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Try `dart pub outdated` for more information.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

```dart
class CreateCustomersTable extends Migration {
  @override
  Future<void> up() async {
    super.up();
    await createTableNotExists('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 createTableNotExists('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 createTableNotExists('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 createTableNotExists('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 createTableNotExists('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");
    });
  }
```

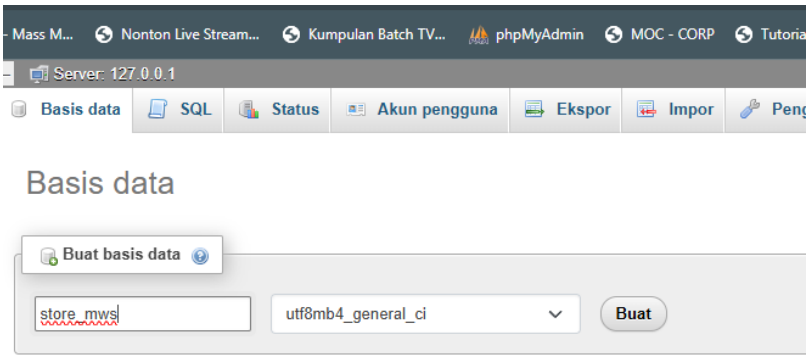
6. Table productnotes

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

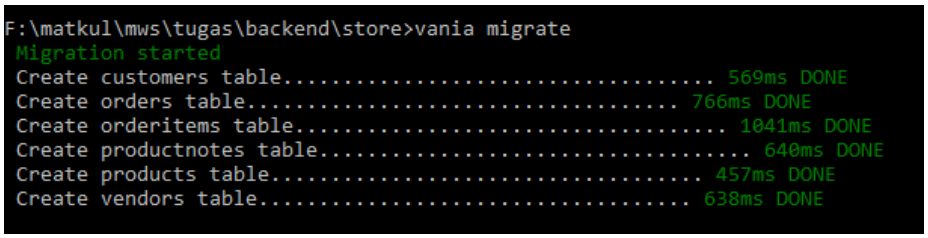
7. Ubah konfigurasi .env

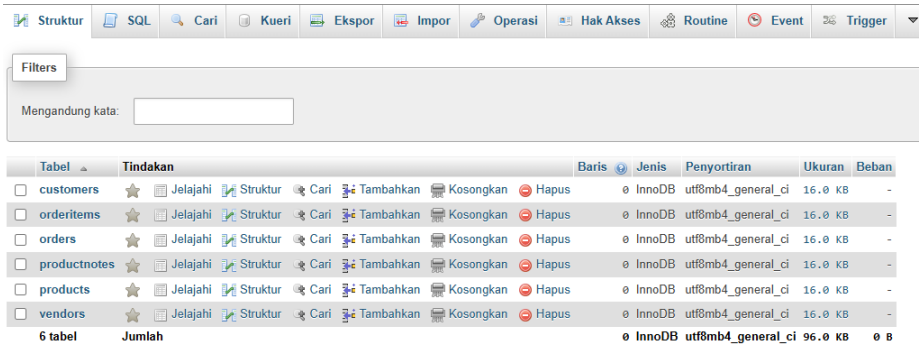
```
# databse driver mysql or postgresql or pgsql
DB_CONNECTION=mysql
DB_HOST=localhost
DB_PORT=3306
DB_DATABASE=store_mws
DB_USERNAME=ibnu
DB_PASSWORD=123
DB_SSL_MODE=false
DB_POOL=false
DB_POOL_SIZE=2
```

8. Create database

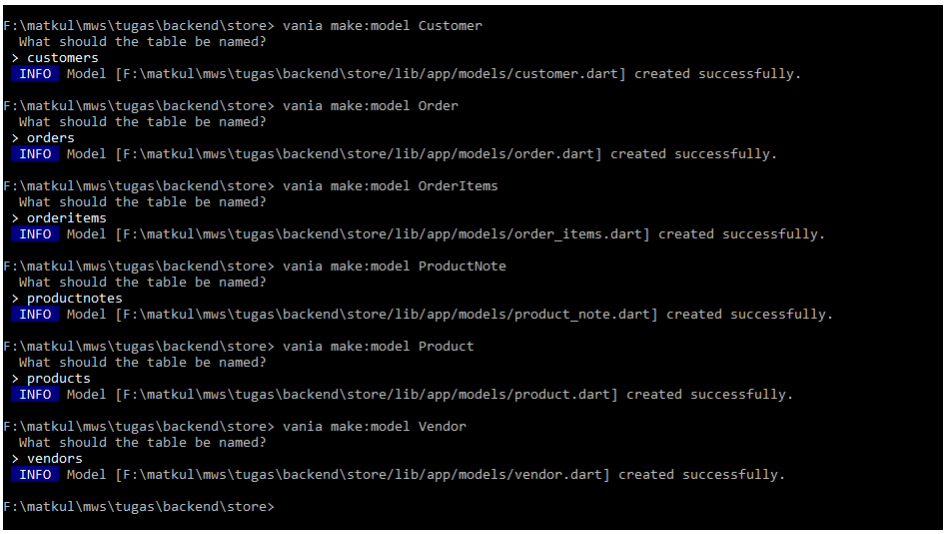


9. Migrate

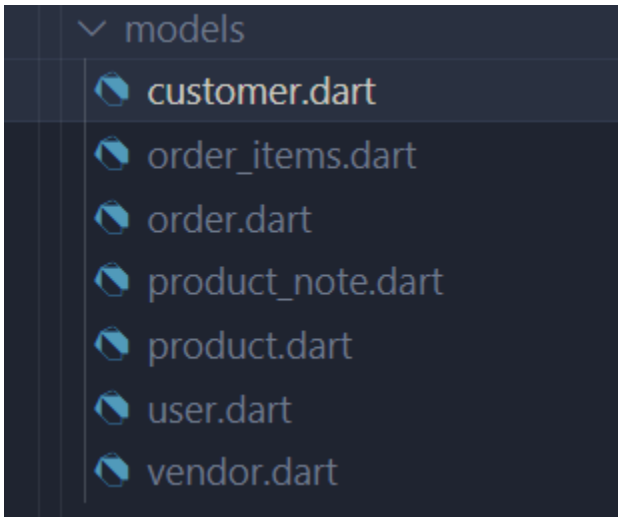




Buat models



Hasil:



Buat Controller

1. Customer_controller
   Index:

```dart
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);
      }
    }
  }
```

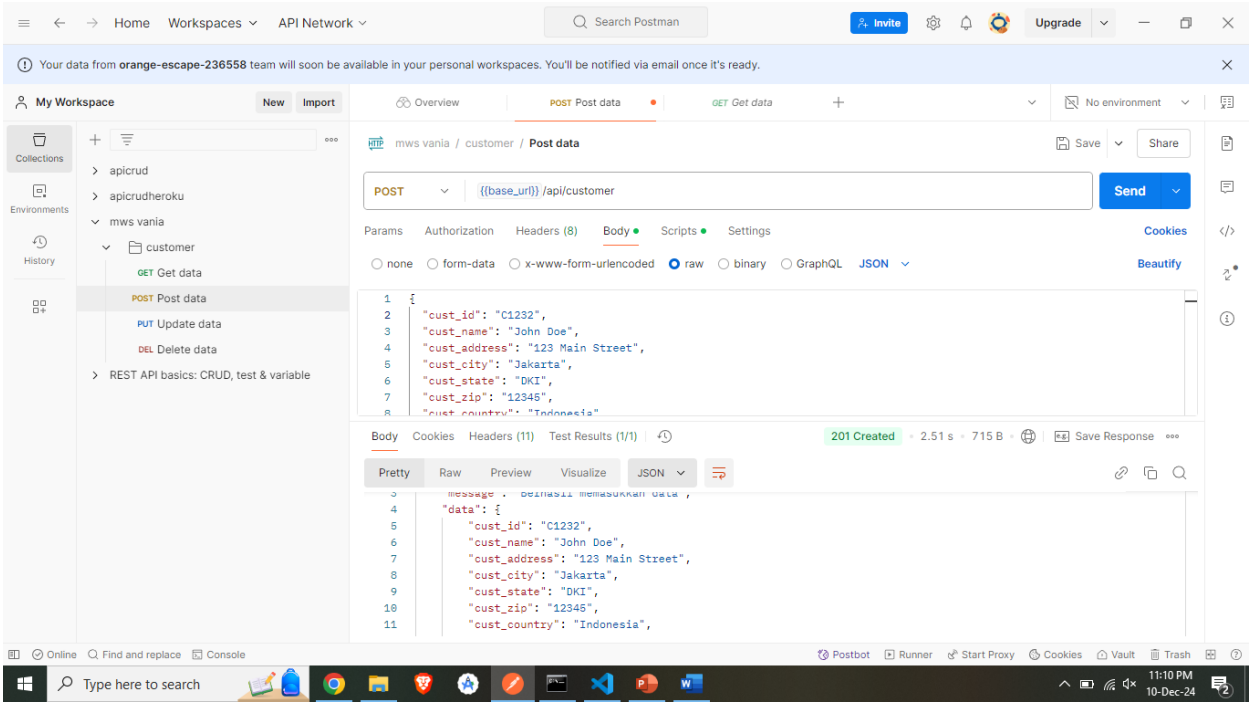
Destroy:

```
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);
  }
}
```

