

Nama : Ibnu Abdul Aziz Alqalibi

NPM : 5200411391

Buat project baru

```
Built vania_cli:vania_cli.  
Installed executable vania.  
Activated vania_cli 1.3.0.  
  
C:\WINDOWS\system32>f:  
  
F:\>cd F:\matkul\mws\tugas\backend  
  
F:\matkul\mws\tugas\backend>vania create store
```

```
+ vania 0.6.2  
  vm_service 14.2.1 (15.0.0 available)  
  web 0.5.1 (1.1.0 available)  
  web_socket_channel 2.4.5 (3.0.1 available)  
Changed 28 dependencies!  
29 packages have newer versions incompatible with dependency constraints.  
Try `dart pub outdated` for more information.  
  
SUCCESS | All done! Build something amazing  
You can find general documentation for Vania at: https://vdart.dev/docs/intro/  
  
In order to run your application, type:  
$ cd store  
$ Vania serve  
  
F:\matkul\mws\tugas\backend>_
```

Buat Migrations nya

1. Table customers

```
class CreateCustomersTable extends Migration {  
  @override  
  Future<void> up() async {  
    super.up();  
    await createTableIfNotExists('customers', () {  
      char('cust_id', length: 5, unique: true, nullable: false);  
      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);  
    });  
  }  
}
```

2. Table orders

```
class CreateOrders extends Migration {
    @override
    Future<void> up() async {
        super.up();
        await createTableIfNotExists('orders', () {
            integer('order_num', length: 11, nullable: false, unique: true);
            date('order_date');
            char('cust_id', length: 5);
        });
    }
}
```

3. Table orderitems

```
class CreateOrderitems extends Migration {
    @override
    Future<void> up() async {
        super.up();
        await createTableIfNotExists('orderitems', () {
            integer('order_item', length: 11, unique: true, nullable: false);
            integer('order_num', length: 11);
            string('prod_id', length: 10);
            integer('quantity', length: 11);
            integer('size', length: 11);
        });
    }
}
```

4. Table vendors

```
class CreateVendors extends Migration {
    @override
    Future<void> up() async {
        super.up();
        await createTableIfNotExists('vendors', () {
            char('vend_id', length: 5, unique: true, nullable: false);
            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);
        });
    }
}
```

5. Table products

```
class CreateProducts extends Migration {
    @override
    Future<void> up() async {
        super.up();
        await createTableIfNotExists('products', () {
            string("prod_id", length: 10, unique: true, nullable: false);
            char("vend_id", length: 5);
            string("prod_name", length: 25);
            integer("prod_price", length: 11);
            text("prod_desc");
        });
    }
}
```

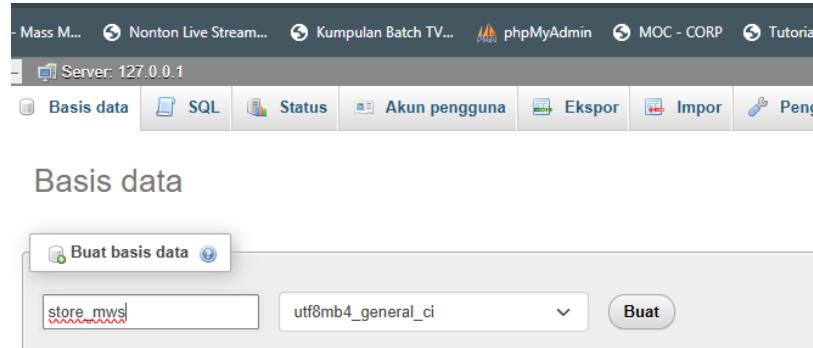
6. Table productnotes

```
class CreateProductnotes extends Migration {
    @override
    Future<void> up() async {
        super.up();
        await createTableIfNotExists('productnotes', () {
            char("note_id", length: 5, unique: true, nullable: false);
            string("prod_id", length: 10);
            date("note_date");
            text('note_text');
        });
    }
}
```

7. Ubah konfigurasi .env

```
# database driver mysql or postgresql or pgsql
DB_CONNECTION=mysql
DB_HOST=localhost
DB_PORT=3306
DB_DATABASE=store_mws
DB_USERNAME=ibnu
DB_PASSWORD=123
DB_SSL_MODE=false
DB_POOL=false
DB_POOL_SIZE=2
```

8. Create database



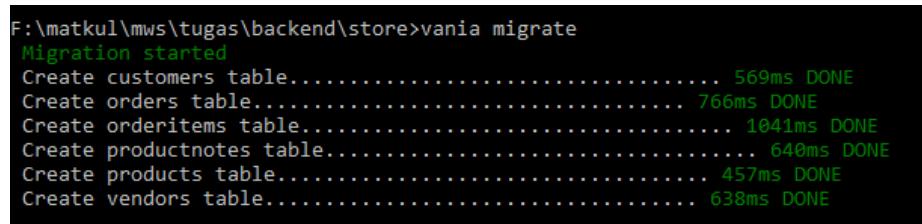
Basis data

Buat basis data

store_mws utf8mb4_general_ci

Buat

9. Migrate



```
F:\matkul\mws\tugas\backend\store>vania migrate
Migration started
Create customers table..... 569ms DONE
Create orders table..... 766ms DONE
Create orderitems table..... 1041ms DONE
Create productnotes table..... 640ms DONE
Create products table..... 457ms DONE
Create vendors table..... 638ms DONE
```

Struktur SQL Cari Kueri Ekspor Impor Operasi Hak Akses Routine Event Trigger

Filters

Mengandung kata:

Tabel	Aksi	Baris	Jenis	Penyortiran	Ukuran	Beban
customers	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_general_ci	16.0 KB	-
orderitems	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_general_ci	16.0 KB	-
orders	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_general_ci	16.0 KB	-
productnotes	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_general_ci	16.0 KB	-
products	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_general_ci	16.0 KB	-
vendors	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_general_ci	16.0 KB	-
6 tabel	Jumlah	0	InnoDB	utf8mb4_general_ci	96.0 KB	0 B

