

# Sistem Reservasi Hotel

## Migration

### 1. Tabel Hotels

```
/**
 * Run the migrations.
 */
public function up(): void
{
    Schema::create('hotels', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->longText('address');
        $table->string('email');
        $table->timestamps();
    });
}
```

### 2. Tabel Rooms

```
/**
 * Run the migrations.
 */
public function up(): void
{
    Schema::create('rooms', function (Blueprint $table) {
        $table->id();
        $table->foreignId('hotel_id')->constrained();
        $table->integer('number');
        $table->enum('type', ['Economy', 'Premium', 'Luxury']);
        $table->integer('price');
        $table->timestamps();
    });
}
```

### 3. Tabel Guests

```
/**
 * Run the migrations.
 */
public function up(): void
{
    Schema::create('guests', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->longText('address');
        $table->string('email');
        $table->timestamps();
    });
}
```

### 4. Tabel Reservations

```

/**
 * Run the migrations.
 */
public function up(): void
{
    Schema::create('reservations', function (Blueprint $table) {
        $table->id();
        $table->foreignId('room_id')->constrained();
        $table->foreignId('guest_id')->constrained();
        $table->date('check_in');
        $table->date('check_out');
        $table->timestamps();
    });
}

```

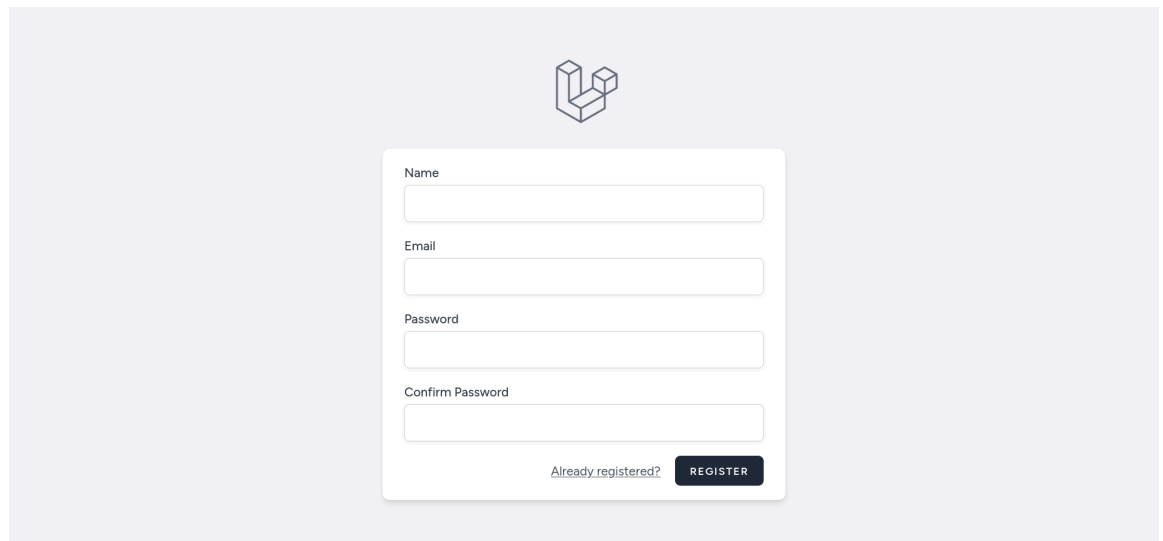
## CRUD

### 1. Fitur Autentikasi

Menggunakan laravel Breeze


#### a. Register

Menerima input nama, email dan password



#### b. Login

Menerima input email dan password serta memiliki checkbox untuk menyimpan session



Email


Password

☐ Remember me

LOG IN

c. Profile

Terdapat fitur Update dan delete user

 Dashboard Hotels Rooms Guests Reservations Araft

Profile

Profile Information

Update your account's profile information and email address.

Name

Araft


Email

arafi@gmail.com

SAVE

2. CRUD Hotel

Tampilan index hotel

 Dashboard Hotels Rooms Guests Reservations Araft

Hotels

CREATE HOTEL

NAME	ADDRESS	EMAIL	ACTIONS
Horison	Jakarta	horison@hotel.com	<div><div>VIEW</div><div>EDIT</div><div>DELETE</div></div>

a. Create

Kode untuk membuat data hotel baru di HotelController.php

```
public function store(Request $request)
{
    $validated = $request->validate([
        'name' => 'required',
        'address' => 'required',
        'email' => 'required',
    ]);

    $store = Hotel::create($validated);

    if ($store){
        return redirect()->route('hotels.index')->with('success', 'Hotel created successfully');
    }else{
        return redirect()->route('hotels.index')->with('error', 'Hotel failed to create');
    }
}
```

