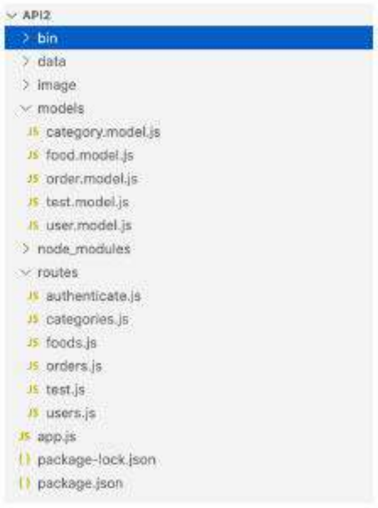
Bài thực hành số 4

Mục tiêu:
Xây dựng Rest API với Node JS và MongoDB



- Bước 1: install nodejs (version mới nhất)
- Bước 2: install express (npm install express) và express-generator (https://expressjs.com/en/starter/generator.html)

và các thư viện như sau:

```
"dependencies": {
  "bcrypt": "^5.0.0",
  "body-parser": "^1.19.0",
  "const": "^1.0.0",
  "cookie-parser": "~1.3.5",
  "express": "^4.17.1",
  "jsonwebtoken": "^8.5.1",
  "mongoose": "^5.11.5",
  "node-datetime": "^2.1.2"
 }
```

- Bước 3: tạo cấu trúc API như hình trên (express no-view)
- Bước 4: Xây dựng app.js

```
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');
var mongoose = require('mongoose');

var users = require('./routes/users');
var foods = require('./routes/foods');
var categories = require('./routes/categories');
var orders = require('./routes/orders');
var users = require('./routes/users');
var test = require('./routes/test');

//khoi tao server voi express framework
var app = express();
//cau hinh server de doc cac thong so dau vao
app.use(express.static('image'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser())
//ket noi csdl mongodb
mongoose.connect('mongodb://localhost:27017/foodDB', {
  useNewUrlParser: true, useUnifiedTopology: true
}).then(() => {
  console.log("Successfully connected to the database");
}).catch(err => {
  console.log('Could not connect to the database. Exiting now...', err);
  process.exit();
});
// dinh nghia endpoint
app.use('/api/v1/users', users);
app.use('/api/foods', foods);
app.use('/api/categories', categories);
app.use('/api/orders', orders);
app.use('/api/users', users);
app.use('/api/test', test);

module.exports = app;
```

- Bước 5: Xây dựng các model:

food.model.js

```
const mongoose = require('mongoose');

const FoodSchema = mongoose.Schema({
  title: String,
  description: String,
  image: String,
  price : String
}, {versionKey: false, collection: 'food'});

module.exports = mongoose.model('Food', FoodSchema);
```

category.model.js

```
const mongoose = require('mongoose');

const CategorySchema = mongoose.Schema({
  title: String,
  image: String,
}, {versionKey: false, collection: 'category'});

module.exports = mongoose.model('Category', CategorySchema);
```

order.model.js

```
const mongoose = require('mongoose');

const OrderSchema = mongoose.Schema({
  username: String,
  address: String,
  createOnDate: String,
  total:String,
  orderDetails : [],
  status:String,
}, {versionKey: false, collection: 'order'});

module.exports = mongoose.model('Order', OrderSchema);
```

- Bước 6: Xây dựng các controller

food.js

```
const Food = require('../models/food.model');
const auth = require('./authenticate');
const express = require('express');
var router = express.Router();


router.get('/', (req, res) => {
   Food.find()
   .then(data => {
      res.send({'food':data});
   })
   .catch(error => {
      res.status(500).send({
         message: error.message
      });
   });
});

router.post('/', (req, res) => {
   const food = new Food({
      title : req.body.title,
      description: req.body.description,
      price : req.body.price,
   });
   food.save()
   .then( data => {
      res.send(data);
   })
   .catch( error => {
      res.status(500).send({
         message: error.message
      });
   });
});

router.put('/:id', (req, res) => {
   Food.findByIdAndUpdate(req.params.id,{
      title : req.body.title,
      description: req.body.description,
      price : req.body.price,
   }, {new:true})
   .then( () => {
      res.send({'message': 'Oke'});
   })
   .catch( error => {
      res.status(500).send({
         message: error.message
      });
   });
});
```

```
router.delete('/:id', (req, res) => {
   Food.findByIdAndRemove(req.params.id)
   .then( () => {
      res.send({'message': 'Oke'});
   })
   .catch( error => {
      res.status(500).send({
         message: error.message
      });
   });
});

module.exports = router;
```