Buat models

```
F:\matkul\mws\tugas\backend\store> vania make:model Customer
What should the table be named?
> customers
INFO Model [F:\matkul\mws\tugas\backend\store\lib\app\models\customer.dart] created successfully.

F:\matkul\mws\tugas\backend\store> vania make:model Order
What should the table be named?
> orders
INFO Model [F:\matkul\mws\tugas\backend\store\lib\app\models\order.dart] created successfully.

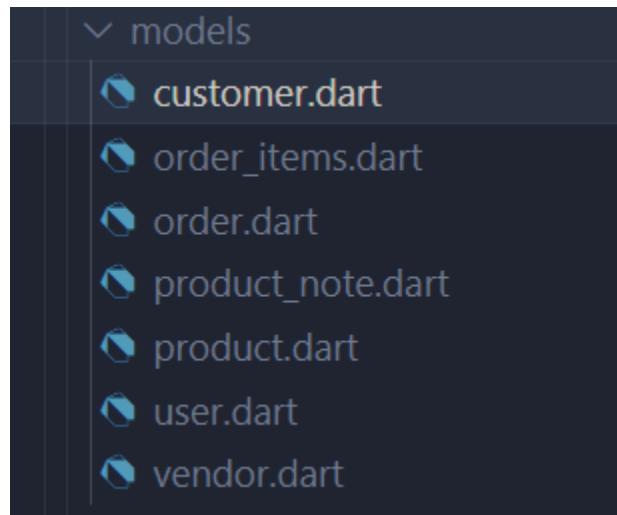
F:\matkul\mws\tugas\backend\store> vania make:model OrderItems
What should the table be named?
> orderitems
INFO Model [F:\matkul\mws\tugas\backend\store\lib\app\models\order_items.dart] created successfully.

F:\matkul\mws\tugas\backend\store> vania make:model ProductNote
What should the table be named?
> productnotes
INFO Model [F:\matkul\mws\tugas\backend\store\lib\app\models\product_note.dart] created successfully.

F:\matkul\mws\tugas\backend\store> vania make:model Product
What should the table be named?
> products
INFO Model [F:\matkul\mws\tugas\backend\store\lib\app\models\product.dart] created successfully.

F:\matkul\mws\tugas\backend\store> vania make:model Vendor
What should the table be named?
> vendors
INFO Model [F:\matkul\mws\tugas\backend\store\lib\app\models\vendor.dart] created successfully.
```

Hasil:



Buat Controller

1. Customer_controller

Index:

```
Future<Response> index() async {
  try {
    final custData = await Customer().query().get();
    return Response.json({
      "success": true,
      "message": "Berhasil menampilkan data",
      "data": custData,
    }, 200);
  } catch (e) {
    return Response.json({
      "success": false,
      "message": e.toString(),
      "data": null,
    }, 500);
  }
}
```

Store:

```
Future<Response> store(Request req) async {
  try {
    req.validate({
      'cust_id': 'required|string|max_length:5',
      'cust_name': 'required|string|max_length:100',
      'cust_address': 'required|string|max_length:50',
      'cust_city': 'required|string|max_length:20',
      'cust_state': 'required|string|max_length:5',
      'cust_zip': 'required|string|max_length:7',
      'cust_country': 'required|string|max_length:25',
      'cust_telp': 'required|string|max_length:15',
    });

    final requestData = req.input();
    final existingProduct = await Customer()
      .query()
      .where('cust_id', '=', requestData['cust_id'])
      .first();

    if (existingProduct != null) {
      return Response.json({
        'message': 'Customer dengan id ini sudah ada.',
        'success': false,
        'data': null
      }, 409);
    }
  }
}
```

```
    await Customer().query().insert(requestData);

    return Response.json([
        "success": true,
        "message": "Berhasil memasukkan data",
        "data": requestData,
    ], 201);
} catch (e) {
    if (e is ValidationException) {
        final errorMessages = e.message;
        return Response.json({
            "success": false,
            "message": errorMessages,
            "data": null,
        }, 400);
    } else {
        return Response.json({
            "success": false,
            "message": e.toString(),
            "data": null,
        }, 500);
    }
}
```

Update

```
Future<Response> update(Request req, String id) async {
  try {
    req.validate({
      'cust_name': 'string|max_length:100',
      'cust_address': 'string|max_length:50',
      'cust_city': 'string|max_length:20',
      'cust_state': 'string|max_length:5',
      'cust_zip': 'string|max_length:7',
      'cust_country': 'string|max_length:25',
      'cust_telp': 'string|max_length:15',
    });

    final requestData = req.input();
    final customer =
      await Customer().query().where('cust_id', '=', id).first();
    if (customer == null) {
      return Response.json({
        "success": false,
        "message": "Customer tidak ditemukan",
        "data": null,
      }, 404);
    }
    await Customer().query().where('cust_id', '=', id).update(requestData);
    final updatedData = await Customer().query().get();

    return Response.json({
      "success": true,
      "message": "Berhasil mengupdate data",
      "data": updatedData,
    }, 200);
  } catch (e) {
    if (e is ValidationException) {
      final errorMessages = e.message;
      return Response.json({
        "success": false,
        "message": errorMessages,
        "data": null,
      }, 400);
    } else {
      return Response.json({
        "success": false,
        "message": e.toString(),
        "data": null,
      }, 500);
    }
  }
}
```

Destroy:

```
Future<Response> destroy(String id) async {
  try {
    await Customer().query().where('cust_id', '=', id).delete();
    return Response.json({
      "success": true,
      "message": "Berhasil menghapus data",
      "data": null,
    }, 200);
  } catch (e) {
    return Response.json({
      "success": false,
      "message": e.toString(),
      "data": null,
    }, 500);
  }
}
```