Tampilan Create di views

The screenshot shows a web application interface with a navigation bar (Dashboard, Hotels, Rooms, Guests, Reservations) and a user profile (Arafi). The 'Hotels' section is active. A 'CREATE HOTEL' button is visible. A modal titled 'Create Hotel' is open, featuring input fields for 'Hotel Name', 'Hotel Address', and 'Hotel Email', along with 'CANCEL' and 'CREATE' buttons. A table in the background displays hotel data:

NAME	ADDRESS
Horison	Jakarta

b. Read

Kode untuk melihat data hotel di HotelController.php

```
public function getDetails(Hotel $hotel)
{
    return response()->json($hotel);
}
```

Tampilan Read di views

The screenshot shows the same web application interface. A modal titled 'View Hotel' is open, displaying the details of a hotel: 'Horison' for the name, 'Jakarta' for the address, and 'horison@hotel.com' for the email. A 'BACK' button is at the bottom. The background shows the same hotel table as the previous screenshot:

NAME	ADDRESS
Horison	Jakarta

### c. Update

Kode untuk mengupdate data hotel di HotelController.php

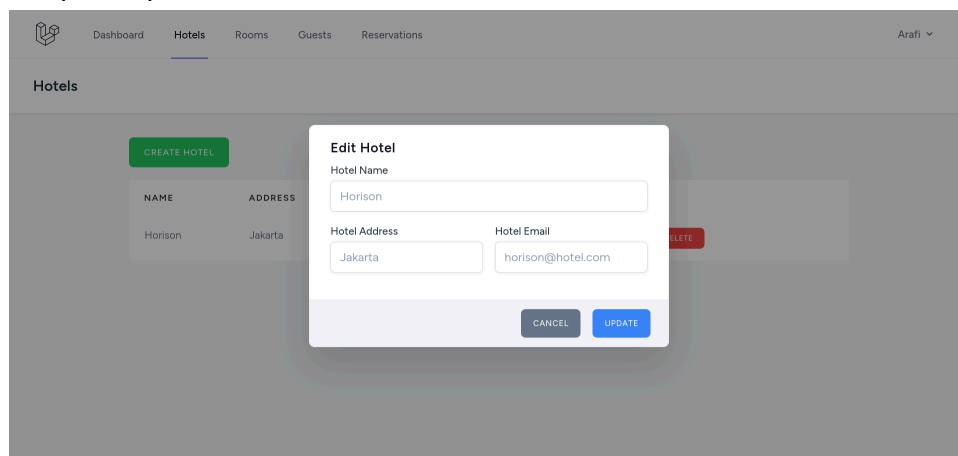
```
public function update(Request $request)
{
    $validated = $request->validate([
        'name' => 'required',
        'address' => 'required',
        'email' => 'required',
    ]);

    $search = Hotel::find($request->id);
    if (!$search){
        return redirect()->route('hotels.index')->with('error', 'Hotel not found');
    }

    $update = Hotel::where('id', $request->id)->update($validated);

    if ($update){
        return redirect()->route('hotels.index')->with('success', 'Hotel updated successfully');
    }else{
        return redirect()->route('hotels.index')->with('error', 'Hotel failed to update');
    }
}
```

Tampilan update di views



The screenshot displays a web interface for managing hotels. A modal titled "Edit Hotel" is open, allowing for the update of a hotel's details. The modal includes three input fields: "Hotel Name" (containing "Horison"), "Hotel Address" (containing "Jakarta"), and "Hotel Email" (containing "horison@hotel.com"). At the bottom of the modal are "CANCEL" and "UPDATE" buttons. In the background, a table with the headers "NAME" and "ADDRESS" is visible, showing a single entry: "Horison" and "Jakarta". A "CREATE HOTEL" button is also present in the top left of the modal area.

