEBbL KismBltkBZq%2FmGP%2Fzc1ZFgclRvN20hp6L
n4NGxeUrPO2giUSDUVE5YMitwTlMSwgOW1JkgGLdvC7I027jpujtiBR5D4q6yGsyJUKmpZ1urtuxcZKShs6DtWGF6laI0b3jMvC4r8LwjHGLckhXhN0D47LV
wxyrcmBjLEWODYLzwbhx24Bhf06rply9ltvyf5NiOCJgn28zdb6RnbRH4DZ47ntTizkW1wVuijmneT5HmGaOtAXM%2FINiKsZf0pOhYlOfldl0NuuYsj1eKrtNia4
KQZnbE3jEcsABp7BLoFIUu%2FSVAgFo7zuWtPgTDWj4tjy%2FqYRa9PvzY9EO1FxuPTWHNtpDrmmzZfa65qrMGXXf4BU44CizNuNvNrXNNObDfX0ledpua
DtHaIive5T20xAtiBi4KKlF%2BjSpzhNiX%2BIGAim482HtDsmfWjWfCaByh2waZs%2FRlXq4Z4R2XHNuo0071hf5MpEyjLug2rVPNdohnTHUa%2F4ypDAfA3C
MEQoU0%2F2P8RyByiWa32TWPoe%2BmmxzCz1%2FMRYNbF0jrUkVR%2FC%2BhLlZjHPX6xpv7EQ06hvN2im6OCnlu4GZ13F0gyhvt2geRI4O7Fc3XvZHcV
CkyMnGLMBiKW7wa4jBtLuEu1jv2jYr%2F2DEqtrgqHvrL4TjGrggxjL3xBdxeo7wai0GZjYBseffmt63ATDGcAnhqf7%2BrXap8Vv9MTJl5j1gBuMDznm9yvgfIt
nAE6jyUf918DINX331Sgg38%3D)

Now we are much more closer to define main actions needed in defined by us Rest API. Main endpoints of the API will be joined with such resource:

- **user** – standard actions like create, delete, update
- **book** – standard actions like create, delete, update
- **book** – borrow/return

- **book** – search books by different criterias (through different fields of a book)
- **reservation** – we can make as many reservations as we want, also we should be able to cancel any reservation at any moment

So let's start defining every endpoints.

List of endpoints for USER resource

As we defined the list of possible actions to do with user is very simple. We need to have possibility to create full CRUD for user resource.

| Method | Endpoint and description |
|--------|--|
| POST | /user – creates new user |
| GET | /user – gets collection (list) of users |
| GET | /user/:id – gets user with concrete id |
| PUT | /user/:id – updates user fields |
| DELETE | /user/:id – deletes user by id |

List of endpoints for BOOK resource

Endpoints for book looks very similar to user. We add some more parameters especially for borrow and return purposes.

| Method | Endpoint and description |
|--------|--|
| POST | /book – creates new book |
| GET | /book – gets collection (list) of users |
| GET | /book/:id – gets book with concrete id |
| PUT | /book/:id – updates book fields |
| DELETE | /book/:id – deletes book by id |
| POST | /book/borrow/:id – borrows book by id |
| POST | /book/return/:id – returns book by id |

Let's focus on two last endpoints that I created to distinguish the standard book CRUD from additional actions like borrowing and returning. Both this two actions are POST endpoints because we want to make a kind of update on selected book that we borrowed or returned it.

List of endpoints for RESERVATION resource

Reservation resource connect directly books with users. In order to make any reservation we need to be logged as a concrete user (known user_id) then we can make reservation for any book.

| Method | Endpoint and description |
|--------|--|
| POST | /reservation/:user_id/:book_id – creates reservations for user and book |
| GET | /reservation – gets list of all reservations |
| GET | /reservation/:user_id – gets list of all reservations for the selected user |
| DELETE | /reservation/:reservation_id – removes selected reservation |

We do not need to define updates actions because user can simple cancel his reservation by removing it. If the action will be needed later than we will add it.

Defining access level for REST API Endpoints

Now we have defined all endpoints in our REST API. The only question we should make is for access different type of users to endpoints.