2. Orders_controller

Index:

```
Future<Response> index() async {
  try {
    final orderData = await Order().query().get();
    return Response.json([
      {
        "success": true,
        "message": "Berhasil menampilkan data",
        "data": orderData,
      }, 200);
  } catch (e) {
    return Response.json([
      {
        "success": false,
        "message": e.toString(),
        "data": null,
      }, 500);
  }
}
```

Store:

```
Future<Response> store(Request req) async {
  try {
    req.validate({
      'order_num': 'required|numeric|max_length:11',
      'order_date': 'required|date',
      'cust_id': 'required|string|max_length:5',
    });

    final requestData = req.input();
    final existingProduct = await Order()
      .query()
      .where('order_num', '=', requestData['order_num'])
      .first();

    if (existingProduct != null) {
      return Response.json({
        'message': 'Order dengan nomor ini sudah ada.',
        'success': false,
        'data': null
      }, 409);
    }
    await Order().query().insert(requestData);
  }
}
```

```
        return Response.json({
            "success": true,
            "message": "Berhasil memasukkan data",
            "data": requestData,
        }, 201);
    } catch (e) {
        if (e is ValidationException) {
            final errorMessages = e.message;
            return Response.json({
                "success": false,
                "message": errorMessages,
                "data": null,
            }, 400);
        } else {
            return Response.json({
                "success": false,
                "message": e.toString(),
                "data": null,
            }, 500);
        }
    }
}
```

Update

```
Future<Response> update(Request req, int id) async {
    try {
        req.validate({
            'order_date': 'date',
            'cust_id': 'string|max_length:5',
        });

        final requestData = req.input();
        final order = await Order().query().where('order_num', '=', id).first();
        if (order == null) {
            return Response.json({
                "success": false,
                "message": "Order tidak ditemukan",
                "data": null,
            }, 404);
        }
        await Order().query().where('order_num', '=', id).update(requestData);
        final updatedData = await Order().query().get();
```

```
        return Response.json({
            "success": true,
            "message": "Berhasil mengupdate data",
            "data": updatedData,
        }, 200);
    } catch (e) {
        if (e is ValidationException) {
            final errorMessages = e.message;
            return Response.json({
                "success": false,
                "message": errorMessages,
                "data": null,
            }, 400);
        } else {
            return Response.json({
                "success": false,
                "message": e.toString(),
                "data": null,
            }, 500);
        }
    }
}
```

Destroy:

```
Future<Response> destroy(int id) async {
    try {
        await Order().query().where('order_num', '=', id).delete();
        return Response.json({
            "success": true,
            "message": "Berhasil menghapus data",
            "data": null,
        }, 200);
    } catch (e) {
        return Response.json({
            "success": false,
            "message": e.toString(),
            "data": null,
        }, 500);
    }
}
```

3. Vendor_controller

Index:

```
Future<Response> index() async {
  try {
    final vendorData = await Vendor().query().get();
    return Response.json({
      "success": true,
      "message": "Berhasil menampilkan data",
      "data": vendorData,
    }, 200);
  } catch (e) {
    return Response.json({
      "success": false,
      "message": e.toString(),
      "data": null,
    }, 500);
  }
}
```

Store:

```
Future<Response> store(Request req) async {
  try {
    req.validate({
      'vend_id': 'required|string|max_length:5',
      'vend_name': 'required|string|max_length:50',
      'vend_address': 'required|string',
      'vend_kota': 'required|string',
      'vend_state': 'required|string|max_length:5',
      'vend_zip': 'required|string|max_length:7',
      'vend_country': 'required|string|max_length:25',
    });

    final requestData = req.input();
    final existingProduct = await Vendor()
      .query()
      .where('vend_id', '=', requestData['vend_id'])
      .first();

    if (existingProduct != null) {
      return Response.json({
        'message': 'Vendor dengan nomor ini sudah ada.',
        'success': false,
        'data': null
      }, 409);
    }
    await Vendor().query().insert(requestData);
```

```
        return Response.json({
            "success": true,
            "message": "Berhasil memasukkan data",
            "data": requestData,
        }, 201);
    } catch (e) {
        if (e is ValidationException) {
            final errorMessages = e.message;
            return Response.json({
                "success": false,
                "message": errorMessages,
                "data": null,
            }, 400);
        } else {
            return Response.json({
                "success": false,
                "message": e.toString(),
                "data": null,
            }, 500);
        }
    }
}
```

Update

```
Future<Response> update(Request req, String id) async {
    try {
        req.validate({
            'vend_name': 'string|max_length:50',
            'vend_address': 'string',
            'vend_kota': 'string',
            'vend_state': 'string|max_length:5',
            'vend_zip': 'string|max_length:7',
            'vend_country': 'string|max_length:25',
        });
    }

    final requestData = req.input();
    final vendor = await Vendor().query().where('vend_id', '=', id).first();
    if (vendor == null) {
        return Response.json({
            "success": false,
            "message": "Vendor tidak ditemukan",
            "data": null,
        }, 404);
    }
    await Vendor().query().where('vend_id', '=', id).update(requestData);
    final updatedData = await Vendor().query().get();
```

```
    return Response.json({
        "success": true,
        "message": "Berhasil mengupdate data",
        "data": updatedData,
    }, 200);
} catch (e) {
    if (e is ValidationException) {
        final errorMessages = e.message;
        return Response.json({
            "success": false,
            "message": errorMessages,
            "data": null,
        }, 400);
    } else {
        return Response.json({
            "success": false,
            "message": e.toString(),
            "data": null,
        }, 500);
    }
}
```

Destroy:

```
Future<Response> destroy(String id) async {
    try {
        await Vendor().query().where('vend_id', '=', id).delete();
        return Response.json({
            "success": true,
            "message": "Berhasil menghapus data",
            "data": null,
        }, 200);
    } catch (e) {
        return Response.json({
            "success": false,
            "message": e.toString(),
            "data": null,
        }, 500);
    }
}
```

