REST API VANIA

PENGEMBANGAN APLIKASI MOBILE LANJUT

ADJANI PRASANA

5210411244

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS & TEKNOLOGI

UNIVERSITAS TEKNOLOGI YOGYAKARTA

YOGYAKARTA

2024

Langkah-langkah Instalasi Framework Vania:

1. **Instalasi Framework Vania**

   Untuk menginstal framework Vania, jalankan perintah berikut pada Command Line di lokasi direktori yang diinginkan:

   ```
   dart pub global activate vania_cli
   ```

2. **Membuat Proyek Baru Vania**

   Untuk membuat proyek baru, gunakan perintah di bawah ini pada direktori pilihan Anda:

   ```
   vania create pamladjani
   ```

3. **Menjalankan Framework Vania**

   Gunakan perintah berikut untuk menjalankan proyek Vania:

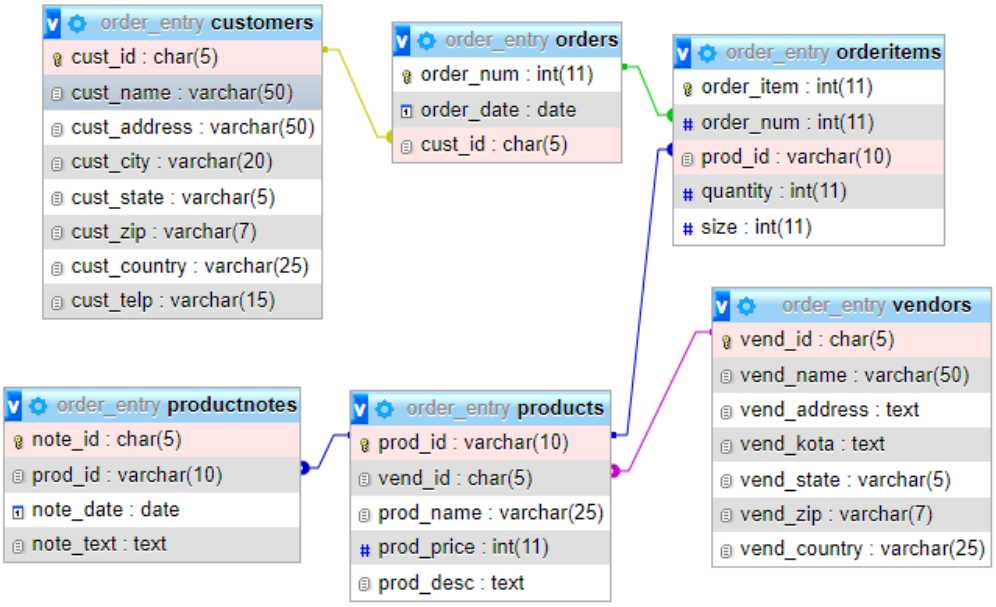
   ```
   vania serve
   ```

Setelah proyek dasar berhasil dibuat, Anda dapat menghapus beberapa komponen bawaan yang dirasa tidak diperlukan.

Konfigurasi Database

Atur konfigurasi database dengan mengedit file .env sesuai kebutuhan. Contoh konfigurasi database adalah sebagai berikut:

```
DB_CONNECTION=mysql
DB_HOST=localhost
DB_PORT=3306
DB_DATABASE=order_entry
DB_USERNAME=project
DB_PASSWORD=pamladjani
DB_SSL_MODE=false
DB_POOL=false
DB_POOL_SIZE=2
```



Persiapan Migrasi Database

Lakukan persiapan migrasi sesuai dengan rancangan relasi tabel database yang Anda butuhkan.

Langkah Migrasi Database:

1. **Buat file migrasi untuk setiap tabel menggunakan perintah berikut:**

```
vania make:migration create_customers_table
vania make:migration create_products_table
vania make:migration create_productnotes_table
vania make:migration create_vendors_table
vania make:migration create_orders_table
vania make:migration create_orderitems_table
vania make:migration create_users_table
vania make:migration create_todos_table
vania make:migration create_personal_token_access_table
```

2. **Buat file migrasi sesuai dengan tabel yang akan digunakan:**
   - **File migrasi untuk *Customers*.**

```
import 'package:vania/vania.dart';

class CreateCustomersTable extends Migration {

  @override
  Future<void> up() async{
   super.up();
```

```
   await createTableNotExists('customers', () {
      char('cust_id', length: 5);
      primary('cust_id');
      string('cust_name', length: 50);
      string('cust_address', length: 50);
      string('cust_city', length: 20);
      string('cust_state', length: 5);
      string('cust_zip', length: 7);
      string('cust_country', length: 25);
      string('cust_telp', length: 15);
      timeStamps();
    });
  }

  @override
  Future<void> down() async {
    super.down();
    await dropIfExists('customers');
  }
}
```

- **File migrasi untuk *Products*.**

```
import 'package:vania/vania.dart';

class CreateProductsTable extends Migration {

  @override
  Future<void> up() async{
   super.up();
   await createTableNotExists('products', () {
      string('prod_id', length: 10);
      primary('prod_id');
      char('vend_id', length: 11);
      foreign('vend_id', 'vendors', 'vend_id', constrained: true, onDelete:
'CASCADE');
      string('prod_name', length: 25);
      bigInt('prod_price', length: 15, defaultValue: 0);
      text('prod_desc');
      timeStamps();
    });
  }

  @override
  Future<void> down() async {
    super.down();
    await dropIfExists('products');
  }
}
```

- **File migrasi untuk *Product Notes*.**

```
import 'package:vania/vania.dart';

class CreateProductnotesTable extends Migration {

  @override
  Future<void> up() async{
   super.up();
   await createTableNotExists('productnotes', () {
      char('note_id', length: 5);
      primary('note_id');
      string('prod_id', length: 10);
      foreign('prod_id', 'products', 'prod_id', constrained: true, onDelete:
'CASCADE');
      date('note_date');
      text('note_text');
      timeStamps();
    });
  }

  @override
  Future<void> down() async {
    super.down();
    await dropIfExists('productnotes');
  }
}
```

- **File migrasi untuk *Vendors*.**

```
import 'package:vania/vania.dart';

class CreateVendorsTable extends Migration {

  @override
  Future<void> up() async{
   super.up();
   await createTableNotExists('vendors', () {
      char('vend_id', length: 5);
      primary('vend_id');
      string('vend_name', length: 50);
      text('vend_address');
      text('vend_kota');
      string('vend_state', length: 5);
      string('vend_zip', length: 7);
      string('vend_country', length: 25);
      timeStamps();
    });
  }

  @override
  Future<void> down() async {
```

```
    super.down();
    await dropIfExists('vendors');
  }
}
```

- **File migrasi untuk *Orders*.**

```
import 'package:vania/vania.dart';

class CreateOrdersTable extends Migration {

  @override
  Future<void> up() async{
   super.up();
   await createTableNotExists('orders', () {
      bigInt('order_num');
      primary('order_num');
      date('order_date');
      char('cust_id', length: 5);
      foreign('cust_id', 'customers', 'cust_id', constrained: true, onDelete:
'CASCADE');
      timeStamps();
    });
  }

  @override
  Future<void> down() async {
    super.down();
    await dropIfExists('orders');
  }
}
```

- **File migrasi untuk *Order Items*.**

```
import 'package:vania/vania.dart';

class CreateOrderitemsTable extends Migration {

  @override
  Future<void> up() async{
   super.up();
   await createTableNotExists('orderitems', () {
      bigInt('order_item', length: 11);
      primary('order_item');
      bigInt('order_num', length: 11);
      foreign('order_num', 'orders', 'order_num', constrained: true, onDelete:
'CASCADE');
      string('prod_id', length: 10);
      foreign('prod_id', 'products', 'prod_id', constrained: true, onDelete:
'CASCADE');
      integer('quantity', defaultValue: 0);
```

```
      integer('size',defaultValue: 0);
    });
  }

  @override
  Future<void> down() async {
    super.down();
    await dropIfExists('orderitems');
  }
}
```

- **File migrasi untuk *Users*.**

```
import 'package:vania/vania.dart';

class CreateUsersTable extends Migration {

  @override
  Future<void> up() async{
   super.up();
   await createTableNotExists('users', () {
      id();
      string('name', length: 100);
      string('email', length: 191);
      string('password', length: 200);
      dateTime('created_at', nullable: true);
      dateTime('updated_at', nullable: true);
      dateTime('deleted_at', nullable: true);
    });
  }

  @override
  Future<void> down() async {
    super.down();
    await dropIfExists('users');
  }
}
```