I simplify this process and we can requirements and assume that:

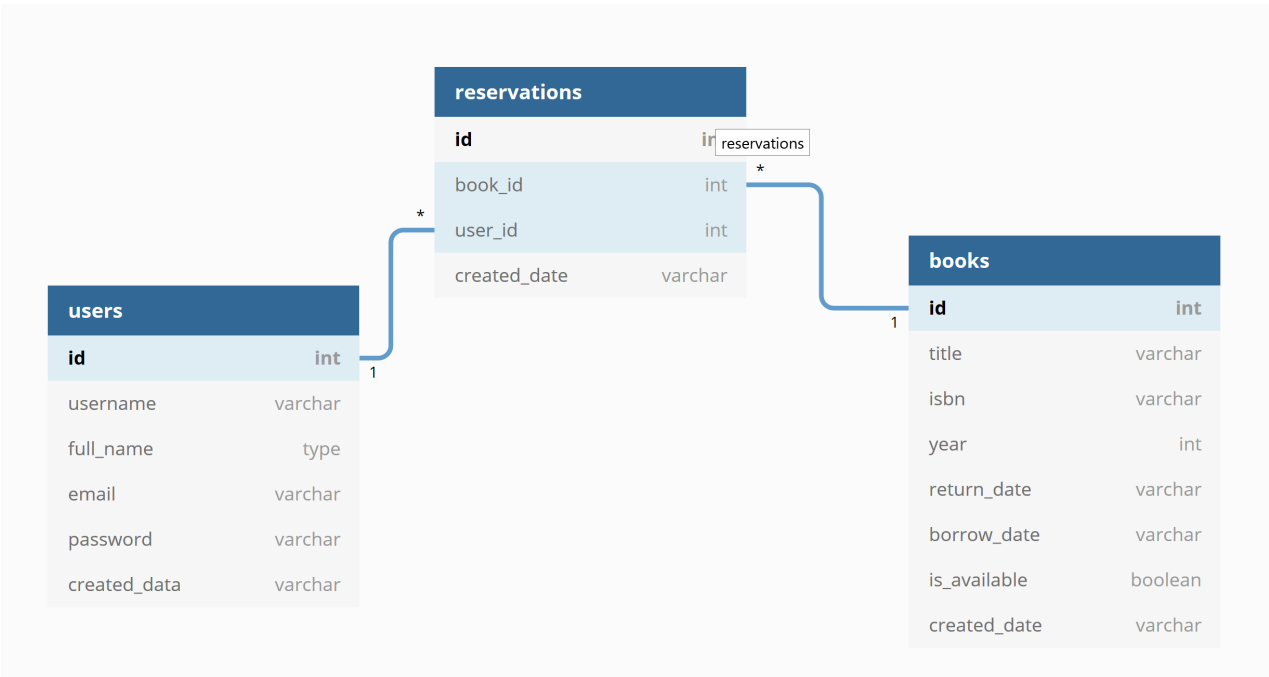
- reader has access to:
 - all information about himself but cannot removes himself (only librarian can remove such user)
 - all information about books but cannot remove, update and delete them

- reservations created by him (so cannot remove reservations created by another user)
- librarian has access to all data and can modify/delete everything in the system

Designing database for REST API

These simple requirements allows us to start designing our database. We have to synthesize all the information we have and think about what fields should each table have. We also need to design relationships between the tables.

So let's start from preparing diagrams by wonderfull website: <https://dbdiagram.io>. Below you can see the draft of designed by me database diagram:



The diagram is also shared here: <https://dbdiagram.io/d/5d29c566ced98361d6dc9eb7> (<https://dbdiagram.io/d/5d29c566ced98361d6dc9eb7>).

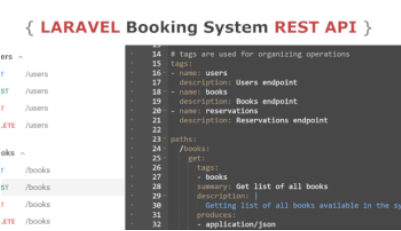
You can extend easily the definition of database defining more fields or complicating the logic of REST API in real business application. My intention is not to build too many logic but just to show the main idea and step you need to go through in designing process.