4. Product_controller

Index:

```
Future<Response> index() async {
  try {
    final productData = await Product().query().get();
    return Response.json({
      "success": true,
      "message": "Berhasil menampilkan data",
      "data": productData,
    }, 200);
  } catch (e) {
    return Response.json({
      "success": false,
      "message": e.toString(),
      "data": null,
    }, 500);
  }
}
```

Store:

```
Future<Response> store(Request req) async {
  try {
    req.validate({
      'prod_id': 'required|string|max_length:10',
      'vend_id': 'required|string|max_length:5',
      'prod_name': 'required|string|max_length:25',
      'prod_price': 'required|numeric|max_length:11',
      'prod_desc': 'required|string',
    });

    final requestData = req.input();
    final existingProduct = await Product()
      .query()
      .where('vend_id', '=', requestData['vend_id'])
      .first();

    if (existingProduct != null) {
      return Response.json({
        'message': 'Product dengan id ini sudah ada.',
        'success': false,
        'data': null
      }, 409);
    }
    await Product().query().insert(requestData);
```

```

        return Response.json({
            "success": true,
            "message": "Berhasil memasukkan data",
            "data": requestData,
        }, 201);
    } catch (e) {
        if (e is ValidationException) {
            final errorMessages = e.message;
            return Response.json({
                "success": false,
                "message": errorMessages,
                "data": null,
            }, 400);
        } else {
            return Response.json({
                "success": false,
                "message": e.toString(),
                "data": null,
            }, 500);
        }
    }
}

```

Update

```

Future<Response> update(Request req, String id) async {
    try {
        req.validate({
            'vend_id': 'string|max_length:5',
            'prod_name': 'string|max_length:25',
            'prod_price': 'numeric|max_length:11',
            'prod_desc': 'string',
        });

        final requestData = req.input();
        final product = await Product().query().where('prod_id', '=', id).first();
        if (product == null) {
            return Response.json({
                "success": false,
                "message": "Product tidak ditemukan",
                "data": null,
            }, 404);
        }
        await Product().query().where('prod_id', '=', id).update(requestData);
        final updatedData = await Product().query().get();
    }
}

```

```
    return Response.json({
      "success": true,
      "message": "Berhasil mengupdate data",
      "data": updatedData,
    }, 200);
} catch (e) {
  if (e is ValidationException) {
    final errorMessages = e.message;
    return Response.json({
      "success": false,
      "message": errorMessages,
      "data": null,
    }, 400);
  } else {
    return Response.json({
      "success": false,
      "message": e.toString(),
      "data": null,
    }, 500);
  }
}
```

Destroy:

```
Future<Response> destroy(String id) async {
try {
  await Product().query().where('prod_id', '=', id).delete();
  return Response.json({
    "success": true,
    "message": "Berhasil menghapus data",
    "data": null,
  }, 200);
} catch (e) {
  return Response.json({
    "success": false,
    "message": e.toString(),
    "data": null,
  }, 500);
}
}
```

5. Product_note_controller

Index:

```
Future<Response> index() async {
  try {
    final productNoteData = await ProductNote().query().get();
    return Response.json({
      "success": true,
      "message": "Berhasil menampilkan data",
      "data": productNoteData,
    }, 200);
  } catch (e) {
    return Response.json({
      "success": false,
      "message": e.toString(),
      "data": null,
    }, 500);
  }
}
```

Store:

```
Future<Response> store(Request req) async {
  try {
    req.validate({
      'note_id': 'required|string|max_length:5',
      'prod_id': 'required|string|max_length:10',
      'note_date': 'required|date',
      'note_text': 'required|string',
    });

    final requestData = req.input();
    final existingProduct = await ProductNote()
      .query()
      .where('note_id', '=', requestData['note_id'])
      .first();

    if (existingProduct != null) {
      return Response.json({
        'message': 'Catatan product dengan id ini sudah ada.',
        'success': false,
        'data': null
      }, 409);
    }
    await ProductNote().query().insert(requestData);
```

```
        return Response.json({
            "success": true,
            "message": "Berhasil memasukkan data",
            "data": requestData,
        }, 201);
    } catch (e) {
        if (e is ValidationException) {
            final errorMessages = e.message;
            return Response.json({
                "success": false,
                "message": errorMessages,
                "data": null,
            }, 400);
        } else {
            return Response.json({
                "success": false,
                "message": e.toString(),
                "data": null,
            }, 500);
        }
    }
}
```

Update

```
Future<Response> update(Request req, String id) async {
    try {
        req.validate({
            'prod_id': 'string|max_length:10',
            'note_date': 'date',
            'note_text': 'string',
        });

        final requestData = req.input();
        final productNote =
            await ProductNote().query().where('note_id', '=', id).first();
        if (productNote == null) {
            return Response.json({
                "success": false,
                "message": "Product tidak ditemukan",
                "data": null,
            }, 404);
        }
        await ProductNote().query().where('note_id', '=', id).update(requestData);
        final updatedData = await ProductNote().query().get();
```

```
        return Response.json({
            "success": true,
            "message": "Berhasil mengupdate data",
            "data": updatedData,
        }, 200);
    } catch (e) {
        if (e is ValidationException) {
            final errorMessages = e.message;
            return Response.json({
                "success": false,
                "message": errorMessages,
                "data": null,
            }, 400);
        } else {
            return Response.json({
                "success": false,
                "message": e.toString(),
                "data": null,
            }, 500);
        }
    }
}
```

Destroy:

```
Future<Response> destroy(String id) async {
    try {
        await ProductNote().query().where('note_id', '=', id).delete();
        return Response.json({
            "success": true,
            "message": "Berhasil menghapus data",
            "data": null,
        }, 200);
    } catch (e) {
        return Response.json({
            "success": false,
            "message": e.toString(),
            "data": null,
        }, 500);
    }
}
```