- **File migrasi untuk *To Do*.**

```
import 'package:vania/vania.dart';

class CreateTodosTable extends Migration {

  @override
  Future<void> up() async{
   super.up();
   await createTableNotExists('todos', () {
      id();
      bigInt('user_id', unsigned: true);
      foreign('user_id', 'users', 'id');
      string('title');
```

```
      text('desciption');
      timeStamps();
    });
  }

  @override
  Future<void> down() async {
    super.down();
    await dropIfExists('todos');
  }
}
```

- **File migrasi untuk *Personal Access Token*.**

```
import 'package:vania/vania.dart';

class CreatePersonalAccessTokensTable extends Migration {

  @override
  Future<void> up() async{
   super.up();
   await createTableNotExists('personal_access_tokens', () {
      id();
      tinyText('name');
      bigInt('tokenable_id');
      string('token');
      timeStamp('last_used_at', nullable: true);
      timeStamp('created_at', nullable: true);
      timeStamp('deleted_at', nullable:true);

      index(ColumnIndex.unique, 'token', ['token']);
    });
  }

  @override
  Future<void> down() async {
    super.down();
    await dropIfExists('personal_access_tokens');
  }
}
```

Membuat Model untuk Tabel

- **Model Customers**

```
import 'package:vania/vania.dart';
import '../utils/generate_id.dart';

class Customers extends Model{
  Customers(){
    super.table('customers');
  }
```

```
    String generateId() {
    const characters =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
    return Utils.generateId(5, characters);
  }
}
```

- **Model Order Items**

```
import 'package:vania/vania.dart';
import '../utils/generate_id.dart';

class OrderItems extends Model {
  OrderItems() {
    super.table('orderitems');
  }
    String generateId() {
    const characters = '0123456789';
    return Utils.generateId(11, characters);
  }
}
```

- **Model Orders**

```
import 'package:vania/vania.dart';
import '../utils/generate_id.dart';

class Order extends Model {
  Order() {
    super.table('orders');
  }
  String generateId() {
    const characters = '0123456789';
    return Utils.generateId(11, characters);
  }
}
```

- **Model Products**

```
import 'package:vania/vania.dart';
import '../utils/generate_id.dart';

class Product extends Model {
  Product() {
    super.table('products');
  }
    String generateId() {
    const characters =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
    return Utils.generateId(5, characters);
  }
}
```

- **Model Product Notes**

```
import 'package:vania/vania.dart';
import '../utils/generate_id.dart';

class ProductNotes extends Model {
  ProductNotes() {
    super.table('productnotes');
  }
  String generateId() {
    const characters =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
    return Utils.generateId(5, characters);
  }
}
```

- **Model To Do**

```
import 'package:vania/vania.dart';
import '../utils/generate_id.dart';

class Todos extends Model {
  Todos() {
    super.table('todos');
  }
  String generateId() {
    const characters =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
    return Utils.generateId(5, characters);
  }
}
```

- **Model Users**

```
import 'package:vania/vania.dart';
import '../utils/generate_id.dart';

class Users extends Model {
  Users() {
    super.table('users');
  }
  String generateId() {
    const characters =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
    return Utils.generateId(5, characters);
  }
}
```

- **Model Vendors**

```
import 'package:vania/vania.dart';
import '../utils/generate_id.dart';

class Vendors extends Model {
```

```
  DateTime createdAt = DateTime.now();
  DateTime updatedAt = DateTime.now();

  Vendors() {
    super.table('vendors');
    createdAt = DateTime.now();
    updatedAt = DateTime.now();
  }

  String generateId() {
    const characters =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
    return Utils.generateId(5, characters);
  }
}
```

- **Utils Id**

```
import 'dart:math';

class Utils {
  static String generateId(int length, String characters) {
    Random random = Random();
    return String.fromCharCodes(Iterable.generate(
      length,
      (_) => characters.codeUnitAt(random.nextInt(characters.length)),
    ));
  }
}
```

Membuat Controller untuk Tabel

Setelah membuat model, buat controller untuk setiap tabel menggunakan perintah:

```
vania make:controller customers_controller
vania make:controller products_controller
vania make:controller productnotes_controller
vania make:controller vendors_controller
vania make:controller orders_controller
vania make:controller todos_controller
vania make:controller auth_controller
```

Controller yang dibuat meliputi:

- ***Customers Controller***

```
import 'package:vania/vania.dart';
import 'package:pamladjani/app/models/customers.dart';

class CustomerController extends Controller {
  Future<Response> listCustomers() async {
```

```
try {
  var queryResults = await Customers()
      .query()
      .join('orders', 'customers.cust_id', '=', 'orders.cust_id')
      .join('orderitems', 'orders.order_num', '=', 'orderitems.order_num')
      .get();

  Map<String, dynamic> customerData = {};
  for (var record in queryResults) {
    String customerId = record['cust_id'];

    if (!customerData.containsKey(customerId)) {
      customerData[customerId] = {
        'cust_id': record['cust_id'],
        'cust_name': record['cust_name'],
        'cust_address': record['cust_address'],
        'cust_city': record['cust_city'],
        'cust_zip': record['cust_zip'],
        'cust_country': record['cust_country'],
        'cust_telp': record['cust_telp'],
        'created_at': record['created_at'],
        'updated_at': record['updated_at'],
        'orders': []
      };
    }

    String orderNumber = record['order_num'].toString();
    var existingOrder = customerData[customerId]['orders'].firstWhere(
        (order) => order['order_num'].toString() == orderNumber,
        orElse: () => null);

    if (existingOrder == null) {
      existingOrder = {
        'order_num': record['order_num'],
        'order_date': record['order_date'],
        'created_at': record['created_at'],
        'updated_at': record['updated_at'],
        'order_items': []
      };

      customerData[customerId]['orders'].add(existingOrder);
    }

    existingOrder['order_items'].add({
      'order_item': record['order_item'],
      'prod_id': record['prod_id'],
      'quantity': record['quantity'],
      'size': record['size'],
```

```
          'created_at': record['created_at'],
          'updated_at': record['updated_at'],
        });
      }

      return Response.json({
        'success': true,
        'message': 'Customers retrieved successfully',
        'data': customerData.values.toList(),
      });
    } catch (error) {
      return Response.json({
        'success': false,
        'message': 'Error retrieving customers',
        'error': error.toString()
      });
    }
  }

  Future<Response> createCustomer() async {
    return Response.json({});
  }

  Future<Response> storeCustomer(Request request) async {
    try {
      var name = request.input('name');
      var address = request.input('address');
      var city = request.input('kota');
      var zipCode = request.input('zip');
      var country = request.input('country');
      var phone = request.input('telp');

      var customerId = Customers().generateId();

      await Customers().query().insert({
        'cust_id': customerId,
        'cust_name': name,
        'cust_address': address,
        'cust_city': city,
        'cust_zip': zipCode,
        'cust_country': country,
        'cust_telp': phone,
        'created_at': DateTime.now().toIso8601String(),
        'updated_at': DateTime.now().toIso8601String(),
      });

      var newCustomer = await Customers().query().where('cust_id', '=',
customerId).first();
```

```
      return Response.json({
        'success': true,
        'message': 'Customer created successfully',
        'data': newCustomer
      });
    } catch (error) {
      return Response.json({
        'success': false,
        'message': 'Error creating customer',
        'error': error.toString()
      });
    }
  }

  Future<Response> showCustomer(String id) async {
    try {
      var customer = await Customers().query().where('cust_id', '=',
id).first();

      if (customer == null) {
        return Response.json({
          'success': false,
          'message': 'Customer not found',
        });
      }
      return Response.json({
        'success': true,
        'message': 'Customer found',
        'data': customer,
      });
    } catch (error) {
      return Response.json({
        'success': false,
        'message': 'Error retrieving customer',
        'error': error.toString()
      });
    }
  }

  Future<Response> editCustomer(int id) async {
    return Response.json({});
  }

  Future<Response> updateCustomer(Request request, String id) async {
    try {
      var name = request.input('name');
      var address = request.input('address');
```

```
      var city = request.input('kota');
      var zipCode = request.input('zip');
      var country = request.input('country');
      var phone = request.input('telp');

      await Customers().query().where('cust_id', '=', id).update({
        'cust_name': name,
        'cust_address': address,
        'cust_city': city,
        'cust_zip': zipCode,
        'cust_country': country,
        'cust_telp': phone,
        'updated_at': DateTime.now().toIso8601String(),
      });

      var updatedCustomer = await Customers().query().where('cust_id', '=',
id).first();

      return Response.json({
        'success': true,
        'message': 'Customer updated successfully',
        'data': updatedCustomer
      });
    } catch (error) {
      return Response.json({
        'success': false,
        'message': 'Error updating customer',
        'error': error.toString()
      });
    }
  }

  Future<Response> deleteCustomer(String id) async {
    try {
      var customer = await Customers().query().where('cust_id', '=',
id).first();

      if (customer == null) {
        return Response.json({
          'success': false,
          'message': 'Customer not found',
        });
      }

      await Customers().query().where('cust_id', '=', id).delete();

      return Response.json({
        'success': true,
```

```
        'message': 'Customer deleted successfully',
      });
    } catch (error) {
      return Response.json({
        'success': false,
        'message': 'Error deleting customer',
        'error': error.toString()
      });
    }
  }
}

final CustomerController customerController = CustomerController();
```