### d. Delete

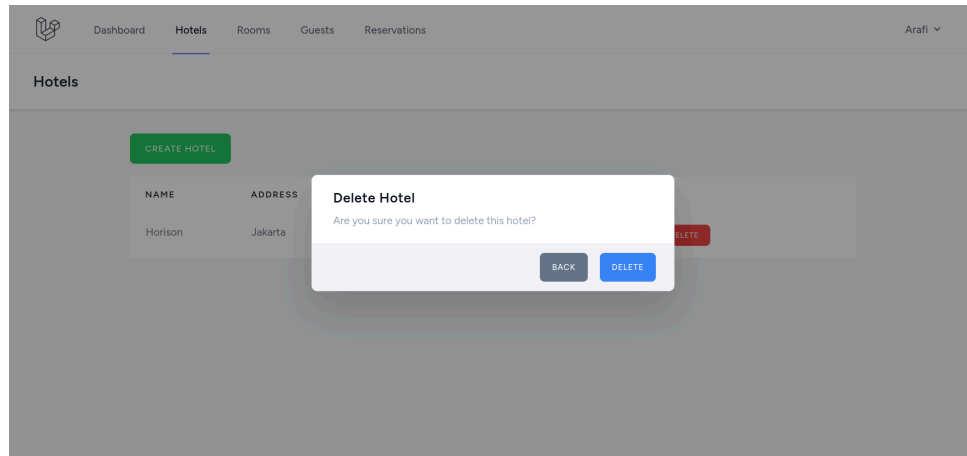
Kode untuk menghapus data hotel di HotelController.php

```
public function destroy(Request $request)
{
    $find = Hotel::find($request->id);
    if (!$find){
        return redirect()->route('hotels.index')->with('error', 'Hotel not found');
    }

    $destroy = Hotel::destroy($request->id);

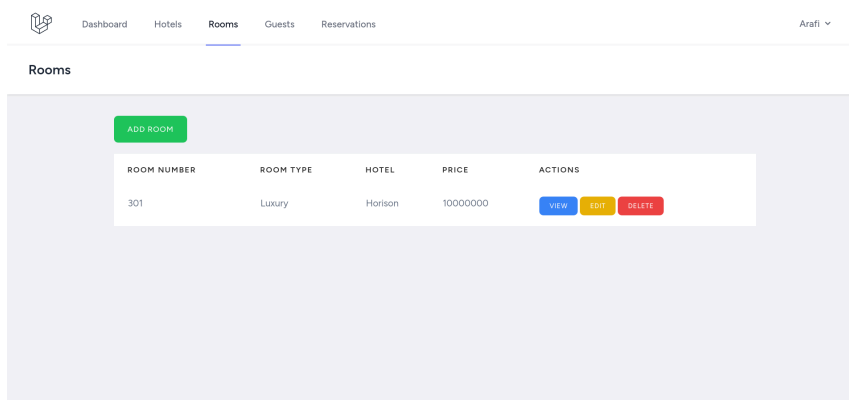
    if ($destroy){
        return redirect()->route('hotels.index')->with('success', 'Hotel deleted successfully');
    }else{
        return redirect()->route('hotels.index')->with('error', 'Hotel failed to delete');
    }
}
```

Tampilan delete di views



### 3. CRUD Room

#### Tampilan index room



#### a. Create

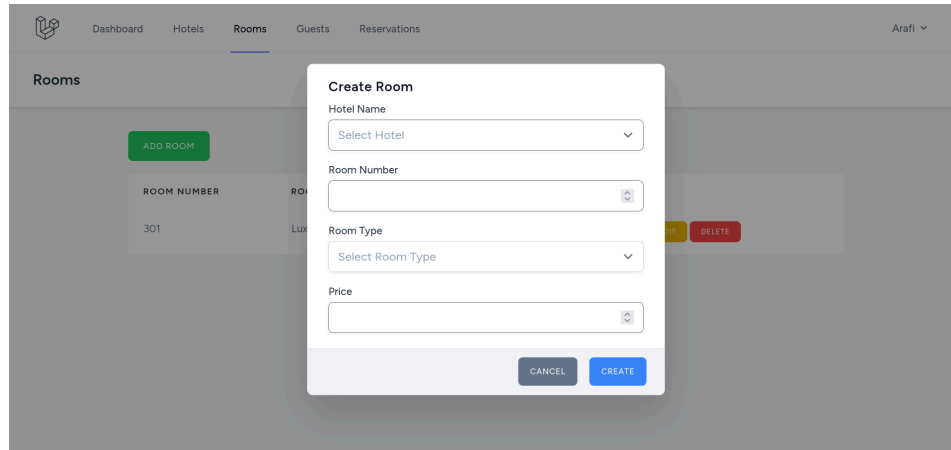
Kode untuk membuat data kamar baru di RoomController.php

```
public function store(Request $request)
{
    $request->validate([
        'hotel_id' => 'required',
        'number' => 'required',
        'type' => 'required',
        'price' => 'required',
    ]);

    $room = Room::create([
        'hotel_id' => $request->hotel_id,
        'number' => $request->number,
        'type' => $request->type,
        'price' => $request->price,
    ]);

    if ($room){
        return redirect()->route('rooms.index')->with('success', 'Room created successfully');
    }else{
        return redirect()->route('rooms.index')->with('error', 'Room failed to create');
    }
}
```