Finally we can also see the definition of database as a schema:

```
public function up() { Schema::create('users', function (Blueprint $table) { $table->increments('id'); $table->string('name'); $table->string('email')->unique(); $table->string('password'); $table->rememberToken(); $table->timestamps(); }); } public function down() { Schema::dropIfExists('users'); } test
```

In the [next post](https://bestin-it.com/laravel-5-4-api-for-books-management-new-project-and-connection-to-database/) (<https://bestin-it.com/laravel-5-4-api-for-books-management-new-project-and-connection-to-database/>) we will create Laravel project and start defining and creating designed database with main endpoints.

Related posts

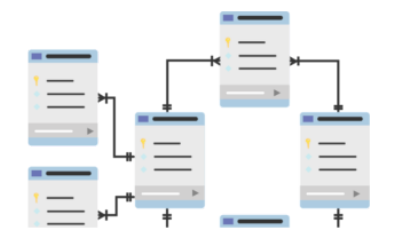


{ LARAVEL Booking System REST API }

Thumbnail showing code snippets for routes and controllers in a Laravel project.

<https://bestin-it.com/implementing-rest-api-endpoints-laravel-5-4-api-for-books-management/>

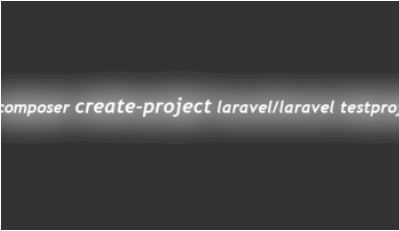
Implementing api endpoints in Laravel REST API - part 3 (<https://bestin-it.com/implementing-rest-api-endpoints-laravel-5-4-api-for-books-management/>)



Thumbnail showing a database diagram for the books management system.

<https://bestin-it.com/laravel-5-4-api-for-books-management-new-project-and-connection-to-database/>

Laravel 5.4 API for Books management - part 2 - new project and connection to database (<https://bestin-it.com/laravel-5-4-api-for-books-management-new-project-and-connection-to-database/>)



Thumbnail showing a terminal window with the command `composer create-project laravel/laravel testproj`.

<https://bestin-it.com/how-to-run-laravel-project-with-several-commands/>

How to run Laravel project with several commands (<https://bestin-it.com/how-to-run-laravel-project-with-several-commands/>)

▼

← previous article (<https://bestin-it.com/php-artisan-with-laravel-5-4-list-of-mostly-used-commands/>)

next article → (<https://bestin-it.com/laravel-5-4-api-for-books-management-new-project-and-connection-to-database/>)

Leave A Reply

Comment

Name

Email

Website url

☐ Save my name, email, and website in this browser for the next time I comment.

SEND COMMENT

Artur Poniedziałek

Software Developer

Don't stop learning. Open your mind to stay better in IT.



(<https://www.facebook.com/bestinitcom/>)(http://twitter.com/bestin_it)(<https://www.linkedin.com>

Blog Search

poniedziałek-738972

Join to the best IT & makers

Categories

3d Printing (<https://Bestin-It.Com/Category/3d-Printing/>)

Databases (<https://Bestin-It.Com/Category/Databases/>)

General (<https://Bestin-It.Com/Category/General/>)

Machine Learning (<https://Bestin-It.Com/Category/Machine-Learning/>)

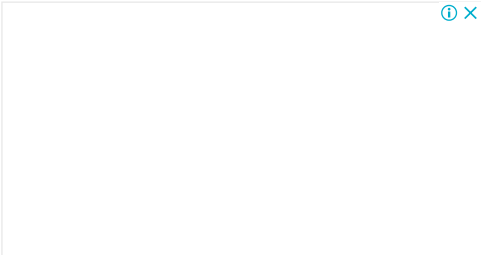
Management Systems (<https://Bestin-It.Com/Category/Management-Systems/>)

Motivation (<https://Bestin-It.Com/Category/Motivation/>)

PHP Laravel (<https://Bestin-It.Com/Category/Php-Laravel/>)

Python (<https://Bestin-It.Com/Category/Python/>)

Tools (<https://Bestin-It.Com/Category/Tools/>)



How Golfers Over 55 Drive 300+
Why big rotation swings taught by young i
BAD for older golfers. Use this instead.