- ***Products Controller***

```
import 'package:vania/vania.dart';
import 'package:pamladjani/app/models/products.dart';
import 'package:pamladjani/app/models/vendors.dart';

class ProductsController extends Controller {
  Future<Response> listProducts() async {
    try {
      var results = await Product()
          .query()
          .join('productnotes', 'products.prod_id', '=',
'productnotes.prod_id')
          .get();
      Map<String, dynamic> productMap = {};
      for (var row in results) {
        String prodId = row['prod_id'];
        if (!productMap.containsKey(prodId)) {
          productMap[prodId] = {
            'prod_id': row['prod_id'],
            'vend_id': row['vend_id'],
            'prod_name': row['prod_name'],
            'prod_price': row['prod_price'],
            'prod_desc': row['prod_desc'],
            'created_at': row['created_at'],
            'updated_at': row['updated_at'],
            'product_notes': []
          };
        } productMap[prodId]['product_notes'].add({
          'note_id': row['note_id'],
          'note_date': row['note_date'],
          'note_text': row['note_text'],
          'created_at': row['created_at'],
          'updated_at': row['updated_at'],
        });
      }
```

```
    return Response.json({
      'success': true,
      'message': 'Products found',
      'data': productMap.values.toList(),
    });
  } catch (e) {
    return Response.json({
      'success': false,
      'message': 'Failed to get products',
      'error': e.toString()
    });
  }
}

Future<Response> create() async {
  return Response.json({});
}

Future<Response> createProduct(Request request) async {
  try {
    var vendorId = request.input('vendor_id');
    var name = request.input('name');
    var price = request.input('price');
    var desc = request.input('desc');

    var isVendorExist =
        await Vendors().query().where('vend_id', '=', vendorId).first();
    if (isVendorExist == null) {
      return Response.json({
        'success': false,
        'message': 'Vendor not found',
      });
    }

    var productId = Product().generateId();
    await Product().query().insert({
      'prod_id': productId,
      'vend_id': isVendorExist['vend_id'],
      'prod_name': name,
      'prod_price': price,
      'prod_desc': desc,
      'created_at': DateTime.now().toIso8601String(),
      'updated_at': DateTime.now().toIso8601String(),
    });

    var product =
        await Product().query().where('prod_id', '=', productId).first();
    return Response.json({
```

```
      'success': true,
      'message': 'Product created successfully',
      'data': product
    });
  } catch (e) {
    return Response.json({
      'success': false,
      'message': 'Store product failed',
      'error': e.toString()
    });
  }
}

Future<Response> showProduct(int id) async {
  return Response.json({});
}

Future<Response> edit(int id) async {
  return Response.json({});
}

Future<Response> editProduct(Request request, String id) async {
  try {
    var product = await Product().query().where('prod_id', '=', id).first();
    if (product == null) {
      return Response.json({
        'success': false,
        'message': 'Product not found',
      });
    }
    var vendorId = request.input('vendor_id');
    if (vendorId != null && vendorId.isNotEmpty) {
      var isVendorExist =
          await Vendors().query().where('vend_id', '=', vendorId).first();
      if (isVendorExist == null) {
        return Response.json({
          'success': false,
          'message': 'Vendor not found',
        });
      }
    }

    var name = request.input('name');
    var price = request.input('price');
    var desc = request.input('desc');

    await Product().query().where('prod_id', '=', id).update({
      'vend_id': vendorId ?? product['vend_id'],
```

```
        'prod_name': name,
        'prod_price': price,
        'prod_desc': desc,
        'updated_at': DateTime.now().toIso8601String(),
      });

      var updatedProduct = await Product().query().where('prod_id', '=',
id).first();
      return Response.json({
        'success': true,
        'message': 'Product updated successfully',
        'data': updatedProduct
      });
    } catch (e) {
      return Response.json({
        'success': false,
        'message': 'Update product failed',
        'error': e.toString()
      });
    }
  }

  Future<Response> deleteProduct(String id) async {
    try {
      var product = await Product().query().where('prod_id', '=', id).first();
      if (product == null) {
        return Response.json({
          'success': false,
          'message': 'Product not found',
        });
      }
      await Product().query().where('prod_id', '=', id).delete();
      return Response.json({
        'success': true,
        'message': 'Product deleted successfully',
      });
    } catch (e) {
      return Response.json({
        'success': false,
        'message': 'Delete product failed',
        'error': e.toString()
      });
    }
  }
}

final ProductsController productsController = ProductsController();
```

- ***Product Notes Controller***

```dart
import 'package:vania/vania.dart';
import 'package:pamladjani/app/models/products.dart';
import 'package:pamladjani/app/models/productnotes.dart';

class ProductnotesController extends Controller {
  Future<Response> listNotes() async {
    try {
      var results = await ProductNotes()
          .query()
          .join('products', 'productnotes.prod_id', '=', 'products.prod_id')
          .select(['productnotes.*', 'products.prod_name']).get();
       return Response.json({
        'success': true,
        'message': 'Product Notes found',
        'data': results,
      });
    } catch (e) {
      return Response.json({
        'success': false,
        'message': 'Failed to get product notes',
        'error': e.toString()
      });
    }
  }

  Future<Response> create() async {
    return Response.json({});
  }

  Future<Response> createNote(Request request) async {
    try {
      var productId = request.input('product_id');
      var isProductExist = await Product().query().where('prod_id', '=',
productId).first();
      if (isProductExist == null) {
        return Response.json({
          'success': false,
          'message': 'Product not found',
        });
      }

      var noteDate = request.input('date');
      var noteText = request.input('text');

      var productNoteId = ProductNotes().generateId();
      await ProductNotes().query().insert({
        'note_id': productNoteId,
        'prod_id': isProductExist['prod_id'],
```

```
        'note_date': noteDate,
        'note_text': noteText,
        'created_at': DateTime.now().toIso8601String(),
        'updated_at': DateTime.now().toIso8601String(),
      });

      var productNote = await ProductNotes()
          .query()
          .where('note_id', '=', productNoteId)
          .first();

      return Response.json({
        'success': true,
        'message': 'Product Note created successfully',
        'data': productNote
      });
    } catch (e) {
      return Response.json({
        'success': false,
        'message': 'Failed to create product note',
        'error': e.toString()
      });
    }
  }

  Future<Response> showNote(int id) async {
    return Response.json({});
  }

  Future<Response> edit(int id) async {
    return Response.json({});
  }

  Future<Response> editNote(Request request, String id) async
{
    try {
      var existingProductNote = await ProductNotes().query().where('note_id',
'=', id).first();
      if (existingProductNote == null) {
        return Response.json({
          'success': false,
          'message': 'Product Note not found',
        });
      }
      var productId = request.input('product_id');
      if (productId != null && productId.isNotEmpty) {
        var isProductExist = await Product().query().where('prod_id', '=',
productId).first();
```

```
        if (isProductExist == null) {
          return Response.json({
            'success': false,
            'message': 'Product not found',
          });
        }
      } else {
        productId = existingProductNote['prod_id'];
      }

      var noteDate = request.input('date');
      var noteText = request.input('text');
      await ProductNotes().query().where('note_id', '=', id).update({
        'prod_id': productId,
        'note_date': noteDate,
        'note_text': noteText,
        'updated_at': DateTime.now().toIso8601String(),
      });

      var productNote = await ProductNotes().query().where('note_id', '=',
id).first();

      return Response.json({
        'success': true,
        'message': 'Product Note updated successfully',
        'data': productNote
      });
    } catch (e) {
      return Response.json({
        'success': false,
        'message': 'Failed to update product note',
        'error': e.toString()
      });
    }
  }

  Future<Response> deleteNote(String id) async {
    try {
      var existingProductNote = await ProductNotes().query().where('note_id',
'=', id).first();
      if (existingProductNote == null) {
        return Response.json({
          'success': false,
          'message': 'Product Note not found',
        });
      }
      await ProductNotes().query().where('note_id', '=', id).delete();
      return Response.json({
```

```
        'success': true,
        'message': 'Product Note deleted successfully',
      });
    } catch (e) {
      return Response.json({
        'success': false,
        'message': 'Failed to delete product note',
        'error': e.toString()
      });
    }
  }
}

final ProductnotesController productnotesController =
ProductnotesController();
```