6. Order_item_controller

Index:

```
Future<Response> index() async {
  try {
    final orderItemsData = await OrderItems().query().get();
    return Response.json({
      "success": true,
      "message": "Berhasil menampilkan data",
      "data": orderItemsData,
    }, 200);
  } catch (e) {
    return Response.json({
      "success": false,
      "message": e.toString(),
      "data": null,
    }, 500);
  }
}
```

Store:

```
Future<Response> store(Request req) async {
  try {
    req.validate({
      'order_item': 'required|numeric|max_length:11',
      'order_num': 'required|numeric|max_length:11',
      'prod_id': 'required|string|max_length:10',
      'quantity': 'required|numeric|max_length:11',
      'size': 'required|numeric|max_length:11',
    });

    final requestData = req.input();
    final existingProduct = await OrderItems()
      .query()
      .where('order_item', '=', requestData['order_item'])
      .first();

    if (existingProduct != null) {
      return Response.json({
        'message': 'Item order dengan id ini sudah ada.',
        'success': false,
        'data': null
      }, 409);
    }
    await OrderItems().query().insert(requestData);
```

```
        return Response.json({
            "success": true,
            "message": "Berhasil memasukkan data",
            "data": requestData,
        }, 201);
    } catch (e) {
        if (e is ValidationException) {
            final errorMessages = e.message;
            return Response.json({
                "success": false,
                "message": errorMessages,
                "data": null,
            }, 400);
        } else {
            return Response.json({
                "success": false,
                "message": e.toString(),
                "data": null,
            }, 500);
        }
    }
}
```

Update

```
Future<Response> update(Request req, int id) async {
    try {
        req.validate({
            'order_num': 'numeric|max_length:11',
            'prod_id': 'string|max_length:10',
            'quantity': 'numeric|max_length:11',
            'size': 'numeric|max_length:11',
        });
    }

    final requestData = req.input();
    final orderItems =
        await OrderItems().query().where('order_item', '=', id).first();
    if (orderItems == null) {
        return Response.json({
            "success": false,
            "message": "Item order tidak ditemukan",
            "data": null,
        }, 404);
    }
    await OrderItems()
        .query()
        .where('order_item', '=', id)
        .update(requestData);
    final updatedData = await OrderItems().query().get();
```

```
        return Response.json({
            "success": true,
            "message": "Berhasil mengupdate data",
            "data": updatedData,
        }, 200);
    } catch (e) {
        if (e is ValidationException) {
            final errorMessages = e.message;
            return Response.json({
                "success": false,
                "message": errorMessages,
                "data": null,
            }, 400);
        } else {
            return Response.json({
                "success": false,
                "message": e.toString(),
                "data": null,
            }, 500);
        }
    }
}
```

Destroy:

```
Future<Response> destroy(int id) async {
    try {
        await OrderItems().query().where('order_item', '=', id).delete();
        return Response.json({
            "success": true,
            "message": "Berhasil menghapus data",
            "data": null,
        }, 200);
    } catch (e) {
        return Response.json({
            "success": false,
            "message": e.toString(),
            "data": null,
        }, 500);
    }
}
```

Testing

customers:

1. Create

The screenshot shows the Postman interface with the following details:

- Collection:** My Workspace
- Environment:** mws vania
- Request Type:** POST
- URL:** {{base_url}}/api/customer
- Body (JSON):**

```
1 {
2     "cust_id": "C1232",
3     "cust_name": "John Doe",
4     "cust_address": "123 Main Street",
5     "cust_city": "Jakarta",
6     "cust_state": "DKI",
7     "cust_zip": "12345",
8     "cust_country": "Indonesia"
}
```

- Response Status:** 201 Created
- Body (Pretty):**

```
message : Berhasil memasukkan data ,
4 "data": [
5     {
6         "cust_id": "C1232",
7         "cust_name": "John Doe",
8         "cust_address": "123 Main Street",
9         "cust_city": "Jakarta",
10        "cust_state": "DKI",
11        "cust_zip": "12345",
12        "cust_country": "Indonesia",
13    }
]
```

2. Read

The screenshot shows the Postman interface with the following details:

- Collection:** My Workspace
- Environment:** mws vania
- Request Type:** GET
- URL:** {{base_url}}/api/customer
- Headers:** (6)
- Query Params:**

Key	Value	Description
Key	Value	Description

- Response Status:** 200 OK
- Body (Pretty):**

```
3 "message": "Berhasil menampilkan data",
4 "data": [
5     {
6         "cust_id": "C1232",
7         "cust_name": "John Doe",
8         "cust_address": "123 Main Street",
9         "cust_city": "Jakarta",
10        "cust_state": "DKI",
11        "cust_zip": "12345",
12        "cust_country": "Indonesia"
13    }
]
```

3. Update

The screenshot shows the Postman interface. On the left, the 'My Workspace' sidebar lists collections like 'apicrud', 'apicrudheroku', and 'mws vania'. Under 'mws vania', there's a 'customer' folder containing 'GET Get data', 'POST Post data', 'PUT Update data' (which is selected), and 'DEL Delete data'. The main workspace shows a 'PUT {{(base_url)}} /api/customer/C1232' request. The 'Body' tab is selected, showing raw JSON: { "cust_name": "Ibnu" }. The response status is '200 OK' with a response time of 8.22 s and a body size of 708 B. The response content is:

```
1 {  
2   "success": true,  
3   "message": "Berhasil mengupdate data",  
4   "data": [  
5     {  
6       "cust_id": "c1232",  
7       "cust_name": "Ibnu",  
8       "cust_address": "123 Main Street",  
9     }  
]
```