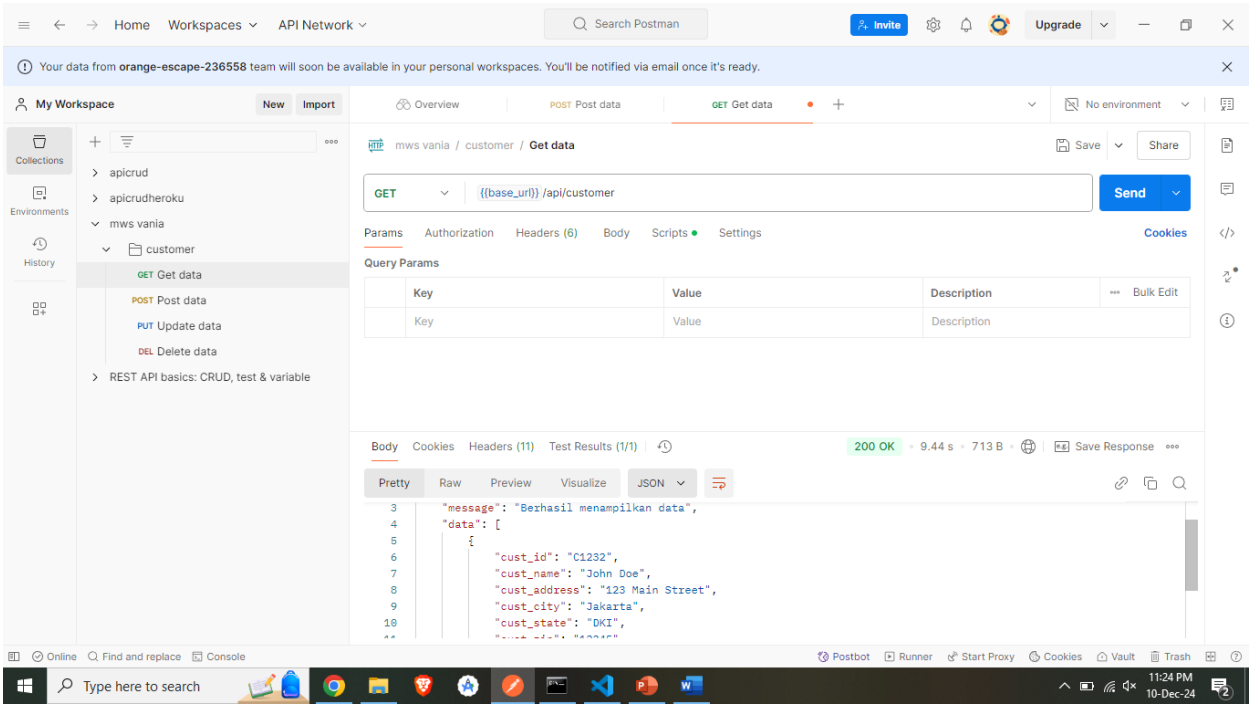
Testing

customers:

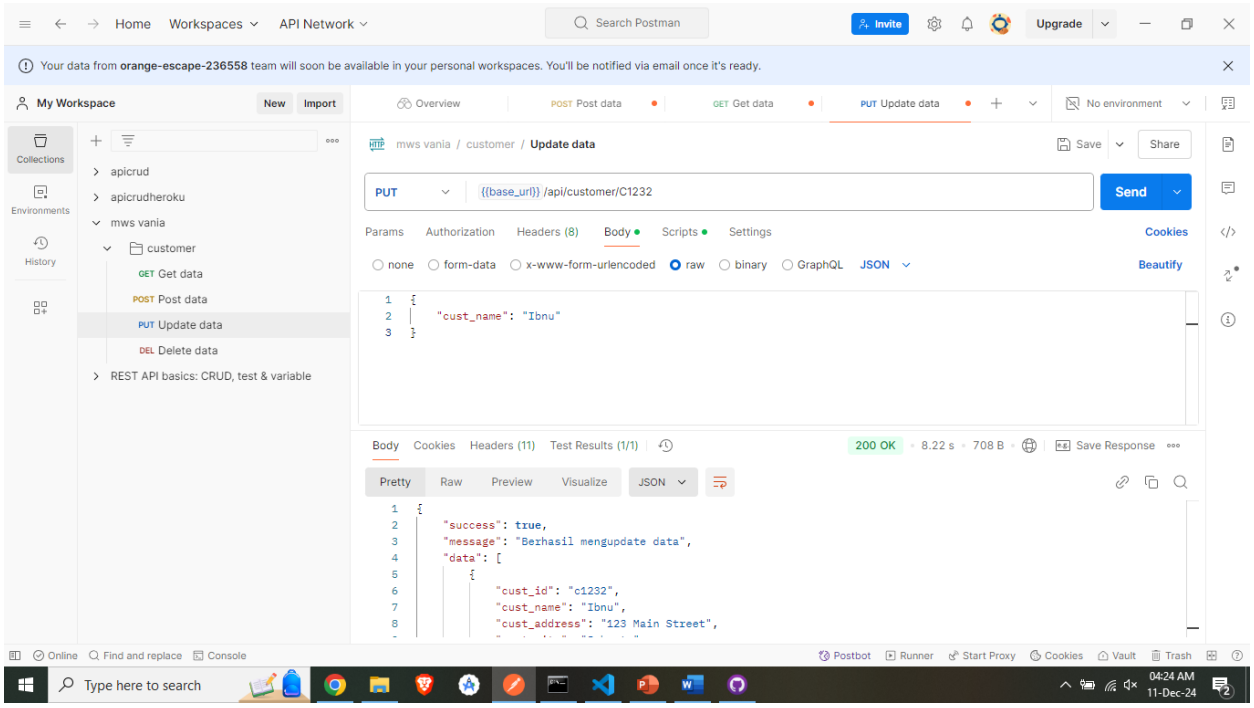
1. Create



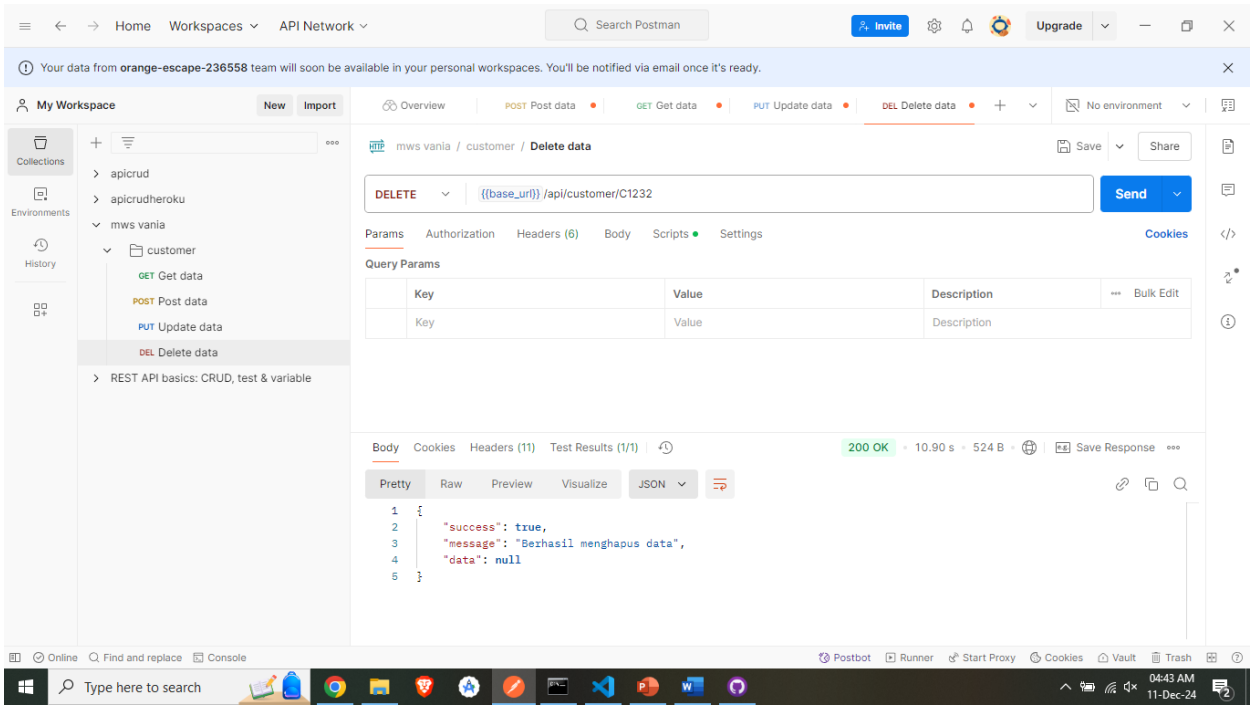
2. Read



3. Update

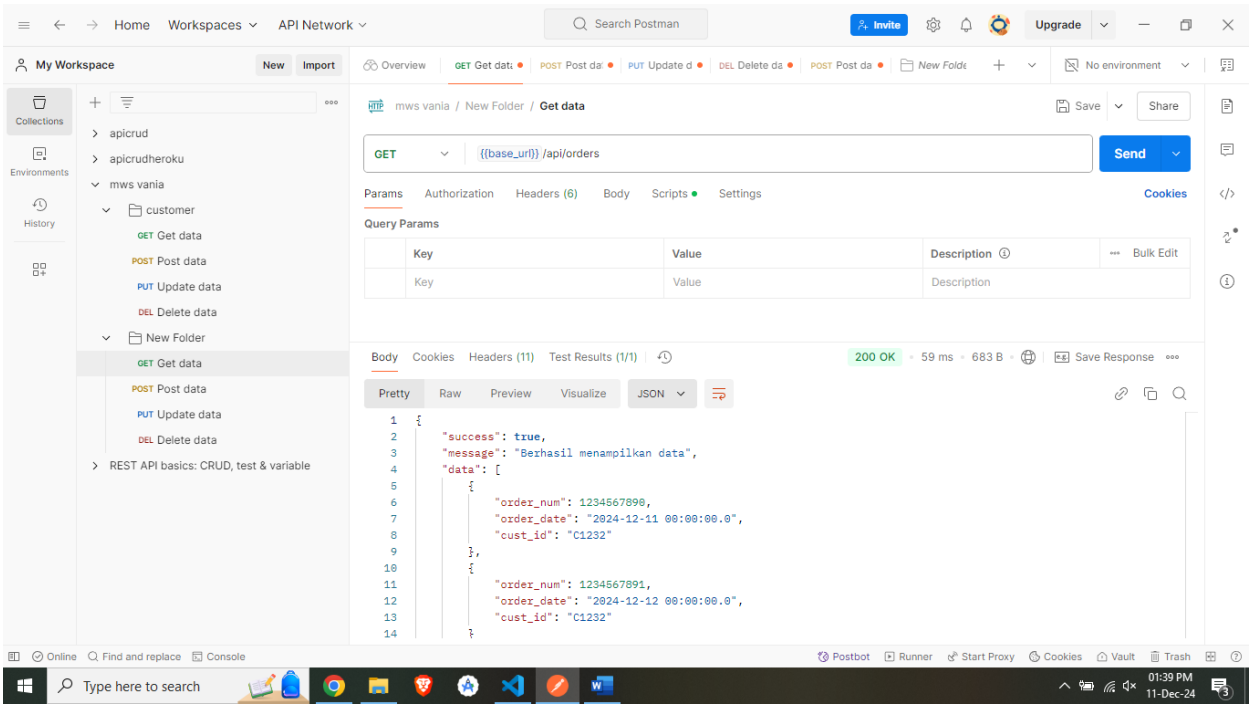


4. Delete

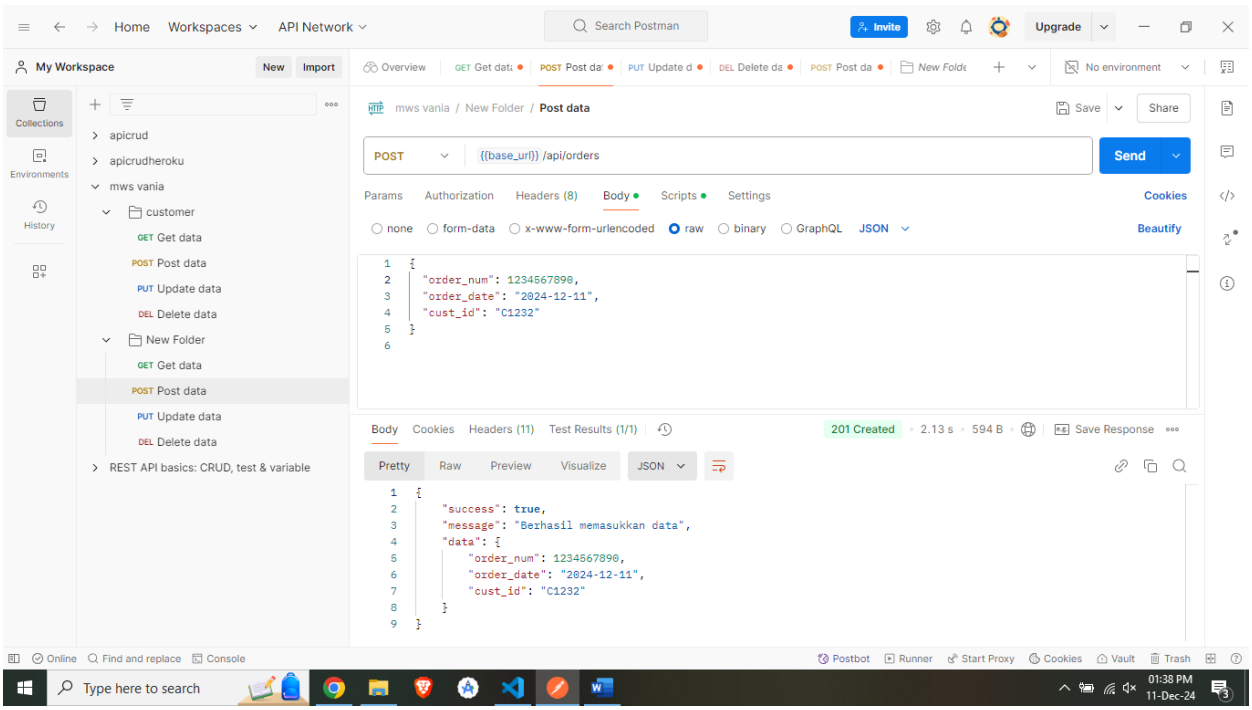


orders:

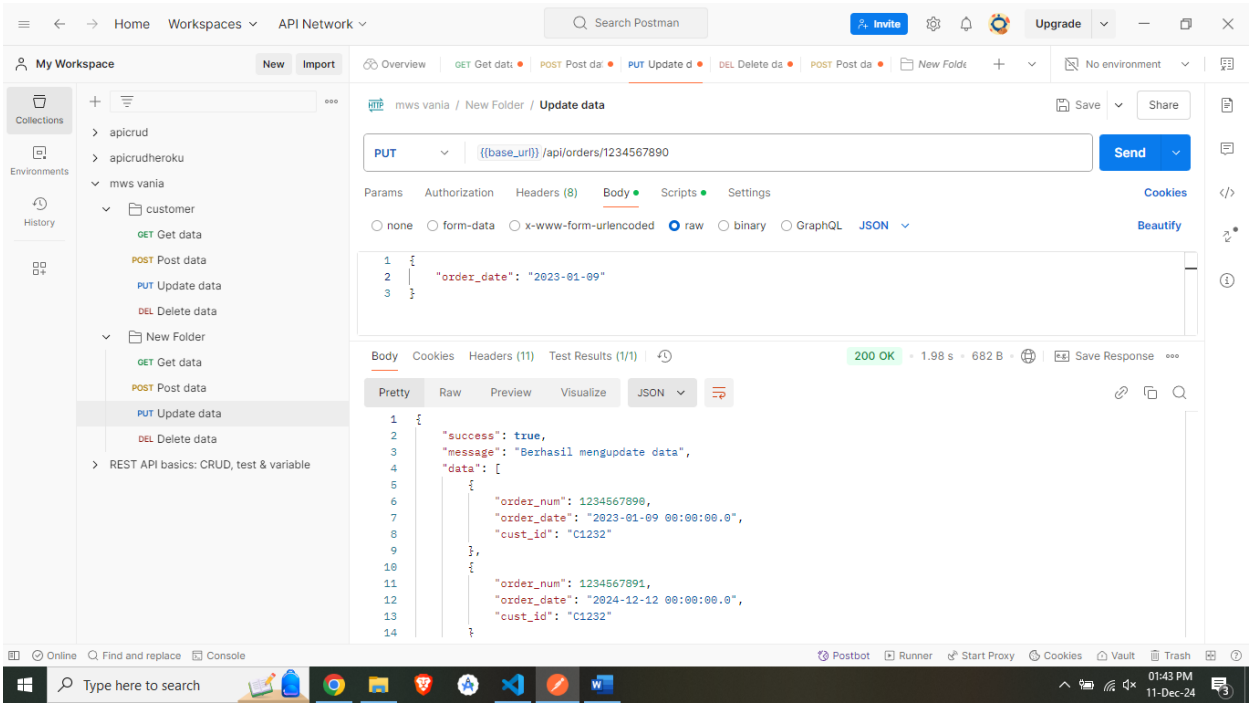
1. Create



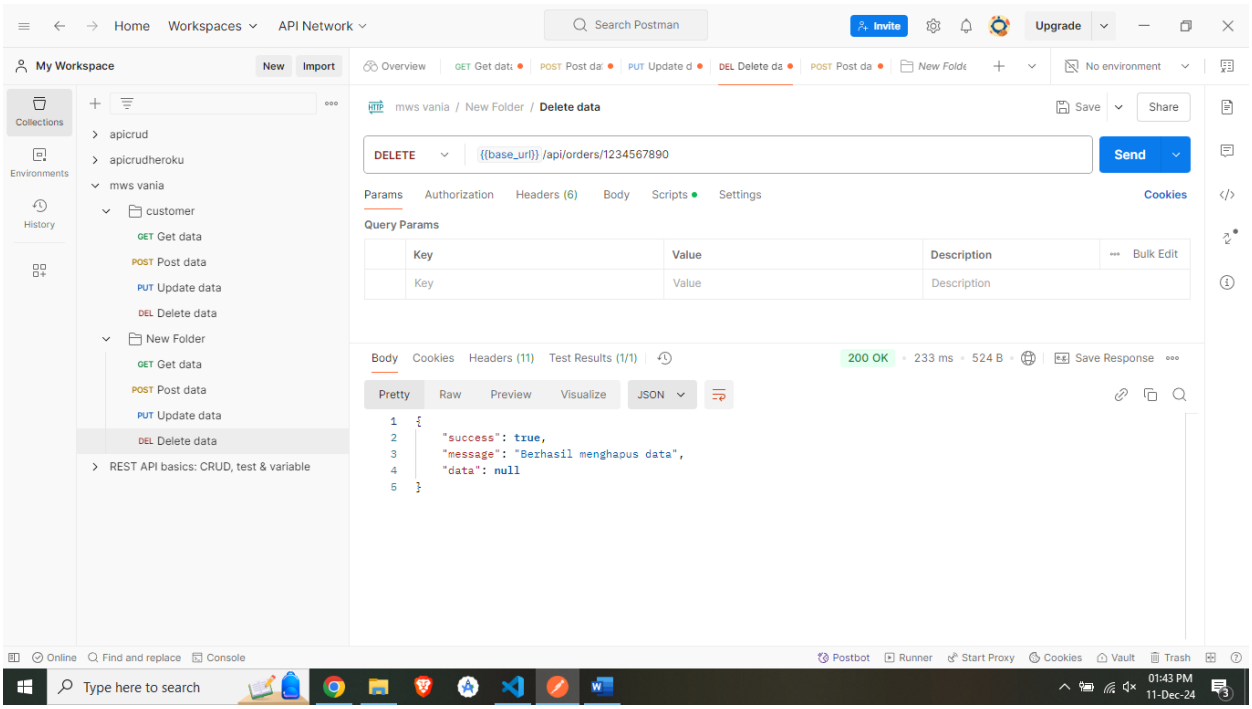
2. Read



3. Update

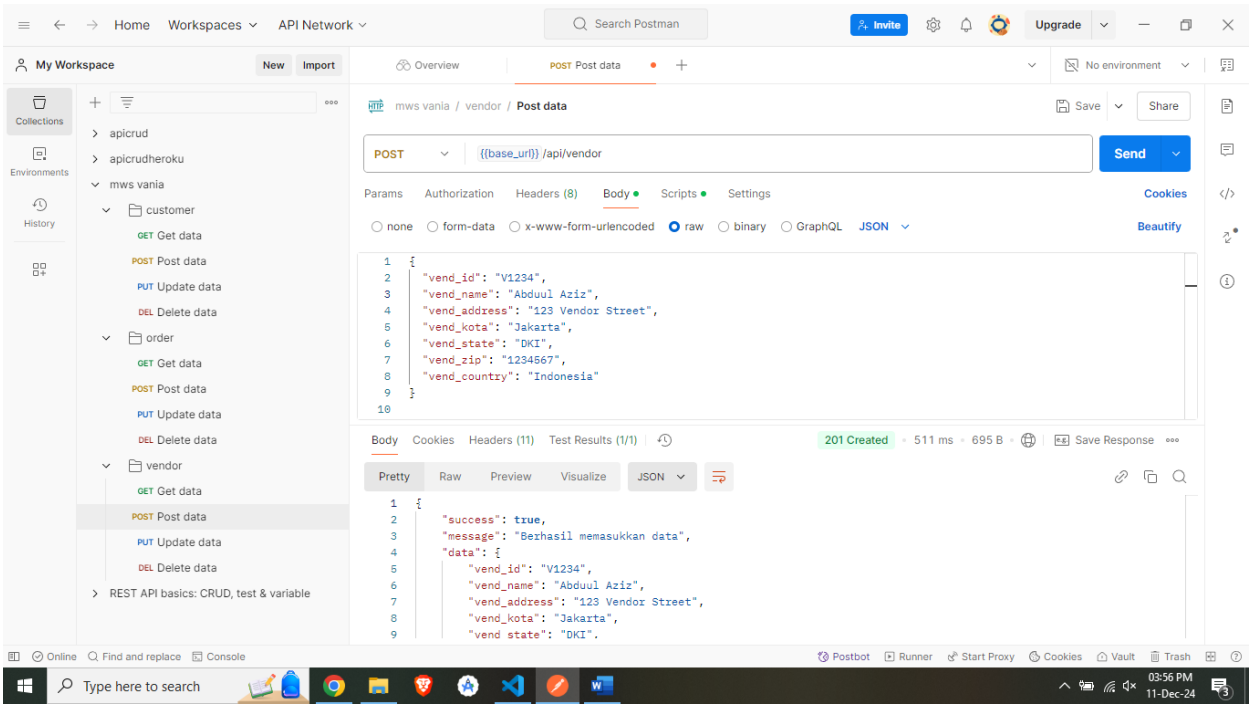


4. Delete

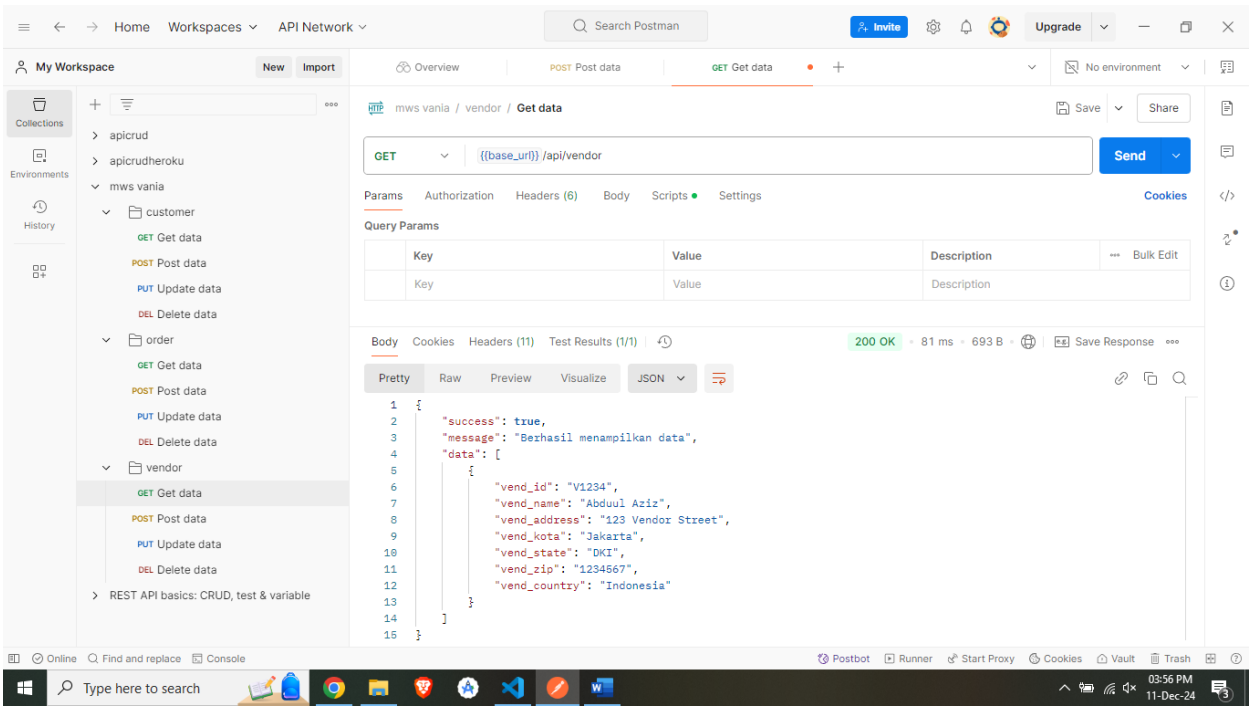


vendors:

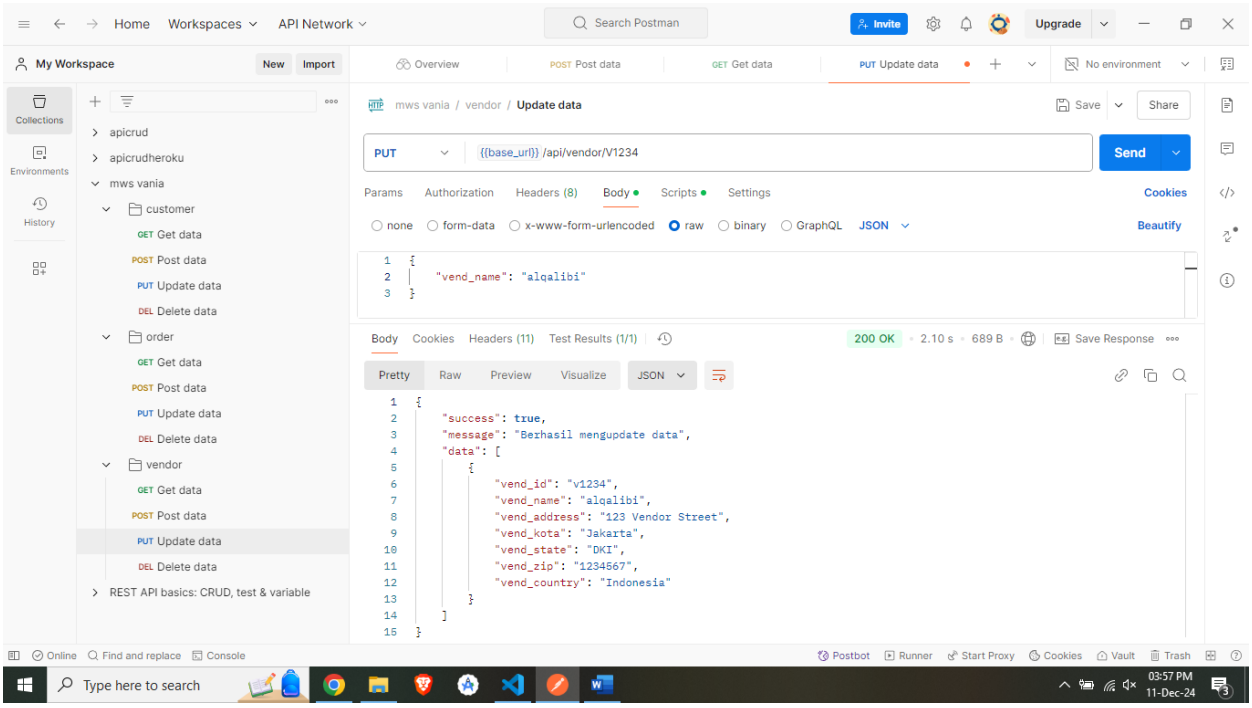
1. Create



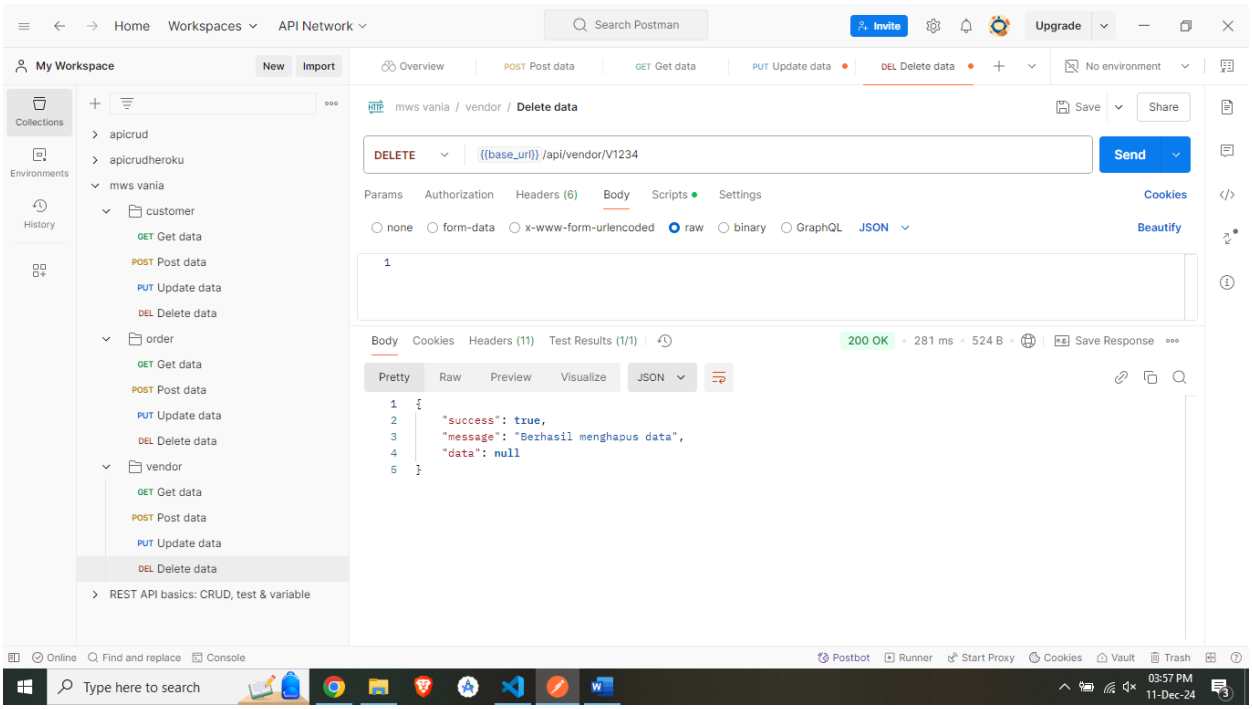
2. Read



3. Update

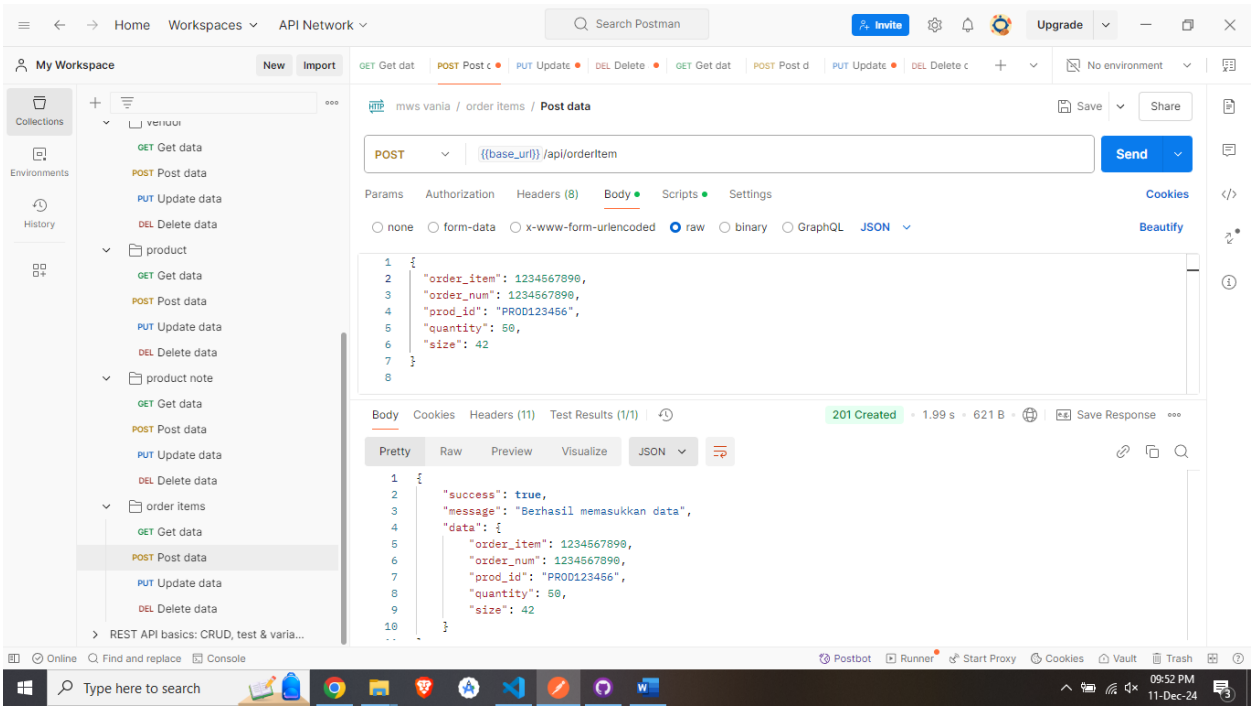


4. Delete

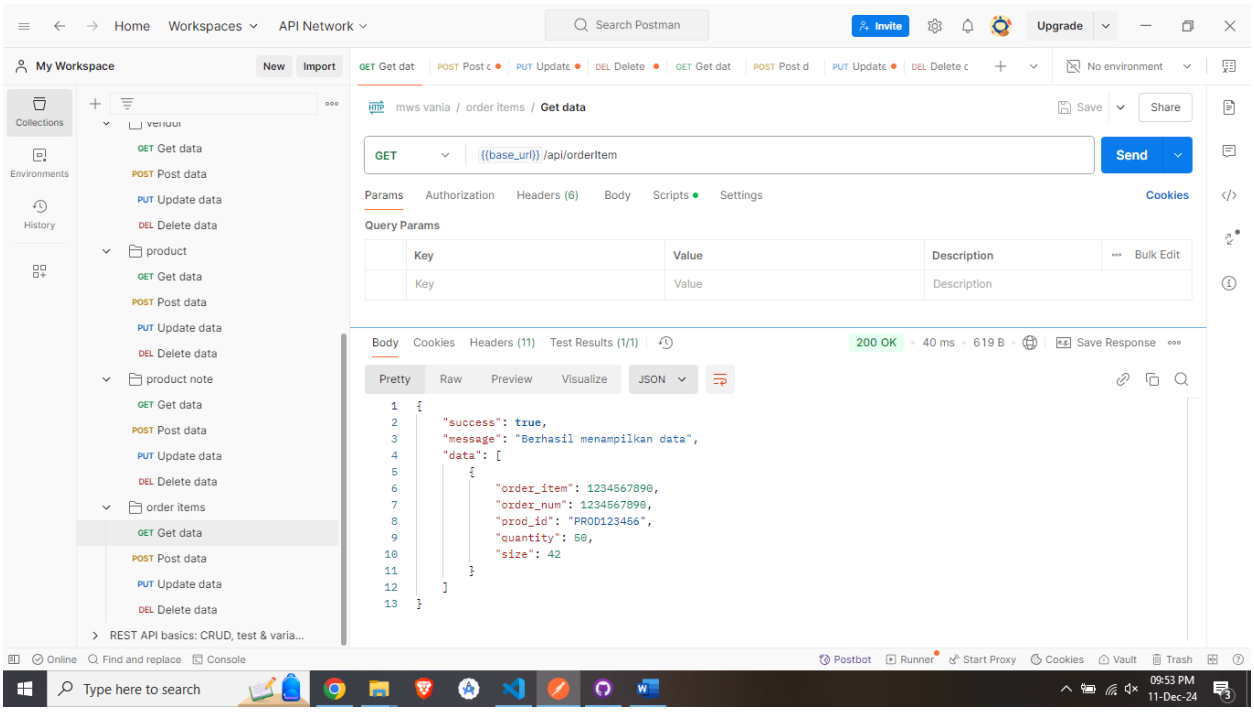


orderitems:

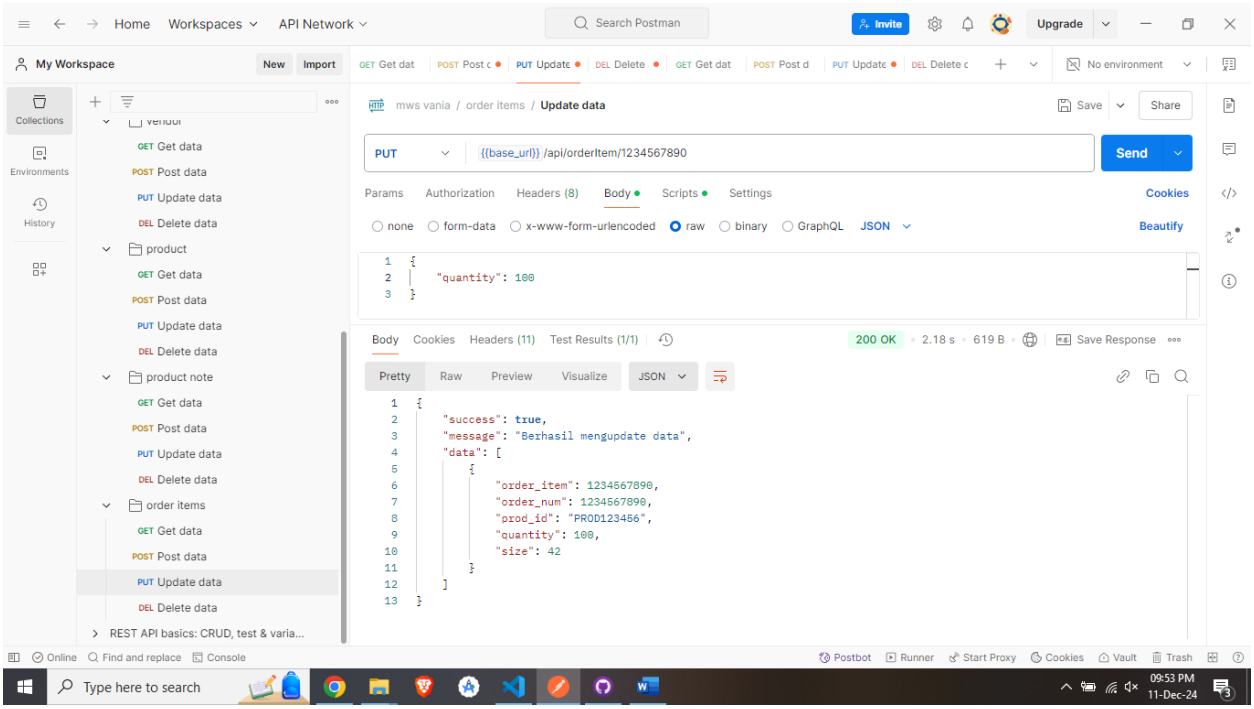
1. Create



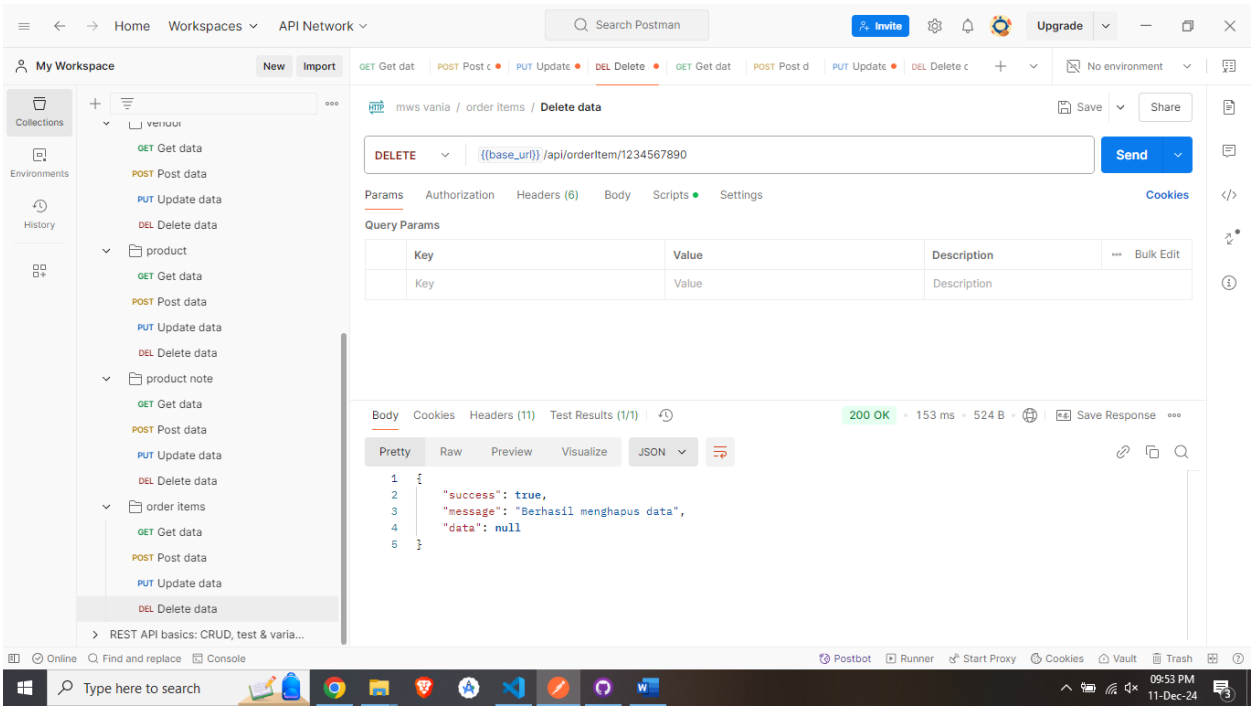
2. Read



3. Update

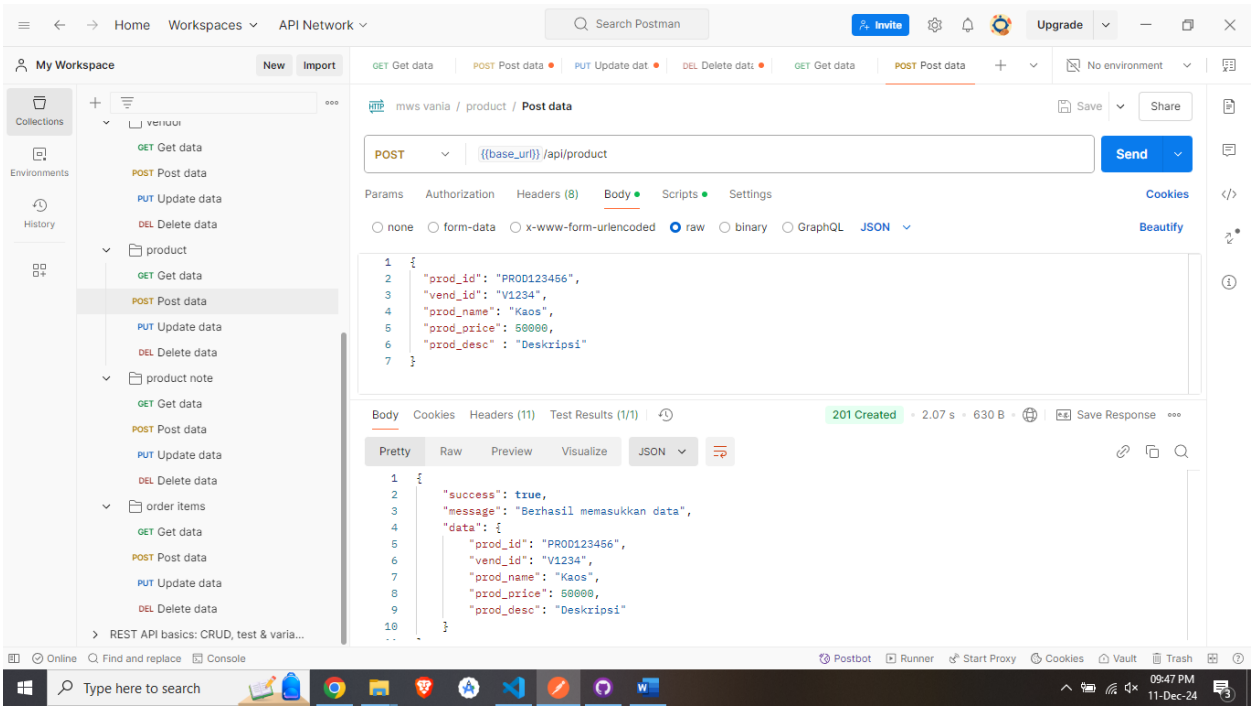


4. Delete

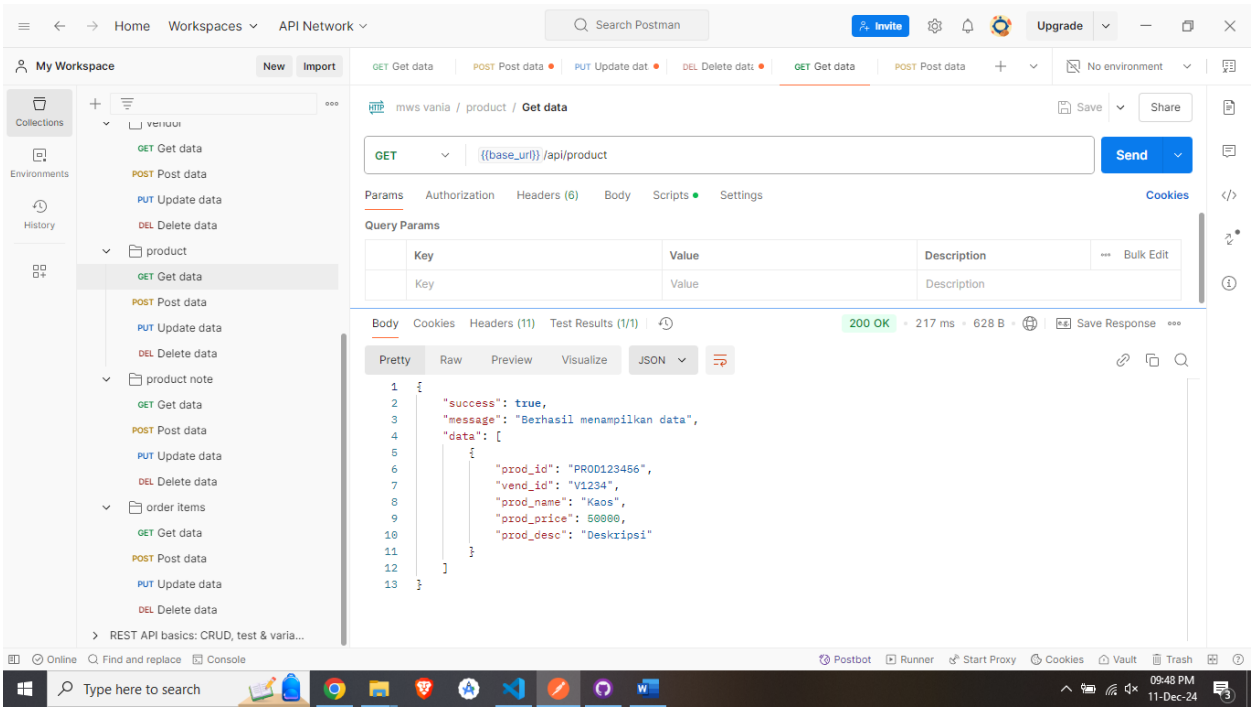


products:

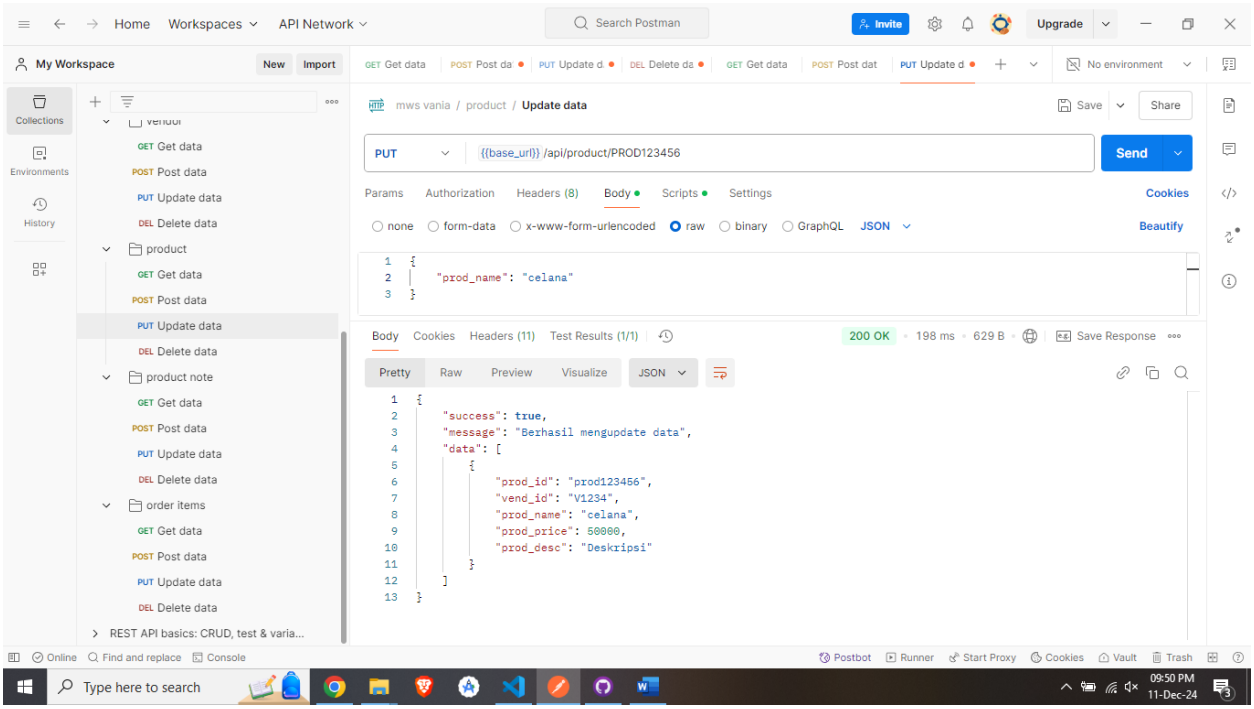
1. Create



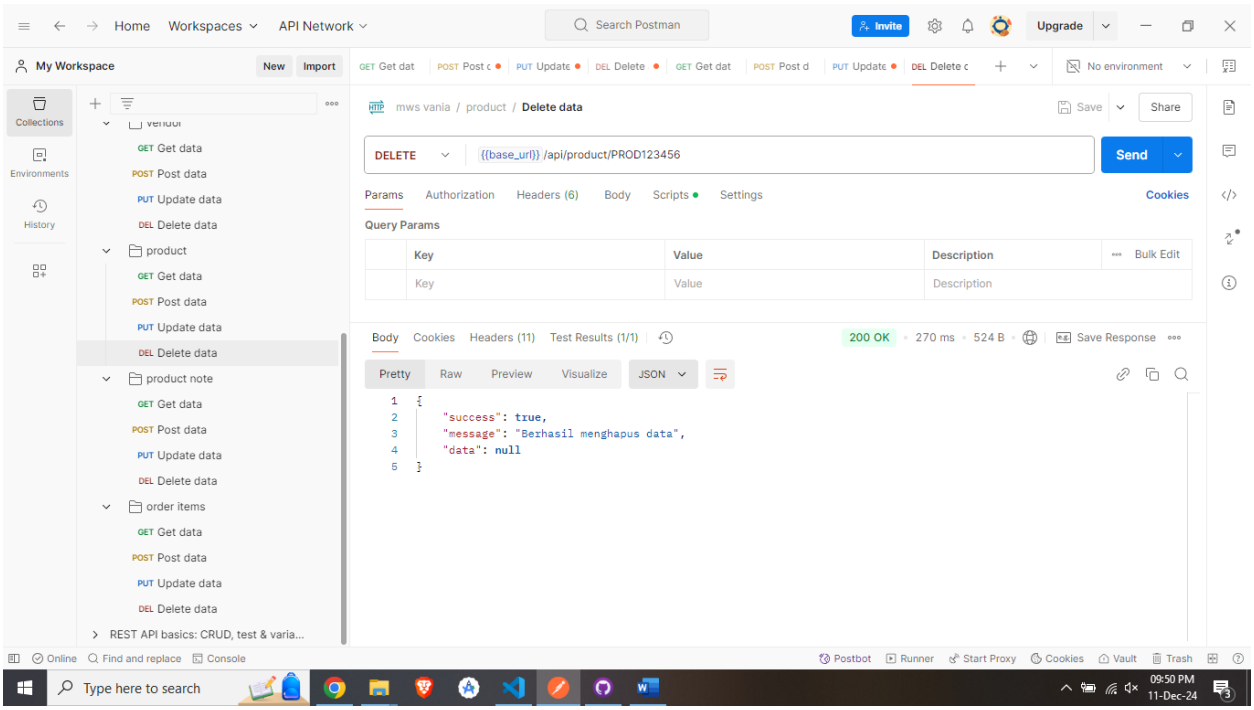
2. Read



3. Update

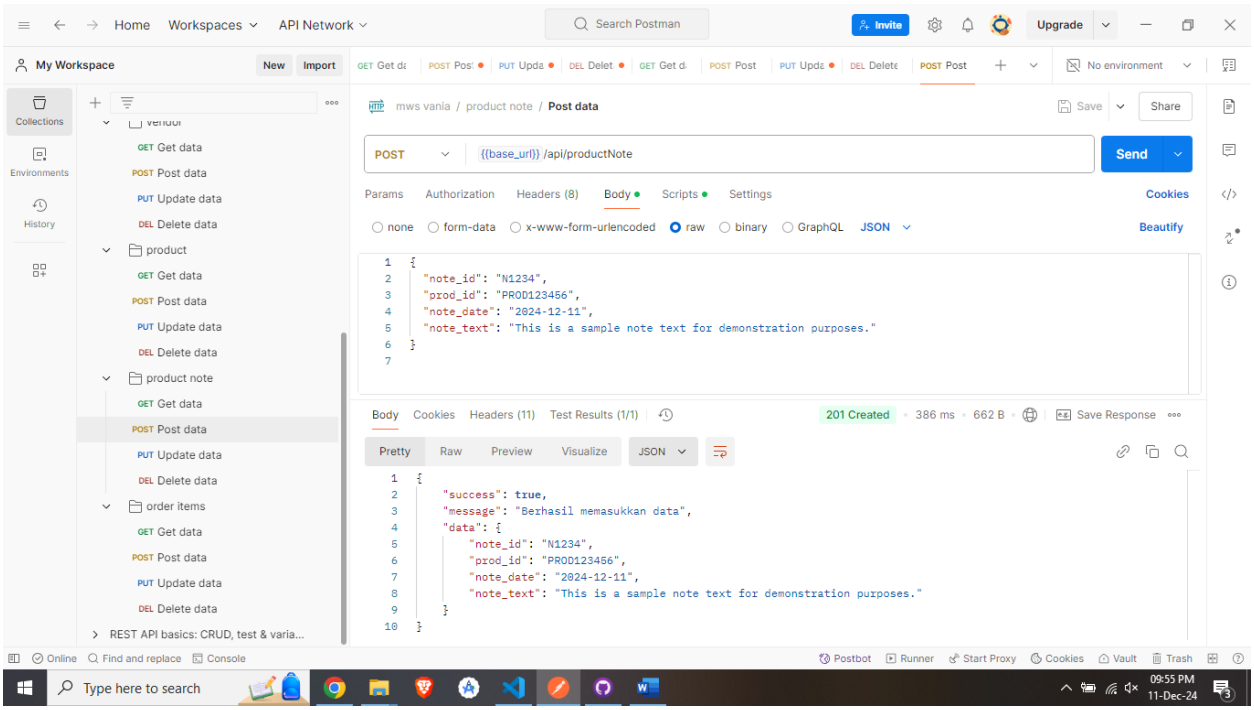