- ***Vendors Controller***

```
import 'package:vania/vania.dart';
import 'package:pamladjani/app/models/vendors.dart';

class VendorsController extends Controller {
  Future<Response> listVendor() async {
    try {
      var vendors = await Vendors().query().get();
      return Response.json({
        'success': true,
        'message': 'Vendors found',
        'data': vendors,
      });
    } catch (e) {
      return Response.json({
        'success': false,
        'message': 'Failed to get vendors',
        'error': e.toString()
      });
    }
  }

  Future<Response> create() async {
    return Response.json({});
  }

  Future<Response> createVendor(Request request) async {
    var name = request.input('name');
    var address = request.input('address');
    var kota = request.input('kota');
    var state = request.input('state');
    var zip = request.input('zip');
    var country = request.input('country');
```

```
    try {
      var vendId = Vendors().generateId();

      await Vendors().query().insert({
        'vend_id': vendId,
        'vend_name': name,
        'vend_address': address,
        'vend_kota': kota,
        'vend_state': state,
        'vend_zip': zip,
        'vend_country': country,
        'created_at': DateTime.now().toIso8601String(),
        'updated_at': DateTime.now().toIso8601String(),
      });

      var vendor = await Vendors().query().where('vend_id', '=',
vendId).first();
      return Response.json({
        'success': true,
        'message': 'Vendor created successfully',
        'data': vendor
      });
    } catch (e) {
      return Response.json({
        'success': false,
        'message': 'Failed to create vendor',
        'error': e.toString()
      });
    }
  }

  Future<Response> showVendor(String id) async {
    try {
      var vendor = await Vendors().query().where('vend_id', '=', id).first();
      if (vendor == null) {
        return Response.json({
          'success': false,
          'message': 'Vendor not found',
        });
      }
      return Response.json(
        {'success': true, 'message': 'Vendor found', 'data': vendor},
      );
    } catch (e) {
      return Response.json({
        'success': false,
        'message': 'Failed to get vendor',
```

```
        'error': e.toString()
      });
    }
  }

  Future<Response> edit(int id) async {
    return Response.json({});
  }

  Future<Response> editVendor(Request request, String id) async {
    try {
      var vendor = await Vendors().query().where('vend_id', '=', id).first();
      if (vendor == null) {
        return Response.json({
          'success': false,
          'message': 'Vendor not found',
        });
      }

      var name = request.input('name');
      var address = request.input('address');
      var kota = request.input('kota');
      var state = request.input('state');
      var zip = request.input('zip');
      var country = request.input('country');
      await Vendors().query().where('vend_id', '=', id).update({
        'vend_name': name,
        'vend_address': address,
        'vend_kota': kota,
        'vend_state': state,
        'vend_zip': zip,
        'vend_country': country,
        'updated_at': DateTime.now().toIso8601String(),
      });

      var updatedVendor = await Vendors().query().where('vend_id', '=',
id).first();

      return Response.json({
        'success': true,
        'message': 'Vendor updated successfully',
        'data': updatedVendor
      });
    } catch (e) {
      return Response.json({
        'success': false,
        'message': 'Failed to update vendor',
        'error': e.toString()
```

```
      });
    }
  }

  Future<Response> deleteVendor(String id) async {
    try {
      var vendor = await Vendors().query().where('vend_id', '=', id).first();
      if (vendor == null) {
        return Response.json({
          'success': false,
          'message': 'Vendor not found',
        });
      }
      await Vendors().query().where('vend_id', '=', id).delete();
      return Response.json({
        'success': true,
        'message': 'Vendor deleted successfully',
      });
    } catch (e) {
      return Response.json({
        'success': false,
        'message': 'Failed to delete vendor',
        'error': e.toString()
      });
    }
  }
}

final VendorsController vendorsController = VendorsController();
```

- ***Orders Controller***

```
import 'package:vania/vania.dart';
import 'package:pamladjani/app/models/customers.dart';
import 'package:pamladjani/app/models/orderitems.dart';
import 'package:pamladjani/app/models/orders.dart';
import 'package:pamladjani/app/models/products.dart';

class OrdersController extends Controller {

  Future<Response> listOrders() async {
    try {
      var results = await Order()
          .query()
          .join('orderitems', 'orders.order_num', '=', 'orderitems.order_num')
          .join('products', 'orderitems.prod_id', '=', 'products.prod_id')
          .join('customers', 'orders.cust_id', '=', 'customers.cust_id')
          .get();

      Map<String, dynamic> orderMap = {};
```

```
    for (var row in results) {
      int orderNum = row['order_num'];

      if (!orderMap.containsKey(orderNum)) {
        orderMap[orderNum.toString()] = {
          'order_num': row['order_num'],
          'order_date': row['order_date'],
          'customer_id': row['cust_id'],
          'customer_name': row['cust_name'],
          'created_at': row['created_at'],
          'updated_at': row['updated_at'],
          'order_items': []
        };
      } orderMap[orderNum.toString()]['order_items'].add({
          'order_item': row['order_item'],
          'product_id': row['prod_id'],
          'product_name': row['prod_name'],
          'quantity': row['quantity'],
          'size': row['size'],
          'created_at': row['created_at'],
          'updated_at': row['updated_at'],
        });
      }

    return Response.json({
      'success': true,
      'message': 'Orders found',
      'data': orderMap.values.toList(),
    });
  } catch (e) {
    print(e.toString());

    return Response.json({
      'success': false,
      'message': 'Failed to get orders',
      'error': e.toString()
    });

  }
}

Future<Response> create() async {
  return Response.json({});
}

Future<Response> createOrder(Request request) async {
  try {
```

```
var orderDate = request.input('order_date');
var customerId = request.input('customer_id');
var isCustomerExist =
    await Customers().query().where('cust_id', '=', customerId).first();
if (isCustomerExist == null) {
  return Response.json({
    'success': false,
    'message': 'Customer not found',
  });
}

var orderItems = request.input('order_items') as List;
if (orderItems.isEmpty) {
  return Response.json({
    'success': false,
    'message': 'Order items is empty',
  });
}


var orderNum = Order().generateId();
await Order().query().insert({
  'order_num': orderNum,
  'order_date': orderDate,
  'cust_id': customerId,
  'created_at': DateTime.now().toIso8601String(),
  'updated_at': DateTime.now().toIso8601String(),
});

List<Map<String, dynamic>> savedOrderItems = [];

for (var item in orderItems) {
  if (item is Map) {
    var prodId = item['prod_id'];
    var quantity = item['quantity'];
    var size = item['size'];

    if (prodId == null || quantity == null || size == null) {
      return Response.json({
        'success': false,
        'message': 'Order items is invalid',
      });
    }

    var isProductExist =
        await Product().query().where('prod_id', '=', prodId).first();

    if (isProductExist == null) {
```

```
          return Response.json({

            'success': false,
            'message': 'Product not found',
          });
        }

        var orderItemData = {
          'order_item': OrderItems().generateId(),
          'order_num': orderNum,
          'prod_id': isProductExist['prod_id'],
          'quantity': quantity,
          'size': size,
          'created_at': DateTime.now().toIso8601String(),
          'updated_at': DateTime.now().toIso8601String(),
        };

        await OrderItems().query().insert(orderItemData);

        savedOrderItems.add(orderItemData);
      }
    }

    return Response.json({
      'success': true,
      'message': 'Order created successfully',
      'data': {
        'orders':
            await Order().query().where('order_num', '=', orderNum).first(),
        'order_items': savedOrderItems,
      }
    });
  } catch (e) {
    return Response.json({
      'success': false,
      'message': 'Failed to create order',
      'error': e.toString()
    });
  }
}

Future<Response> showOrder(int id) async {
  return Response.json({});
}

Future<Response> edit(int id) async {
  return Response.json({});
}
```

```
  Future<Response> update(Request request, int id) async {
    return Response.json({});
  }

  Future<Response> deleteOrder(int id) async {
    try {
      var order = await Order().query().where('order_num', '=', id).first();
      if (order == null) {
        return Response.json({
          'success': false,
          'message': 'Order not found',
        });
      }

      await OrderItems().query().where('order_num', '=', id).delete();
      await Order().query().where('order_num', '=', id).delete();
      return Response.json({
        'success': true,
        'message': 'Order deleted successfully',
      });
    } catch (e) {
      return Response.json({
        'success': false,
        'message': 'Failed to delete order',
        'error': e.toString()
      });
    }
  }
}

final OrdersController ordersController = OrdersController();
```