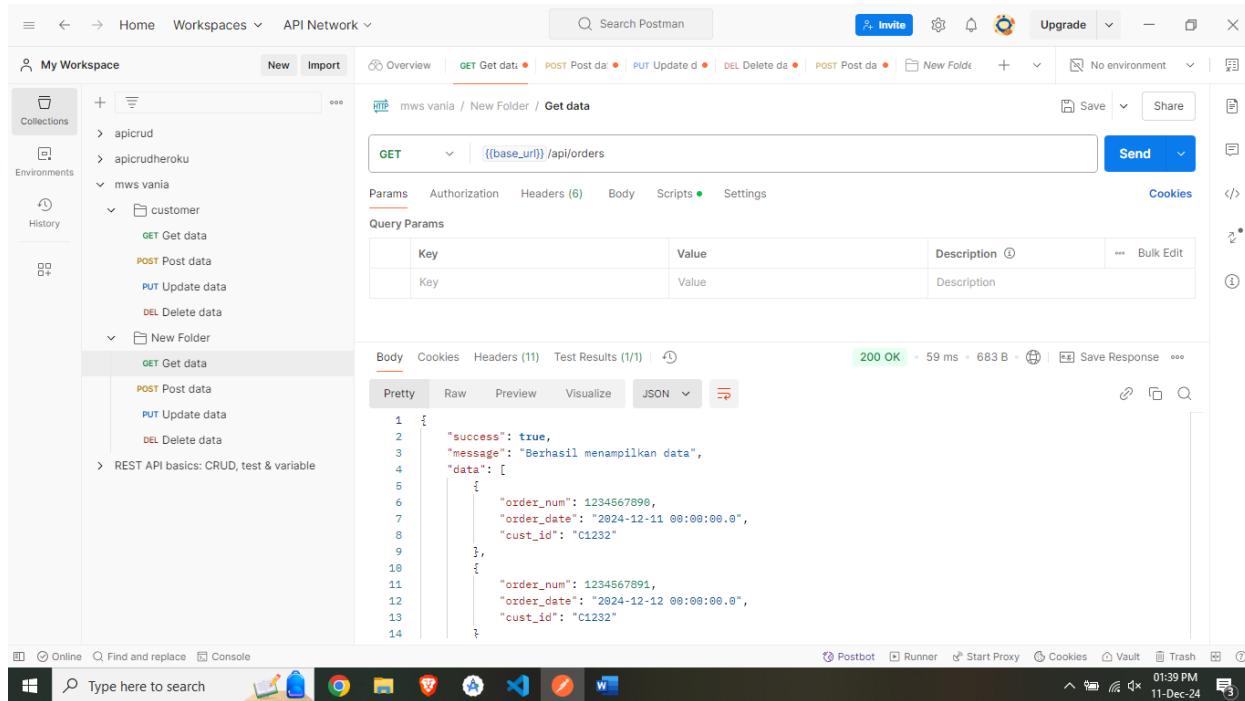
4. Delete

The screenshot shows the Postman interface. The 'My Workspace' sidebar is identical to the previous screenshot. The main workspace shows a 'DELETE {{(base_url)}} /api/customer/C1232' request. The 'Body' tab is selected, showing raw JSON: { "Key": "Value" }. The response status is '200 OK' with a response time of 10.90 s and a body size of 524 B. The response content is:

```
1 {  
2   "success": true,  
3   "message": "Berhasil menghapus data",  
4   "data": null  
5 }
```

orders:

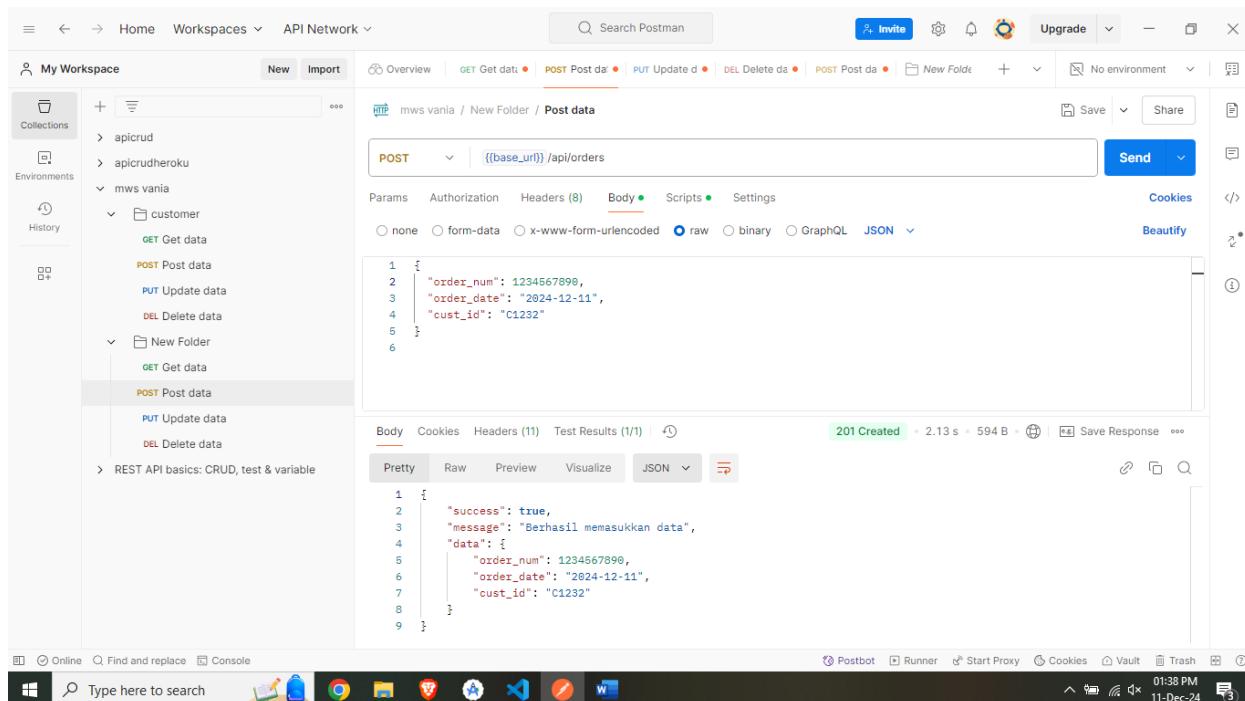
1. Create



The screenshot shows the Postman interface with a workspace named "mws vania". A collection named "customer" contains a "New Folder" which has a "GET Get data" item selected. The request URL is `({base_url}) /api/orders`. The response status is 200 OK, and the response body is:

```
1 {
2     "success": true,
3     "message": "Berhasil menampilkan data",
4     "data": [
5         {
6             "order_num": 1234567890,
7             "order_date": "2024-12-11 00:00:00.0",
8             "cust_id": "C1232"
9         },
10        {
11            "order_num": 1234567891,
12            "order_date": "2024-12-12 00:00:00.0",
13            "cust_id": "C1232"
14        }
15    ]
}
```