Tampilan Create di views



## b. Read

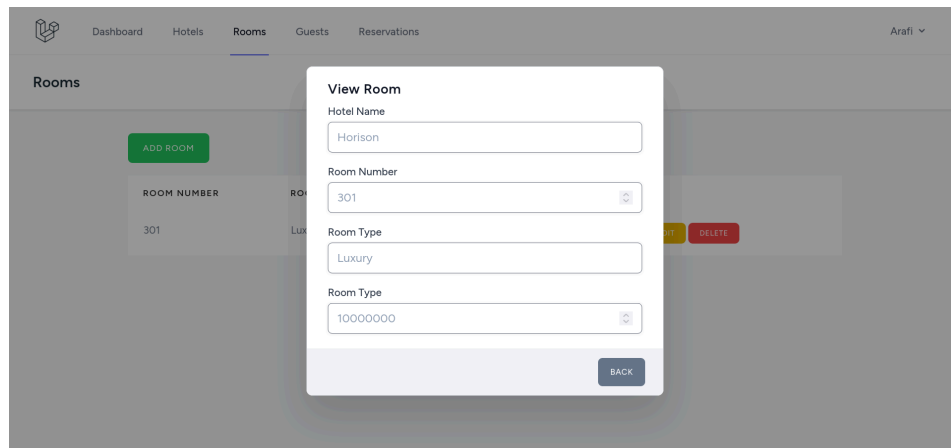
Kode untuk melihat data kamar di RoomController.php

```
public function getDetails(Room $Room)
{
    $hotel = Hotel::where('id', $Room->hotel_id)->first();

    if (!$hotel) {
        return response()->json(['error' => 'Hotel not found for this room.']);
    }

    return response()->json([
        'room' => $Room,
        'hotel' => $hotel
    ]);
}
```

Tampilan Read di views



## c. Update

Kode untuk mengupdate data kamar di RoomController.php

```

public function update(Request $request)
{
    $request->validate([
        'hotel_id' => 'required',
        'number' => 'required',
        'type' => 'required',
        'price' => 'required',
    ]);

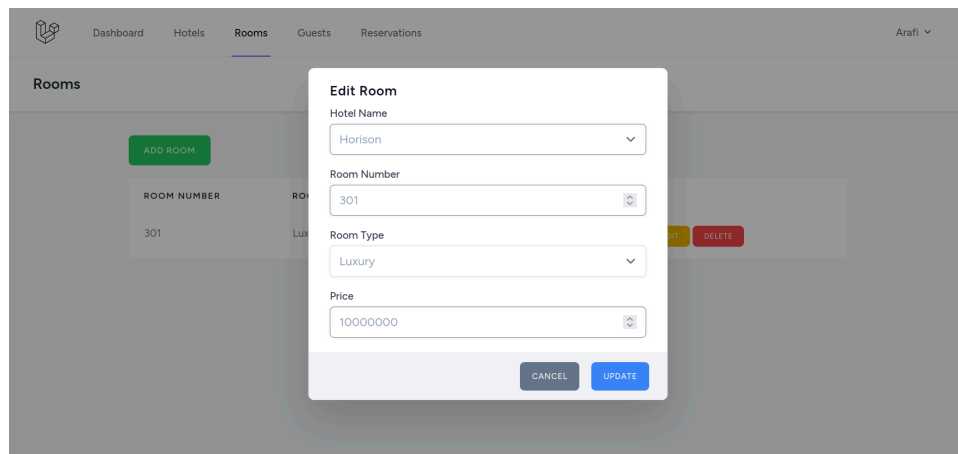
    $search = Room::find($request->id);
    if (!$search){
        return redirect()->route('rooms.index')->with('error', 'Room not found');
    }

    $store = Room::where('id', $request->id)->update([
        'hotel_id' => $request->hotel_id,
        'number' => $request->number,
        'type' => $request->type,
        'price' => $request->price,
    ]);

    if ($store){
        return redirect()->route('rooms.index')->with('success', 'Room updated successfully');
    }else{
        return redirect()->route('rooms.index')->with('error', 'Room failed to update');
    }
}

```

### Tampilan update di views



### d. Delete

#### Kode untuk menghapus data kamar di RoomController.php

```

public function destroy(Request $request)
{
    $find = Room::find($request->id);
    if (!$find){
        return redirect()->route('rooms.index')->with('error', 'Room not found');
    }

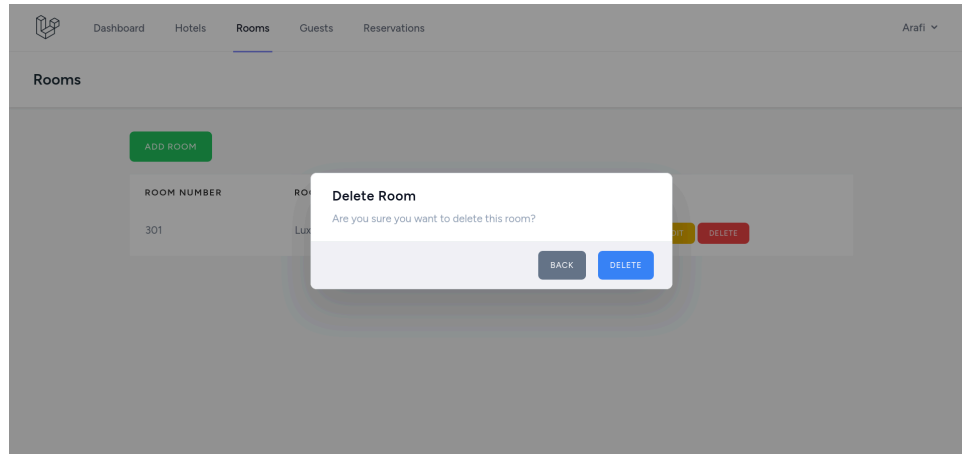
    $destroy = Room::destroy($request->id);

    if ($destroy){
        return redirect()->route('rooms.index')->with('success', 'Room deleted successfully');
    }else{
        return redirect()->route('rooms.index')->with('error', 'Room failed to delete');
    }
}