- ***Users Controller***

```
import 'package:vania/vania.dart';
import 'package:pamladjani/app/models/users.dart';

class UsersController extends Controller {
  Future<Response> index() async {
    Map? user = Auth().user();

    if (user != null) {
      user.remove('password');
      return Response.json({
        'status': 'success',
        'message': 'Data pengguna berhasil diambil',
        'data': user,
      });
    } else {
```

```
      return Response.json({
        'status': 'error',
        'message': 'Pengguna tidak terautentikasi',
      }, 401);
    }
  }

  Future<Response> updatePassword(Request request) async {
    request.validate({
      'current_password': 'required',
      'password': 'required|min_length:6|confirmed'
    }, {
      'current_password.required': 'Password saat ini wajib diisi',
      'password.required': 'Password baru wajib diisi',
      'password.min_length': 'Password baru harus memiliki minimal 6
karakter',
      'password.confirmed': 'Konfirmasi password tidak cocok',
    });

    String currentPassword = request.string('current_password');

    Map<String, dynamic>? user = Auth().user();

    if (user != null) {
      if (Hash().verify(currentPassword, user['password'])) {
        await Users().query().where ('id', '=', Auth().id()).update({
          'password': Hash().make(request.string('password')),
        });
        return Response.json({
          'status': 'success',
          'message': 'Password berhasil diperbarui',
        });
      } else {
        return Response.json({
          'status': 'error',
          'message': 'Password saat ini tidak cocok',
        }, 401);
      }
    } else {
      return Response.json({
        'status': 'error',
        'message': 'Pengguna tidak ditemukan',
      }, 404);
    }
  }
}

final UsersController usersController = UsersController();
```

- ***Authentication Controller***

```dart
import 'package:vania/vania.dart';
import 'package:pamladjani/app/models/users.dart';

class AuthController extends Controller {
  Future<Response> register(Request request) async {
    request.validate({
      'name': 'required',
      'email': 'required|email',
      'password': 'required|min_length:6|confirmed',
    }, {
      'name.required': 'nama tidak boleh kosong',
      'email.required': 'email tidak boleh kosong',
      'email.email': 'email yang dimasukkan tidak valid',
      'password.required': 'password tidak boleh kosong',
      'password.min_length': 'password harus terdiri dari minimal 6 karakter',
      'password.confirmed': 'konfirmasi password tidak sesuai',
    });

    final name = request.input('name');
    final email = request.input('email');
    var password = request.input('password');

    var user = await Users().query().where('email', '=', email).first();
    if (user != null) {
      return Response.json({
        "message": "user sudah ada",
      },409);
    }

    String userId = Users().generateId();
    password = Hash().make(password);
    await Users().query().insert({
      "id": userId,
      "name": name,
      "email": email,
      "password": password,
      "created_at": DateTime.now().toIso8601String(),
    });
    return Response.json({
      "message": "Berhasil mendaftarkan user"
      },201);
  }

  Future<Response> login(Request request) async {
    request.validate({
      'email': 'required|email',
      'password': 'required',
    }, {
```

```
      'email.required': 'email tidak boleh kosong',
      'email.email': 'email tidak valid',
      'password.required': 'password tidak boleh kosong',
    });

    final email = request.input('email');
    var password = request.input('password').toString();

    var user = await Users().query().where('email', '=', email).first();
    if (user == null) {
      return Response.json({
        "message": "user belum terdaftar",
      }, 409);
    }

    if (!Hash().verify(password, user['password'])) {
      return Response.json({
        "message": "Kata sandi yang anda masukan salah",
      }, 401);
    }

    final token = await Auth()
      .login(user)
      .createToken(expiresIn: Duration(days: 30), withRefreshToken: true);
    return Response.json({
      "message": "Berhasil Login",
      "token": token,
    });
  }
}

final AuthController authController = AuthController();
```

- ***To Do Controller***

```
import 'package:vania/vania.dart';
import 'package:pamladjani/app/models/todos.dart';

class TodosController extends Controller {
  Future<Response> index() async {
    return Response.json({'message': 'Hello World'});
  }

  Future<Response> store(Request request) async {
    request.validate({
      'title': 'required',
      'description': 'required',
    }, {
      'title.required': 'Judul To do wajib diisi',
      'description.required': 'Deskripsi To do wajib diisi',
```

```
    });

    Map<String, dynamic> data = request.all();

    Map<String, dynamic>? user = Auth().user();

    if (user != null) {
      String todoId = Todos().generateId();
      var todos = await Todos().query().create({
        'id': todoId,
        'user_id': Auth().id(),
        'title': data['title'],
        'description': data['description'],
      });

      return Response.json({
        'status': 'success',
        'message': 'To do berhasil dibuat',
        'data': todos,
      }, 201);
    } else {
      return Response.json({
        'status': 'error',
        'message': 'Pengguna tidak terautentikasi',
      }, 401);
    }
  }
}

final TodosController todosController = TodosController();
```

Mengatur Rute

Setelah seluruh controller selesai dibuat, konfigurasi rute agar API dapat diakses sesuai kebutuhan.

```
import 'package:vania/vania.dart';
import 'package:pamladjani/app/http/controllers/customers_controller.dart';
import 'package:pamladjani/app/http/controllers/orders_controller.dart';
import 'package:pamladjani/app/http/controllers/products_controller.dart';
import 'package:pamladjani/app/http/controllers/productnotes_controller.dart';
import 'package:pamladjani/app/http/controllers/vendors_controller.dart';
import 'package:pamladjani/app/http/controllers/users_controller.dart';
import 'package:pamladjani/app/http/controllers/todos_controller.dart';
import 'package:pamladjani/app/http/controllers/auth_controller.dart';
import 'package:pamladjani/app/http/middleware/authenticate.dart';

class ApiRoute implements Route {
  @override
  void register() {
```

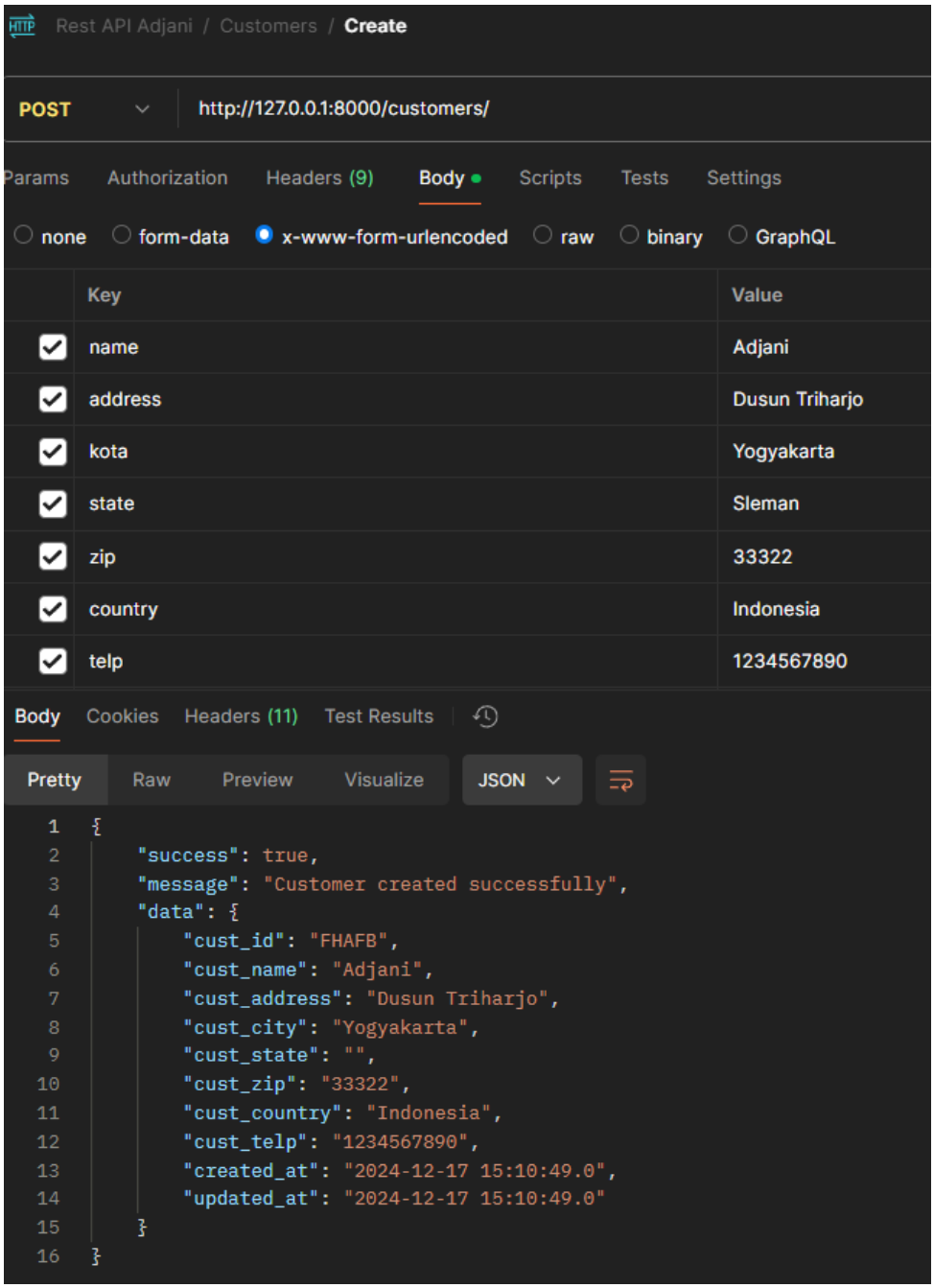
```
Router.group(){
  Router.post('/register', authController.register);
  Router.post('/login', authController.login);
}, prefix: '/auth');

Router.group(() {
  Router.patch('/update-password', usersController.updatePassword);
  Router.get('/', usersController.index);
}, prefix: '/user', middleware: [AuthenticateMiddleware()]);

Router.group(() {
  Router.post('/todo', todosController.store);
}, prefix: '/user', middleware: [AuthenticateMiddleware()]);

Router.group(() {
  Router.get('/', customerController.listCustomers);
  Router.post('/', customerController.storeCustomer);
  Router.get('/{id}', customerController.showCustomer);
  Router.put('/{id}', customerController.updateCustomer);
  Router.delete('/{id}', customerController.deleteCustomer);
}, prefix: '/customers');

Router.group(() {
  Router.get('/', ordersController.listOrders);
  Router.post('/', ordersController.createOrder);
  Router.get('/{id}', ordersController.showOrder);
  Router.delete('/{id}', ordersController.deleteOrder);
}, prefix: '/orders');

Router.group(() {
  Router.get('/', productsController.listProducts);
  Router.post('/', productsController.createProduct);
  Router.get('/{id}', productsController.showProduct);
  Router.put('/{id}', productsController.editProduct);
  Router.delete('/{id}', productsController.deleteProduct);
}, prefix: '/products');

Router.group(() {
  Router.get('/', productnotesController.listNotes);
  Router.post('/', productnotesController.createNote);
  Router.get('/{id}', productnotesController.showNote);
  Router.put('/{id}', productnotesController.editNote);
  Router.delete('/{id}', productnotesController.deleteNote);
}, prefix: '/product-notes');

Router.group(() {
  Router.get('/', vendorsController.listVendor);
  Router.post('/', vendorsController.createVendor);
```

```
      Router.get('/{id}', vendorsController.showVendor);
      Router.put('/{id}', vendorsController.editVendor);
      Router.delete('/{id}', vendorsController.deleteVendor);
    }, prefix: '/vendors');
  }
}
```