2. Read



The screenshot shows the Postman interface with the same workspace and collection setup. A "POST Post data" item in the "customer" collection is selected. The request URL is `({base_url}) /api/orders`. The request body is:

```
1 {
2     "order_num": 1234567890,
3     "order_date": "2024-12-11",
4     "cust_id": "C1232"
5 }
```

The response status is 201 Created, and the response body is:

```
1 {
2     "success": true,
3     "message": "Berhasil memasukkan data",
4     "data": [
5         {
6             "order_num": 1234567890,
7             "order_date": "2024-12-11",
8             "cust_id": "C1232"
9         }
10    ]
}
```

3. Update

The screenshot shows the Postman interface with the following details:

- Collection:** mws vania / New Folder
- Request Type:** PUT
- URL:** {{base_url}} /api/orders/1234567890
- Headers:** (8)
Content-Type: application/json
- Body:** (raw JSON)

```
1 {  
2   "order_date": "2023-01-09"  
3 }
```
- Response:** 200 OK
{"success": true, "message": "Berhasil mengupdate data", "data": [{"order_num": 1234567890, "order_date": "2023-01-09 00:00:00.0", "cust_id": "C1232"}, {"order_num": 1234567891, "order_date": "2024-12-12 00:00:00.0", "cust_id": "C1232"}]}

4. Delete

The screenshot shows the Postman interface with the following details:

- Collection:** mws vania / New Folder
- Request Type:** DELETE
- URL:** {{base_url}} /api/orders/1234567890
- Headers:** (6)
- Query Params:** (Table)

Key	Value	Description	Bulk Edit
Key	Value	Description	
- Response:** 200 OK
{"success": true, "message": "Berhasil menghapus data", "data": null}

vendors:

1. Create

The screenshot shows the Postman interface with a workspace named "mws vania". A collection named "customer" contains a POST request for "Post data". The request URL is `((base_url)) /api/vendor`. The "Body" tab is selected, showing the following JSON payload:

```
1 {
2     "vend_id": "V1234",
3     "vend_name": "Abdul Aziz",
4     "vend_address": "123 Vendor Street",
5     "vend_kota": "Jakarta",
6     "vend_state": "DKI",
7     "vend_zip": "1234567",
8     "vend_country": "Indonesia"
9 }
```

The response status is 201 Created, with a response time of 511 ms and a response size of 695 B. The response body is:

```
1 {
2     "success": true,
3     "message": "Berhasil memasukkan data",
4     "data": [
5         {
6             "vend_id": "V1234",
7             "vend_name": "Abdul Aziz",
8             "vend_address": "123 Vendor Street",
9             "vend_kota": "Jakarta",
10            "vend_state": "DKI".
11        }
12    ]
13 }
```

2. Read

The screenshot shows the Postman interface with the same workspace and collection. A GET request for "Get data" is selected. The request URL is `((base_url)) /api/vendor`. The "Body" tab is selected, showing the following JSON response:

```
1 {
2     "success": true,
3     "message": "Berhasil menampilkan data",
4     "data": [
5         {
6             "vend_id": "V1234",
7             "vend_name": "Abdul Aziz",
8             "vend_address": "123 Vendor Street",
9             "vend_kota": "Jakarta",
10            "vend_state": "DKI",
11        }
12    ]
13 }
```

3. Update

The screenshot shows the Postman interface with the following details:

- Collection:** mws vania / vendor
- Request Type:** PUT
- URL:** {{base_url}}/api/vendor/V1234
- Body (JSON):**

```
1 {
2   "vend_name": "alqalibi"
3 }
```
- Response Status:** 200 OK
- Response Body (Pretty JSON):**

```
1 {
2   "success": true,
3   "message": "Berhasil mengupdate data",
4   "data": [
5     {
6       "vend_id": "v1234",
7       "vend_name": "alqalibi",
8       "vend_address": "123 Vendor Street",
9       "vend_kota": "Jakarta",
10      "vend_state": "DKI",
11      "vend_zip": "1234567",
12      "vend_country": "Indonesia"
13    }
14  ]
15 }
```

4. Delete

The screenshot shows the Postman interface with the following details:

- Collection:** mws vania / vendor
- Request Type:** DELETE
- URL:** {{base_url}}/api/vendor/V1234
- Body (Raw):**

```
1
```
- Response Status:** 200 OK
- Response Body (Pretty JSON):**

```
1 {
2   "success": true,
3   "message": "Berhasil menghapus data",
4   "data": null
5 }
```

orderitems:

1. Create

The screenshot shows the Postman interface with a collection named "mws vania". A POST request is selected for the "order items" endpoint at `POST {{base_url}}/api/orderitem`. The request body is set to "raw" JSON, containing the following data:

```
1 {  
2     "order_item": 1234567890,  
3     "order_num": 1234567890,  
4     "prod_id": "PROD123456",  
5     "quantity": 50,  
6     "size": 42  
7 }  
8
```

The response status is 201 Created, indicating success.

2. Read

The screenshot shows the Postman interface with the same "mws vania" collection. A GET request is selected for the "order items" endpoint at `GET {{base_url}}/api/orderitem`. The request includes a "Query Params" table with a single entry: "Key" and "Value". The response status is 200 OK, indicating success.