4. Delete

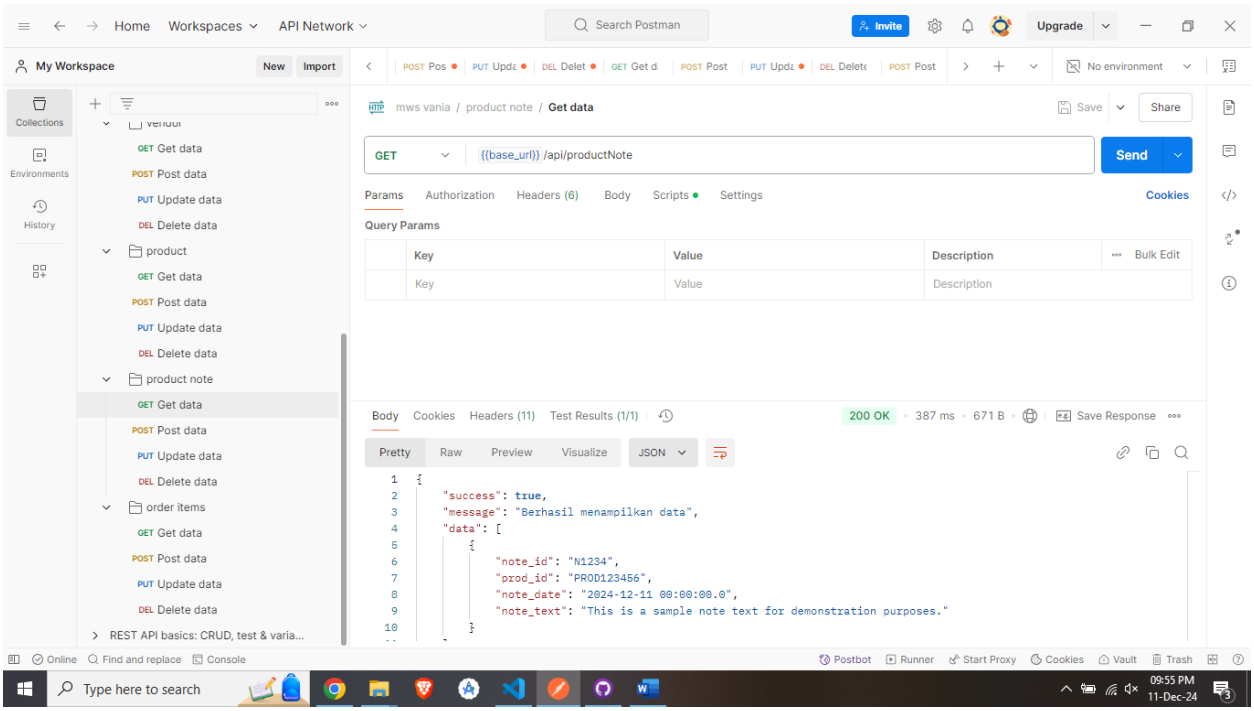


productnotes:

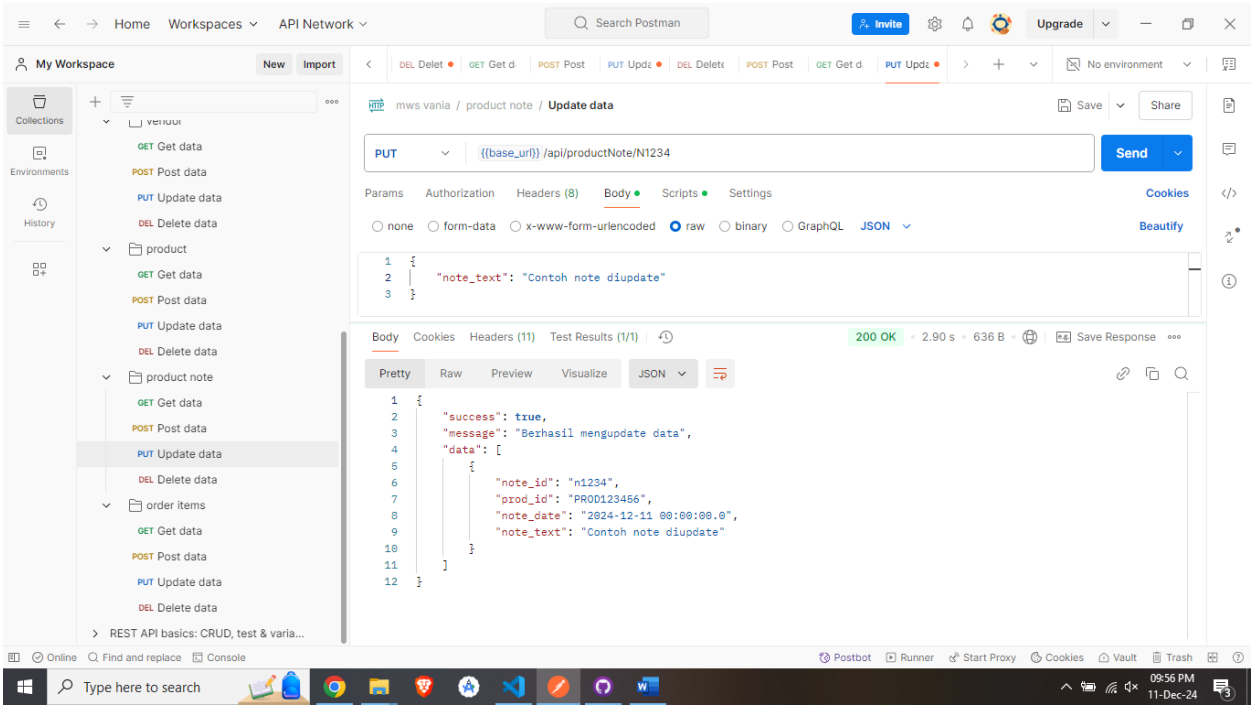
1. Create



2. Read



3. Update



4. Delete