Pengujian menggunakan Postman

Gunakan Postman untuk menguji fungsi CRUD pada masing-masing tabel. Hasil yang diharapkan:

- ***Customers***
    - **Create**



- **Read**

Rest API Adjani / Customers / **Read**

GET http://127.0.0.1:8000/customers

Params Authorization Headers (7) Body Scripts Tests Settings

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "success": true,
    "message": "Customers retrieved successfully",
    "data": [
        {
            "cust_id": "FHAFB",
            "cust_name": "Adjani",
            "cust_address": "Dusun Triharjo",
            "cust_city": "Yogyakarta",
            "cust_zip": "33322",
            "cust_country": "Indonesia",
            "cust_telp": "1234567890",
            "created_at": "2024-12-17 15:23:55.0",
            "updated_at": "2024-12-17 15:23:55.0",
            "orders": [
                {
                    "order_num": 11310662623,
                    "order_date": "2025-01-10 00:00:00.0",
                    "created_at": "2024-12-17 15:23:55.0",
                    "updated_at": "2024-12-17 15:23:55.0",
                    "order_items": [
                        {
                            "order_item": 5930028234,
                            "prod_id": "PDTbJ",
                            "quantity": 1,
                            "size": 1,
                            "created_at": "2024-12-17 15:23:55.0",
                            "updated_at": "2024-12-17 15:23:55.0"
                        }
                    ]
                }
            ]
        }
    ]
```

- **Update**

Rest API Adjani / Customers / **Update**

PUT http://127.0.0.1:8000/customers/FHAFB

Params Authorization Headers (9) Body Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

Key	Value
name	Adjani Updated
address	Jl. Magelang Km 27
kota	Yogyakarta
state	Sleman
zip	88888
country	Indonesia
telp	090807060504

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "success": true,
    "message": "Customer updated successfully",
    "data": {
        "cust_id": "FHAFB",
        "cust_name": "Adjani Updated",
        "cust_address": "Jl. Magelang Km 27",
        "cust_city": "Yogyakarta",
        "cust_state": "",
        "cust_zip": "88888",
        "cust_country": "Indonesia",
        "cust_telp": "90807060504",
        "created_at": "2024-12-17 15:10:49.0",
        "updated_at": "2024-12-17 15:28:33.0"
    }
}
```

- **Delete**

Rest API Adjani / Customers / **Delete**

DELETE http://127.0.0.1:8000/customers/FHAFB

Params Authorization Headers (7) Body Scripts Tests

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "success": true,
    "message": "Customer deleted successfully"
}
```


- *Vendors*
  - **Create**

Rest API Adjani / Vendors / **Create**

POST http://127.0.0.1:8000/vendors/

Params Authorization Headers (9) Body Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

Key	Value
name	Honda
address	Jl. Jogja-Solo
kota	Yogyakarta
state	DI Yogya
zip	12345
country	Indonesia
Key	Value

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "success": true,
    "message": "Vendor created successfully",
    "data": {
        "vend_id": "FEHIq",
        "vend_name": "Honda",
        "vend_address": "Jl. Jogja-Solo",
        "vend_kota": "Yogyakarta",
        "vend_state": "DI Yo",
        "vend_zip": "12345",
        "vend_country": "Indonesia",
        "created_at": "2024-12-17 15:14:12.0",
        "updated_at": "2024-12-17 15:14:12.0"
    }
}
```

- **Read**

Rest API Adjani / Vendors / **Read**

GET http://127.0.0.1:8000/vendors/

Params Authorization Headers (7) Body Scripts Tests S

Query Params

	Key
	Key

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "success": true,
    "message": "Vendors found",
    "data": [
        {
            "vend_id": "FEHIq",
            "vend_name": "Honda",
            "vend_address": "Jl. Jogja-Solo",
            "vend_kota": "Yogyakarta",
            "vend_state": "DI Yo",
            "vend_zip": "12345",
            "vend_country": "Indonesia",
            "created_at": "2024-12-17 15:14:12.0",
            "updated_at": "2024-12-17 15:14:12.0"
        }
    ]
}
```

- **Update**

Rest API Adjani / Vendors / **Update**

PUT http://127.0.0.1:8000/vendors/FEHIq

Params Authorization Headers (9) Body Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

Key	Value
name	Honda Update
address	Coming Soon
kota	Coming Soon
state	Coming Soon
zip	11111
country	Indonesia
Key	Value

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "success": true,
    "message": "Vendor updated successfully",
    "data": {
        "vend_id": "FEHIq",
        "vend_name": "Honda Update",
        "vend_address": "Coming Soon",
        "vend_kota": "Coming Soon",
        "vend_state": "Comin",
        "vend_zip": "11111",
        "vend_country": "Indonesia",
        "created_at": "2024-12-17 15:14:12.0",
        "updated_at": "2024-12-17 15:29:42.0"
    }
}
```

- **Delete**

Rest API Adjani / Vendors / **Delete**

DELETE http://127.0.0.1:8000/vendors/FEHIq

Params Authorization Headers (7) Body Scripts Tests

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2     "success": true,
3     "message": "Vendor deleted successfully"
4 }
```

- ***Products***
  - **Create**

Rest API Adjani / Products / **Create**

POST http://127.0.0.1:8000/products/

Params Authorization Headers (9) Body Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

	Key	Value
☑	vendor_id	FEHIq
☑	name	Brio RS
☑	price	219899999
☑	desc	Transmisi Matic CVT, Bensin-1200cc
	Key	Value

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```
1  {
2      "success": true,
3      "message": "Product created successfully",
4      "data": {
5          "prod_id": "PDTbJ",
6          "vend_id": "FEHIq",
7          "prod_name": "Brio RS",
8          "prod_price": 219899999,
9          "prod_desc": "Transmisi Matic CVT, Bensin-1200cc",
10         "created_at": "2024-12-17 15:16:13.0",
11         "updated_at": "2024-12-17 15:16:13.0"
12     }
13 }
```