3. Update

The screenshot shows the Postman interface with the following details:

- Collection:** My Workspace / verkuil
- Request:** PUT mws vanila / order items / Update data
- Method:** PUT
- URL:** {{base_url}} /api/orderItem/1234567890
- Headers:** (8) - Body (highlighted)
- Body:** JSON
- Body Content:**

```
1: {  
2:   "quantity": 100  
3: }
```
- Response Status:** 200 OK
- Response Time:** 2.18 s
- Response Size:** 619 B

4. Delete

The screenshot shows the Postman interface with the following details:

- Collection:** My Workspace / verkuil
- Request:** DELETE mws vanila / order items / Delete data
- Method:** DELETE
- URL:** {{base_url}} /api/orderItem/1234567890
- Headers:** (6) - Body (highlighted)
- Query Params:** Key: Value
- Response Status:** 200 OK
- Response Time:** 153 ms
- Response Size:** 524 B

products:

1. Create

The screenshot shows the Postman interface with a collection named "My Workspace". A POST request is selected for the "product" endpoint. The request URL is `https://mws.vania.id/product`. The request body is set to raw JSON:

```
1 {  
2     "prod_id": "PROD123456",  
3     "vend_id": "V1234",  
4     "prod_name": "Kaos",  
5     "prod_price": 50000,  
6     "prod_desc": "Deskripsi"  
7 }
```

The response status is 201 Created, and the response body is:

```
1 {  
2     "success": true,  
3     "message": "Berhasil memasukkan data",  
4     "data": [  
5         {  
6             "prod_id": "PROD123456",  
7             "vend_id": "V1234",  
8             "prod_name": "Kaos",  
9             "prod_price": 50000,  
10            "prod_desc": "Deskripsi"  
11        }  
12    ]  
13 }
```

2. Read

The screenshot shows the Postman interface with a collection named "My Workspace". A GET request is selected for the "product" endpoint. The request URL is `https://mws.vania.id/product`. The request body is empty, and the response status is 200 OK. The response body is:

```
1 {  
2     "success": true,  
3     "message": "Berhasil menampilkan data",  
4     "data": [  
5         {  
6             "prod_id": "PROD123456",  
7             "vend_id": "V1234",  
8             "prod_name": "Kaos",  
9             "prod_price": 50000,  
10            "prod_desc": "Deskripsi"  
11        }  
12    ]  
13 }
```

3. Update

The screenshot shows the Postman interface with the following details:

- Collection:** My Workspace
- Request Type:** PUT
- URL:** {{base_url}} /api/product/PROD123456
- Body (JSON):**

```
1 {
2   "prod_name": "celana"
3 }
```
- Response Status:** 200 OK
- Response Body (Pretty JSON):**

```
1 {
2   "success": true,
3   "message": "Berhasil mengupdate data",
4   "data": [
5     {
6       "prod_id": "prod123456",
7       "vend_id": "V1234",
8       "prod_name": "celana",
9       "prod_price": 50000,
10      "prod_desc": "Deskripsi"
11    }
12  ]
13 }
```

4. Delete

The screenshot shows the Postman interface with the following details:

- Collection:** My Workspace
- Request Type:** DELETE
- URL:** {{base_url}} /api/product/PROD123456
- Headers:** (6)
- Query Params:** (1)
- Response Status:** 200 OK
- Response Body (Pretty JSON):**

```
1 {
2   "success": true,
3   "message": "Berhasil menghapus data",
4   "data": null
5 }
```

productnotes:

1. Create

The screenshot shows the Postman interface with the following details:

- Collection:** My Workspace
- Request:** POST {{base_url}}/api/productNote
- Body:** JSON (raw)

```
1 {
2     "note_id": "N1234",
3     "prod_id": "PROD123456",
4     "note_date": "2024-12-11",
5     "note_text": "This is a sample note text for demonstration purposes."
6 }
```
- Response:** 201 Created (386 ms, 662 B)

```
1 {
2     "success": true,
3     "message": "Berhasil memasukkan data",
4     "data": [
5         {
6             "note_id": "N1234",
7             "prod_id": "PROD123456",
8             "note_date": "2024-12-11",
9             "note_text": "This is a sample note text for demonstration purposes."
10        }
11    ]
12 }
```

2. Read

The screenshot shows the Postman interface with the following details:

- Collection:** My Workspace
- Request:** GET {{base_url}}/api/productNote
- Headers:** (11) - includes Authorization, Content-Type, User-Agent, etc.
- Query Params:** (empty)
- Response:** 200 OK (387 ms, 671 B)

```
1 {
2     "success": true,
3     "message": "Berhasil menampilkan data",
4     "data": [
5         {
6             "note_id": "N1234",
7             "prod_id": "PROD123456",
8             "note_date": "2024-12-11 00:00:00.0",
9             "note_text": "This is a sample note text for demonstration purposes."
10        }
11    ]
12 }
```

3. Update

The screenshot shows the Postman interface with a collection named "mws vania / product note". A PUT request is selected with the URL `PUT {{base_url}} /api/productNote/N1234`. The "Body" tab is active, showing the following JSON payload:

```
1 {
2   "note_text": "Contoh note diupdate"
}
```

The response status is 200 OK, and the response body is:

```
1 {
2   "success": true,
3   "message": "Berhasil mengupdate data",
4   "data": [
5     {
6       "note_id": "n1234",
7       "prod_id": "PROD123456",
8       "note_date": "2024-12-11 00:00:00.0",
9       "note_text": "Contoh note diupdate"
10    }
11  ]
12 }
```

4. Delete

The screenshot shows the Postman interface with the same collection. A DELETE request is selected with the URL `DELETE {{base_url}} /api/productNote/N1234`. The "Body" tab is active, showing the following JSON payload:

```
1 {
2   "note_id": "N1234"
}
```

The response status is 200 OK, and the response body is:

```
1 {
2   "success": true,
3   "message": "Berhasil menghapus data",
4   "data": null
5 }
```