Performance Golf Zone **Learn**

Recent Posts

Managing postgresql database using SQLAlchemy ORM in Python (<https://bestin-it.com/managing-postgresql-database-using-sqlalchemy-orm-in-python/>)

Python Excel Tutorial with Pandas Library (<https://bestin-it.com/reading-and-writing-data-using-python-and-excel-datasheet/>)

Effective running and debugging python scripts in Visual Studio Code (<https://bestin-it.com/running-and-debugging-python-scripts-in-visual-studio-code/>)

Learn how to install python on Windows (<https://bestin-it.com/learn-how-to-install-python-on-windows/>)

How to install PostgreSQL and create first database with pgAdmin (<https://bestin-it.com/how-to-install-postgresql-and-create-first-database-with-pgadmin/>)

Recent Comments

Artur Poniedziałek (<https://bestin-it.com>) ON Confluence macros overview (<https://bestin-it.com/confluence-macros-overview/#comment-3252>)

Artur Poniedziałek (<https://bestin-it.com>) ON How to build best 3d printer from scratch – HyperCube Evolution (<https://bestin-it.com/how-to-build-best-3d-printer-from-scratch-hypercube-evolution/#comment-3011>)

Amos ON How to build best 3d printer from scratch – HyperCube Evolution (<https://bestin-it.com/how-to-build-best-3d-printer-from-scratch-hypercube-evolution/#comment-2996>)

Brandie ON Confluence macros overview (<https://bestin-it.com/confluence-macros-overview/#comment-2962>)

altyazili ON Managing postgresql database using SQLAlchemy ORM in Python (<https://bestin-it.com/managing-postgresql-database-using-sqlalchemy-orm-in-python/#comment-2536>)

Archives

April 2020 (<https://bestin-it.com/2020/04/>)

[March 2020 \(https://bestin-it.com/2020/03/\)](https://bestin-it.com/2020/03/)

[February 2020 \(https://bestin-it.com/2020/02/\)](https://bestin-it.com/2020/02/)

[August 2019 \(https://bestin-it.com/2019/08/\)](https://bestin-it.com/2019/08/)

[July 2019 \(https://bestin-it.com/2019/07/\)](https://bestin-it.com/2019/07/)

[June 2019 \(https://bestin-it.com/2019/06/\)](https://bestin-it.com/2019/06/)

[November 2018 \(https://bestin-it.com/2018/11/\)](https://bestin-it.com/2018/11/)

[October 2018 \(https://bestin-it.com/2018/10/\)](https://bestin-it.com/2018/10/)

[September 2018 \(https://bestin-it.com/2018/09/\)](https://bestin-it.com/2018/09/)

[August 2017 \(https://bestin-it.com/2017/08/\)](https://bestin-it.com/2017/08/)

[July 2017 \(https://bestin-it.com/2017/07/\)](https://bestin-it.com/2017/07/)

[June 2017 \(https://bestin-it.com/2017/06/\)](https://bestin-it.com/2017/06/)

Meta

[Log in \(https://bestin-it.com/wp-login.php\)](https://bestin-it.com/wp-login.php)

[Entries feed \(https://bestin-it.com/feed/\)](https://bestin-it.com/feed/)

[Comments feed \(https://bestin-it.com/comments/feed/\)](https://bestin-it.com/comments/feed/)

[WordPress.org \(https://en-gb.wordpress.org/\)](https://en-gb.wordpress.org/)

Copyright © 2017-2021 BestIn-It.com

[Terms and Conditions \(https://bestin-it.com/terms-and-conditions/\)](https://bestin-it.com/terms-and-conditions/) [Privacy Policy \(https://bestin-it.com/privacy-policy/\)](https://bestin-it.com/privacy-policy/)

[Get in Touch \(https://bestin-it.com/get-in-touch/\)](https://bestin-it.com/get-in-touch/)