- **Read**

Rest API Adjani / Products / **Read**

GET http://127.0.0.1:8000/products/

Params Authorization Headers (7) Body Scripts Tests Settings

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (11) Test Results — 200 OK · 9 ms · 1000 B

Pretty Raw Preview Visualize JSON


```
{
    "success": true,
    "message": "Products found",
    "data": [
        {
            "prod_id": "PDTbJ",
            "vend_id": "FEHIq",
            "prod_name": "Brio RS",
            "prod_price": 219899999,
            "prod_desc": "Transmisi Matic CVT, Bensin-1200cc",
            "created_at": "2024-12-17 15:22:30.0",
            "updated_at": "2024-12-17 15:22:30.0",
            "product_notes": [
                {
                    "note_id": "rs8Tg",
                    "note_date": "2025-01-05 00:00:00.0",
                    "note_text": "Mobil dengan fitur adaptive cruise control, LIDAR, dan Honda Assist System yang dapat meningkatkan pengalaman berkemudi",
                    "created_at": "2024-12-17 15:22:30.0",
                    "updated_at": "2024-12-17 15:22:30.0"
                }
            ]
        }
    ]
}
```


- **Update**

Rest API Adjani / Products / **Update**

PUT http://127.0.0.1:8000/products/PDTbJ

Params Authorization Headers (9) Body Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

Key	Value
vendor_id	FEHIq
name	Civic RS
price	1010100000
desc	Mobil Ngebut
Key	Value

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "success": true,
    "message": "Product updated successfully",
    "data": {
        "prod_id": "PDTbJ",
        "vend_id": "FEHIq",
        "prod_name": "Civic RS",
        "prod_price": 1010100000,
        "prod_desc": "Mobil Ngebut",
        "created_at": "2024-12-17 15:16:13.0",
        "updated_at": "2024-12-17 15:30:46.0"
    }
}
```

- **Delete**

Rest API Adjani / Products / **Delete**

DELETE http://127.0.0.1:8000/products/PDTbJ

Params Authorization Headers (7) Body Scripts Tests

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2     "success": true,
3     "message": "Product deleted successfully"
4 }
```

- ***Product Notes***
  - **Create**

Rest API Adjani / Product Notes / **Create**

POST http://127.0.0.1:8000/product-notes/

Params Authorization Headers (9) Body Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

Key	Value	Description
product_id	PDTbJ	
date	2025-01-05	
text	Mobil dengan fitur adaptive cruise control, LIDAR, dan Honda Assist...	
Key	Value	Description

Body Cookies Headers (11) Test Results 200 OK 25 ms 811 B

Pretty Raw Preview Visualize JSON

```
1 {
2     "success": true,
3     "message": "Product Note created successfully",
4     "data": {
5         "note_id": "rs8Tg",
6         "prod_id": "PDTbJ",
7         "note_date": "2025-01-05 00:00:00.0",
8         "note_text": "Mobil dengan fitur adaptive cruise control, LIDAR, dan Honda Assist System yang dapat meningkatkan pengalaman berkemudi",
9         "created_at": "2024-12-17 15:22:30.0",
10        "updated_at": "2024-12-17 15:22:30.0"
11    }
12 }
```

- **Read**

Rest API Adjani / Product Notes / **Read**

GET http://127.0.0.1:8000/product-notes/

Params Authorization Headers (7) Body Scripts Tests Settings

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (11) Test Results 200 OK 8 ms 821 B

Pretty Raw Preview Visualize JSON


```
{
    "success": true,
    "message": "Product Notes found",
    "data": [
        {
            "note_id": "rs8Tg",
            "prod_id": "PDTbJ",
            "note_date": "2025-01-05 00:00:00.0",
            "note_text": "Mobil dengan fitur adaptive cruise control, LIDAR, dan Honda Assist System yang dapat meningkatkan pengalaman berkemudi",
            "created_at": "2024-12-17 15:22:30.0",
            "updated_at": "2024-12-17 15:22:30.0",
            "prod_name": "Brio RS"
        }
    ]
}
```


- **Update**

Rest API Adjani / Product Notes / **Update**

PUT http://127.0.0.1:8000/product-notes/rs8Tg

Params Authorization Headers (9) Body Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

	Key	Value
☑	product_id	PDTbJ
☑	date	2025-01-08
☑	text	Super ngebut
	Key	Value

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "success": true,
    "message": "Product Note updated successfully",
    "data": {
        "note_id": "rs8Tg",
        "prod_id": "PDTbJ",
        "note_date": "2025-01-08 00:00:00.0",
        "note_text": "Super ngebut",
        "created_at": "2024-12-17 15:22:30.0",
        "updated_at": "2024-12-17 15:31:51.0"
    }
}
```

- **Delete**

Rest API Adjani / Product Notes / **Delete**

DELETE http://127.0.0.1:8000/product-notes/rs8Tg

Params Authorization Headers (7) Body Scripts Tests

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "success": true,
    "message": "Product Note deleted successfully"
}
```


- *Orders*
  - **Create**

Rest API Adjani / Orders / **Create**

POST http://127.0.0.1:8000/orders/

Params Authorization Headers (9) Body Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL


```
{
    "order_date": "2025-01-10",
    "customer_id": "FHAFB",
    "order_items": [
        {
            "prod_id": "PDTbJ",
            "quantity": 1,
            "size": 1
        }
    ]
}
```


Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON


```
    "success": true,
    "message": "Order created successfully",
    "data": {
        "orders": {
            "order_num": 11310662623,
            "order_date": "2025-01-10 00:00:00.0",
            "cust_id": "FHAFB",
            "created_at": "2024-12-17 15:23:55.0",
            "updated_at": "2024-12-17 15:23:55.0"
        },
        "order_items": [
            {
                "order_item": "05930028234",
                "order_num": "11310662623",
                "prod_id": "PDTbJ",
                "quantity": 1,
                "size": 1,
                "created_at": "2024-12-17T15:23:55.241864",
                "updated_at": "2024-12-17T15:23:55.241864"
```

- **Read**

Rest API Adjani / Orders / **Read**

GET http://127.0.0.1:8000/orders/

Params Authorization Headers (7) Body Scripts Tests Settings

Query Params

	Key	Value

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "success": true,
    "message": "Orders found",
    "data": [
        {
            "order_num": 11310662623,
            "order_date": "2025-01-10 00:00:00.0",
            "customer_id": "FHAFB",
            "customer_name": "Adjani",
            "created_at": "2024-12-17 15:10:49.0",
            "updated_at": "2024-12-17 15:10:49.0",
            "order_items": [
                {
                    "order_item": 5930028234,
                    "product_id": "PDTbJ",
                    "product_name": "Brio RS",
                    "quantity": 1,
                    "size": 1,
                    "created_at": "2024-12-17 15:10:49.0",
                    "updated_at": "2024-12-17 15:10:49.0"
                }
            ]
        }
    ]
}
```

- **Delete**

Rest API Adjani / Orders / **Delete**

DELETE http://127.0.0.1:8000/orders/11310662623

Params Authorization Headers (7) Body Scripts Tests

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2     "success": true,
3     "message": "Order deleted successfully"
4 }
```

- ***Update*: untuk transaksi tidak tersedia.**

- **User**
  - **Register**

Berhasil

Rest API Adjani / Users / **Register**

POST http://127.0.0.1:8000/auth/register

Params Authorization Headers (9) Body Scripts Tests

none form-data x-www-form-urlencoded raw binary

```
1 {
2     "name": "Adjani",
3     "email": "adjaniprasana@gmail.com",
4     "password": "adjani24",
5     "password_confirmation": "adjani24"
6 }
```

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2     "message": "Berhasil mendaftarkan user"
3 }
```

Tidak berhasil karena email atau password

Rest API Adjani / Users / **Register**

POST http://127.0.0.1:8000/auth/register

Params Authorization Headers (9) **Body** Scripts Tests Settings

none form-data x-www-form-urlencoded **raw** binary GraphQL JSON

```
1 {
2     "name": "Adjani",
3     "email": "adjaniprasanagmail.com",
4     "password": "adjani24",
5     "password_confirmation": "testadmin"
6 }
```

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2     "email": "email yang dimasukkan tidak valid",
3     "password": "konfirmasi password tidak sesuai"
4 }
```

- **Login**

Berhasil

Rest API Adjani / Users / **Login** Save Share

POST http://127.0.0.1:8000/auth/login Send

Params Authorization Headers (9) **Body** Scripts Tests Settings Cookies

none form-data **x-www-form-urlencoded** raw binary GraphQL

Key	Value	Description
email	adjaniprasana@gmail.com	
password	adjani24	
Key	Value	Description