```

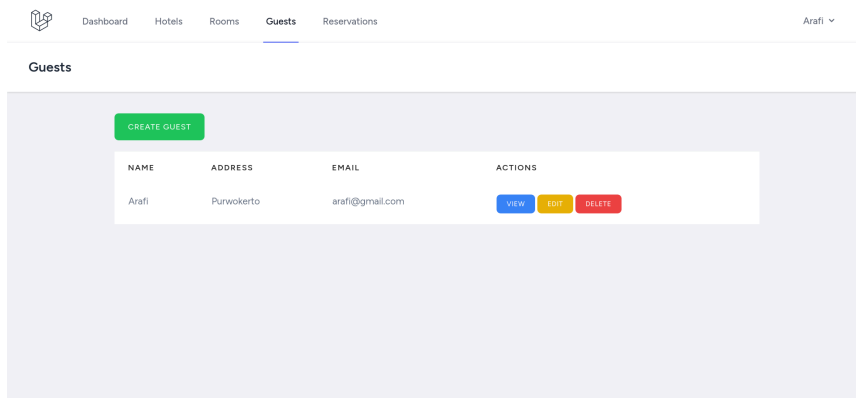
### Tampilan delete di views





#### 4. CRUD Guest

##### Tampilan index guest



##### a. Create

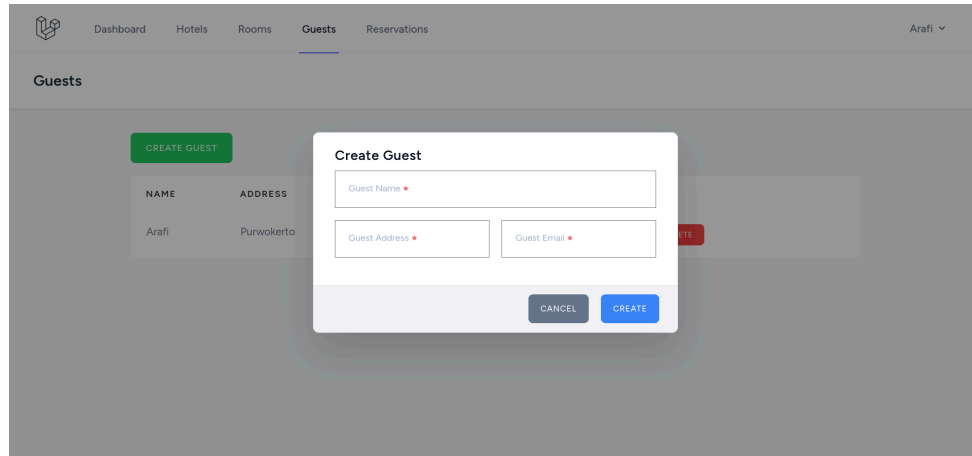
Kode untuk membuat data tamu baru di GuestController.php

```
public function store(Request $request)
{
    $validated = $request->validate([
        'name' => 'required',
        'address' => 'required',
        'email' => 'required',
    ]);

    $store = Guest::create($validated);

    if ($store){
        return redirect()->route('guests.index')->with('success', 'Guest created successfully');
    }else{
        return redirect()->route('guests.index')->with('error', 'Guest failed to create');
    }
}
```

Tampilan Create di views

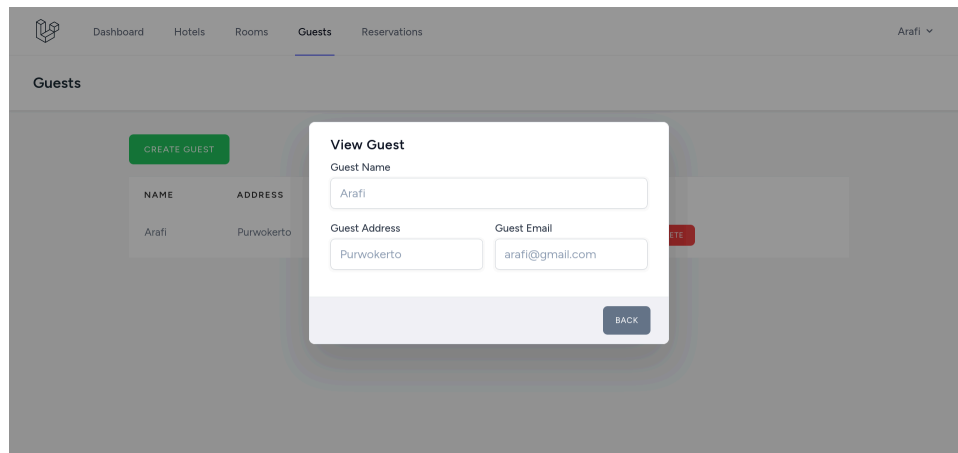


## b. Read

Kode untuk melihat data tamu di GuestController.php

```
public function getDetails(Guest $guest)
{
    return response()->json($guest);
}
```