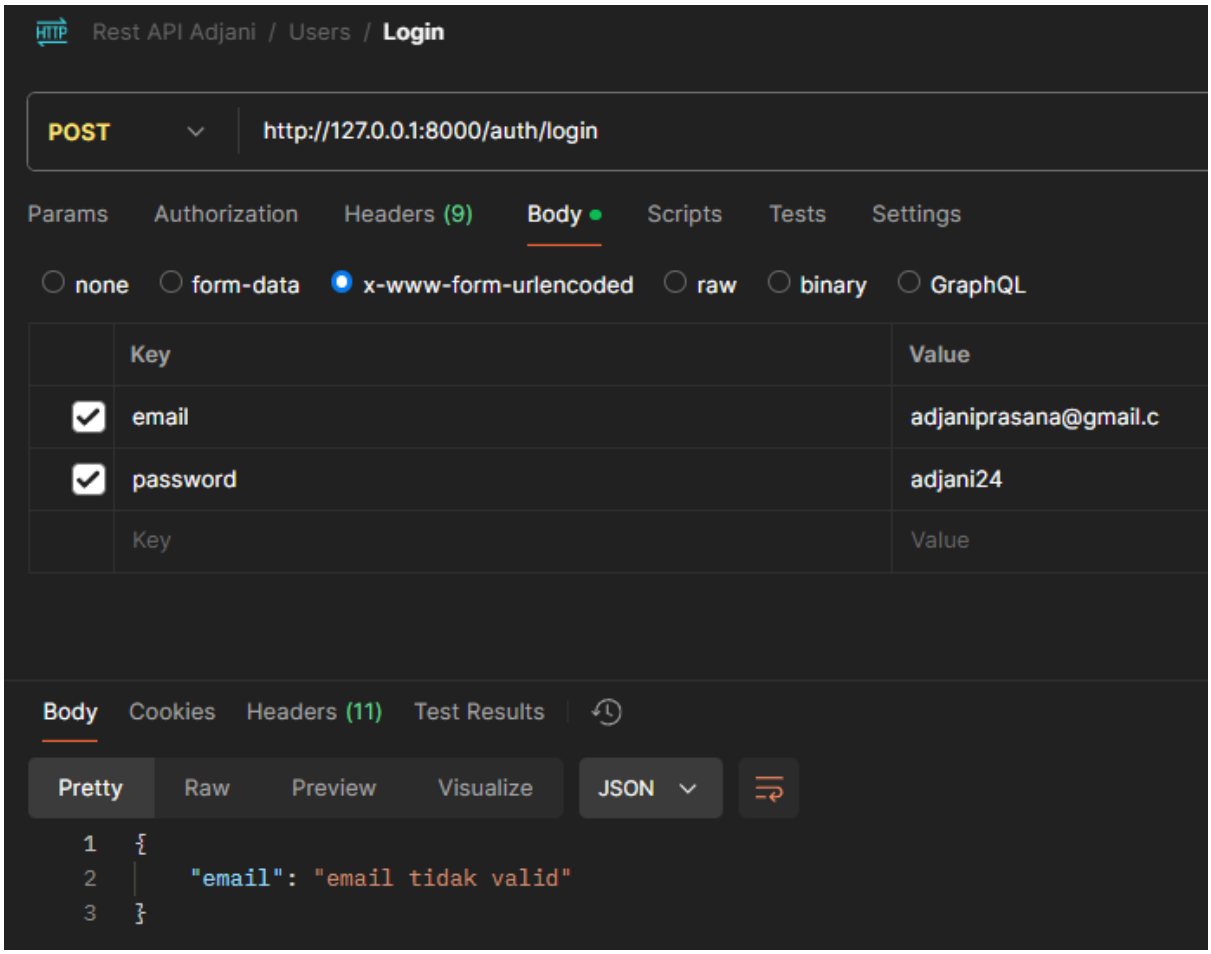
Bulk Edit

Body Cookies Headers (11) Test Results 200 OK 74 ms 1.24 KB Save Response

Pretty Raw Preview Visualize JSON


```
1 {
2     "message": "Berhasil Login",
3     "token": {
4         "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
              eyJ1c2VyIjoie1wiaWRcIjoyLFwibmFtZVwiOlwiQWRqYW5pXCIsXCJlbWFpbFwiOlwiYWRqYW5pcHJhc2FuYUBnbWFpbC5jb21cIixcIn8hc3N3b3JkXCI6XCJockw8N2gvTGJNb2FRbnZ5N3lEaUFpOWk4O
              XN6WnRiSmRDYWFSdFpvQmlGWGtBWXZuTSttcklaNUVnbjhtWGNtRkNkeGFvNkVKcC9acGE5MTQwN1hzL3JoZEE9PVwiLFwiY3JlYXRlZF9hdFwiOlwiMjAyNC0xMi0xNyAxNjowNzoyNS4wXCIsXCJ1cGRhdG
              VkX2F0XCI6bnVsbCxcImRlbGV0ZWRfYXRcIjpudWxsfSIsInR5cGUiOiJhY2Nlc3NfdG9rZW4iLCJpZCI6MiwiaWF0IjoxNzM0NDI2NTAwLCJleHAiOjE3MzcwMTg1MDB9.
              4egpnOyeswvucKnAm1CK-aSC1QzMTIEVJlI7E0FN9KI",
5         "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwidHlwZSI6InJlZnJlc2hfdG9rZW4iLCJpYXQiOjE3MzQ0MjY1MDAsImV4cCI6MTczNzAxODUwMH0.
              ilaVAK8uNZkbwLMpMlv8LF8PBJJ9s100qlmjyd0kn_4",
6         "expires_in": "2025-01-16T16:08:20.720190"
7     }
8 }
```

Email salah


Rest API Adjani / Users / Login
POST
http://127.0.0.1:8000/auth/login
Params
Authorization
Headers (9)
Body
Scripts
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
Key
Value
email
adjaniprasana@gmail.c
password
adjani24
Key
Value
Body
Cookies
Headers (11)
Test Results
Pretty
Raw
Preview
Visualize
JSON
{
"email": "email tidak valid"
}

Rest API Adjani / Users / **Login**

POST http://127.0.0.1:8000/auth/login

Params Authorization Headers (9) Body Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

	Key	Value
☑	email	adjaniprasana24@gmail.com
☑	password	adjani24
	Key	Value

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2     "message": "user belum terdaftar"
3 }
```

Password salah

Rest API Adjani / Users / **Login**

POST http://127.0.0.1:8000/auth/login

Params Authorization Headers (9) Body Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

	Key	Value
☑	email	adjaniprasana@gmail.com
☑	password	adjani
	Key	Value

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2     "message": "Kata sandi yang anda masukan salah"
3 }
```

Cek Akun berdasarkan token

Rest API Adjani / Users / **Check User By Token**

GET http://127.0.0.1:8000/user/

Params | Authorization | Headers (8) | Body | Scripts | Tests | Settings

Auth Type: Bearer Token

Token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey. ...

The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.

Body | Cookies | Headers (11) | Test Results

Pretty | Raw | Preview | Visualize | JSON


```
{
    "status": "success",
    "message": "Data pengguna berhasil diambil",
    "data": {
        "id": 2,
        "name": "Adjani",
        "email": "adjaniprasana@gmail.com",
        "created_at": "2024-12-17 16:07:25.0",
        "updated_at": null,
        "deleted_at": null
    }
}
```


Reset Password

Rest API Adjani / Users / **Update Password User**

PATCH http://127.0.0.1:8000/user/update-password

Params | Authorization | Headers (10) | Body | Scripts | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL

Key	Value
current_password	adjani24
password	update24
password_confirmation	update24
access_token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoie1wiaWRcIjoyL....
Key	Value

Body | Cookies | Headers (11) | Test Results

Pretty | Raw | Preview | Visualize | JSON


```
{
    "status": "success",
    "message": "Password berhasil diperbarui"
}
```

- **To Do**
  - **Create**

Rest API Adjani / Users / **To Do**

POST http://127.0.0.1:8000/user/todo

Params | Authorization | Headers (10) | Body | Scripts | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL

Key	Value
id	2
title	Testing Adjani
description	Planning Schedule Next Week
Key	Value

Body | Cookies | Headers (11) | Test Results

Pretty | Raw | Preview | Visualize | JSON


```
{
    "status": "success",
    "message": "To do berhasil dibuat",
    "data": {
        "id": 2,
        "user_id": 2,
        "title": "Testing Adjani",
        "description": "Planning Schedule Next Week",
        "created_at": null,
        "updated_at": null
    }
}
```