Tampilan Read di views



## c. Update

Kode untuk mengupdate tamu hotel di GuestController.php

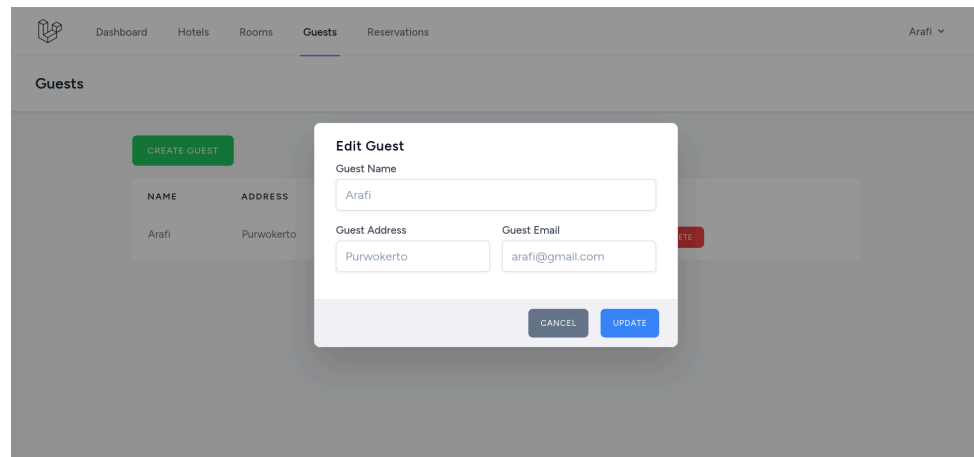
```
public function update(Request $request)
{
    $validated = $request->validate([
        'name' => 'required',
        'address' => 'required',
        'email' => 'required',
    ]);

    $search = Guest::find($request->id);
    if (!$search){
        return redirect()->route('guests.index')->with('error', 'Guest not found');
    }

    $update = Guest::where('id', $request->id)->update($validated);

    if ($update){
        return redirect()->route('guests.index')->with('success', 'Guest updated successfully');
    }else{
        return redirect()->route('guests.index')->with('error', 'Guest failed to update');
    }
}
```

## Tampilan update di views



### d. Delete

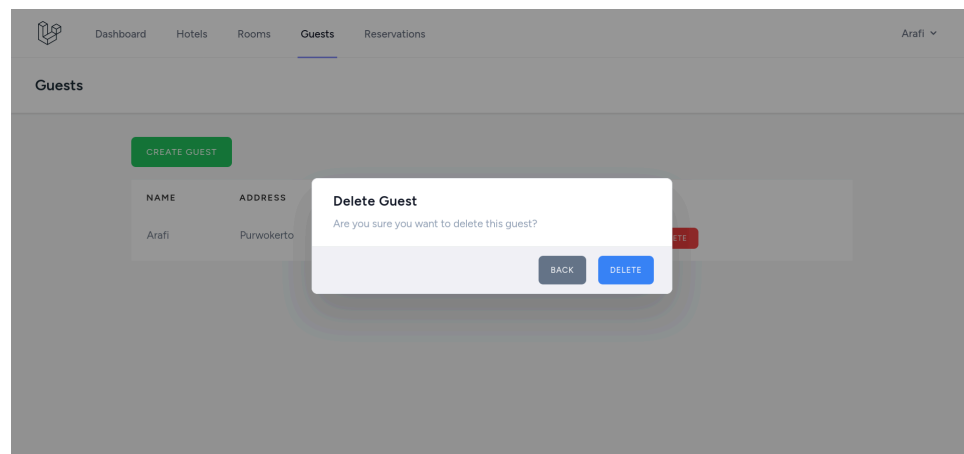
Kode untuk menghapus tamu hotel di GuestController.php

```
public function destroy(Request $request)
{
    $find = Guest::find($request->id);
    if (!$find){
        return redirect()->route('guests.index')->with('error', 'Guest not found');
    }

    $destroy = Guest::destroy($request->id);

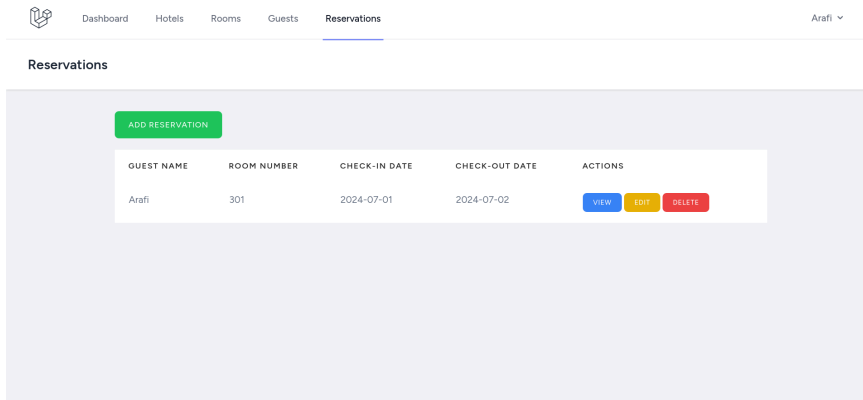
    if ($destroy){
        return redirect()->route('guests.index')->with('success', 'Guest deleted successfully');
    }else{
        return redirect()->route('guests.index')->with('error', 'Guest failed to delete');
    }
}
```

## Tampilan delete di views



## 5. CRUD Reservation

Tampilan index reservation



### a. Create

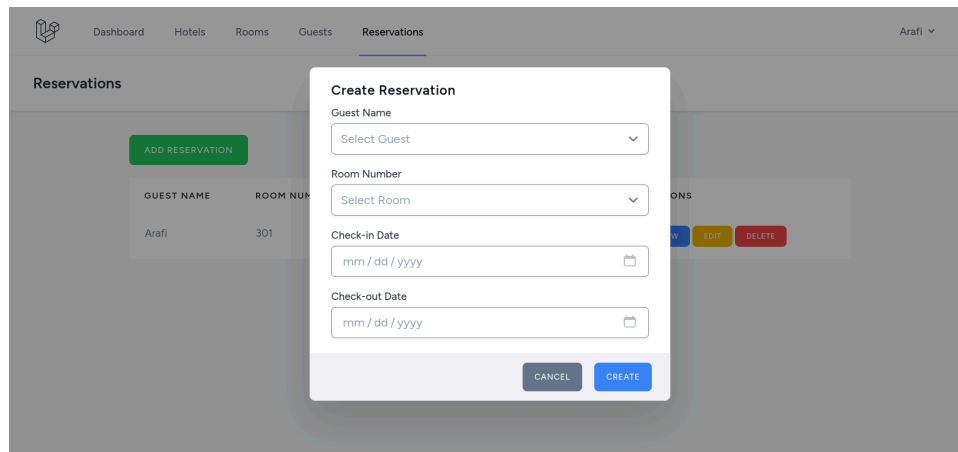
Kode untuk membuat data reservasi baru di ReservationController.php

```
public function store(Request $request)
{
    $request->validate([
        'room_id' => 'required',
        'guest_id' => 'required',
        'check_in' => 'required|date',
        'check_out' => 'required|date',
    ]);

    $reservation = Reservation::create([
        'room_id' => $request->room_id,
        'guest_id' => $request->guest_id,
        'check_in' => $request->check_in,
        'check_out' => $request->check_out,
    ]);

    if ($reservation){
        return redirect()->route('reservations.index')->with('success', 'Reservation created successfully');
    }else{
        return redirect()->route('reservations.index')->with('error', 'Reservation failed to create');
    }
}
```

Tampilan Create di views



### b. Read

Kode untuk melihat data reservasi di ReservationController.php

```

public function getDetails(Reservation $Reservation)
{
    $room = Room::where('id', $Reservation->room_id)->first();
    $guest = Guest::where('id', $Reservation->guest_id)->first();

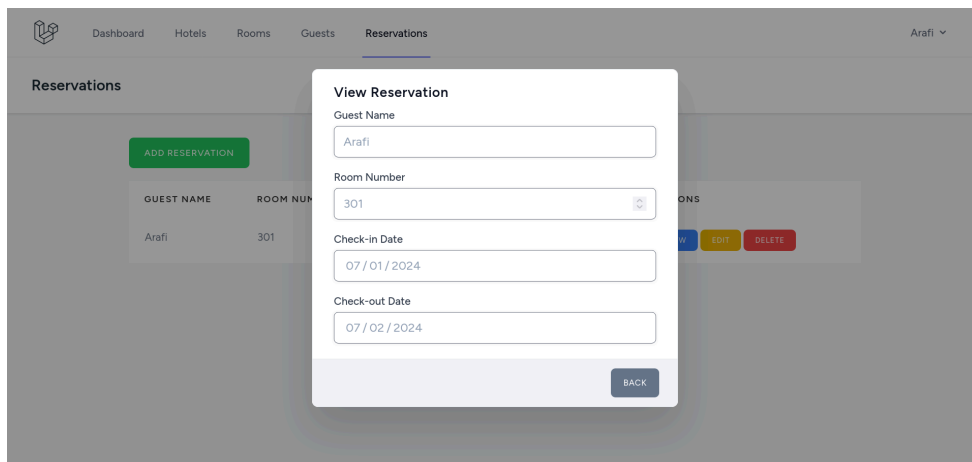
    if (!$room) {
        return response()->json(['error' => 'Room not found for this reservation.']);
    }

    if (!$guest) {
        return response()->json(['error' => 'Guest not found for this reservation.']);
    }

    return response()->json([
        'reservation' => $Reservation,
        'room' => $room,
        'guest' => $guest,
    ]);
}

```

## Tampilan Read di views



## c. Update

Kode untuk mengupdate data reservasi di ReservationController.php

```

public function update(Request $request)
{
    $request->validate([
        'room_id' => 'required',
        'guest_id' => 'required',
        'check_in' => 'required',
        'check_out' => 'required',
    ]);

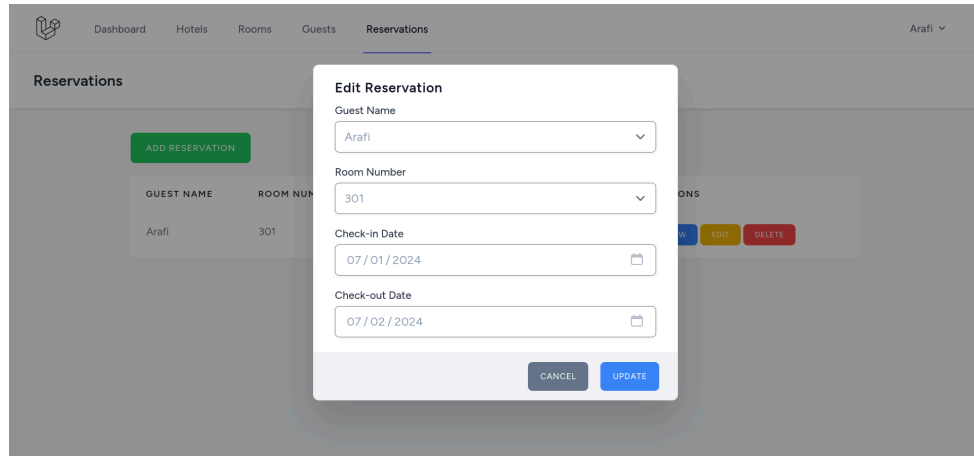
    $search = Reservation::find($request->id);
    if (!$search){
        return redirect()->route('reservations.index')->with('error', 'Reservation not found');
    }

    $store = Reservation::where('id', $request->id)->update([
        'room_id' => $request->room_id,
        'guest_id' => $request->guest_id,
        'check_in' => $request->check_in,
        'check_out' => $request->check_out,
    ]);

    if ($store){
        return redirect()->route('reservations.index')->with('success', 'Reservation updated successfully');
    }else{
        return redirect()->route('reservations.index')->with('error', 'Reservation failed to update');
    }
}

```

## Tampilan update di views



#### d. Delete

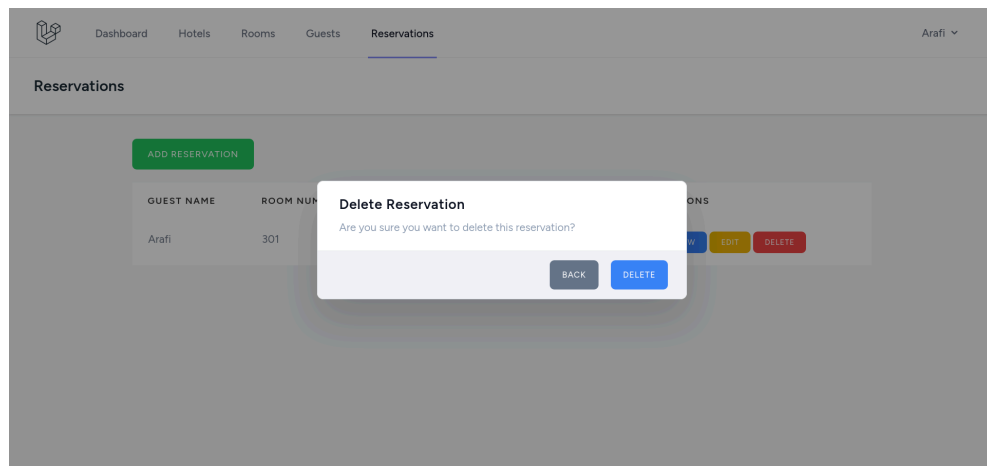
Kode untuk menghapus data reservasi di ReservationController.php

```
public function destroy(Request $request)
{
    $find = Reservation::find($request->id);
    if (!$find){
        return redirect()->route('reservations.index')->with('error', 'Reservation not found');
    }

    $destroy = Reservation::destroy($request->id);

    if ($destroy){
        return redirect()->route('reservations.index')->with('success', 'Reservation deleted successfully');
    }else{
        return redirect()->route('reservations.index')->with('error', 'Reservation failed to delete');
    }
}
```

Tampilan delete di views



Github : <https://github.com/zakijamaliarafi/pemweb-II-uas>