category.js

```
const Categories = require('../models/category.model');
const express = require('express');
var router = express.Router();

router.get('/', (req, res) => {
   Categories.find()
   .then(data => {
      res.send({'categories':data});
   })
   .catch(error => {
      res.status(500).send({
         message: error.message
      });
   });
});

router.get('/:id', (req, res) => {
   Categories.findById(req.params.id)
   .then(data => {
      res.send(data);
   })
   .catch(error => {
      res.status(500).send({
         message: error.message
      });
   });
});

module.exports = router;
```

order.js

```
const Order = require('../models/order.model');
const express = require('express');
var router = express.Router();

var dateTime = require('node-datetime');
var dt = dateTime.create();
var formatted = dt.format('d-m-Y H:M:S');


router.post('/checkout',(req,res) => {
  const order = new Order({
    username : req.body.username,
    createOnDate: formatted,
    status : '0',
    total: '0',
  });
  order.save()
  .then( data => {
    res.send(data);
  })
  .catch( error => {
    res.status(500).send({
      message: error.message
    });
  });
} );
```

```
router.post('/checkout',(req,res) => {
  const order = new Order({
    username : req.body.username,
    createOnDate: formatted,
    status : '0',
    total: '0',
  });
  order.save()
  .then( data => {
    res.send(data);
  })
  .catch( error => {
    res.status(500).send({
      message: error.message
    });
  });
} );
```

```
router.post('/placeorder', (req, res) => {
  Order.findByIdAndUpdate({_id: req.body.id},
    { status: "1", total: req.body.total,$push: {  orderDetails: req.body.orderDetails } }, {new: true})
  .then(data => {
    res.send(data);
  });
});
```

```
router.get('/', (req,res) => {
   Order.find({status : "1"})
   .then(data => {
      res.send({'order':data});
   })
   .catch(error => {
      res.status(500).send({
         message: error.message
      });
   });
})


module.exports = router;
```

Bước 7: Sử dụng Postman để test api trên

Bài thực hành số 5

Mục tiêu:

Kết nối ứng dụng vơi hệ thống API được xây dựng ở bài 4

Thêm các phương thức trong lớp Utilities

```
String url = 'http://192.168.0.100:3000/api/food';

static List<Products> data = [];

Future<List<Products>> getProducts() async{
  var res = await http.get(url);
  if (res.statusCode == 200) {
    var content = res.body;
    print(content.toString());
    var arr = json.decode(content)['food'] as List;
    // for every element of arr map to _fromJson
    // and convert the array to list
    return arr.map((e) => _fromJson(e)).toList();
  }

  return List<Products>();
}

Products _fromJson(Map<String, dynamic> item) {
  return new Products(
      description: item['description'],
      title: item['title'],
      image: item['image'],
      price: double.parse(item['price']));
}
```

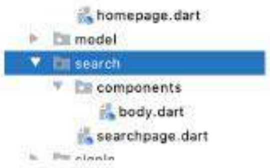
ProductPopular

```
import 'package:flutter/material.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/widgets.dart';
import 'package:flutter_foodnow_app/detail/productpage.dart';
import 'package:flutter_foodnow_app/model/products.dart';
import 'package:flutter_foodnow_app/model/utilities.dart';

class ProductPopular extends StatelessWidget {


  @override
  Widget build(BuildContext context) {
    var products = Utilities().getProducts();
    return Padding(
      padding: const EdgeInsets.all(8.0),
      child: Column(
        mainAxisSize: MainAxisSize.max,
        children: [
          Row(
            children: [
              Expanded(child: Text('Popular Products ', style: TextStyle(
                  fontSize: 18,
              fontWeight: FontWeight.bold,
              color: Colors.green),)),
            Text('See more',
              style: TextStyle(fontSize: 16, color: Colors.lightGreen),),
          ],
        ),
        SizedBox(height: 10,),
        Container(
          child:
          GridView.builder(
            scrollDirection: Axis.vertical,
            shrinkWrap: true,
            primary: false,
            itemCount: products.length,
            gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
              crossAxisCount: 3,
              mainAxisSpacing: 10,
              crossAxisSpacing: 10,
              childAspectRatio: 0.7

            ),
            itemBuilder: (context, index) {
              return ProductItem(
                product: products[index],
              );
            })


        ),

      ],
    ),
  );

 }
}

class ProductItem extends StatelessWidget {
 Products product;

 ProductItem({this.product});

 @override
 Widget build(BuildContext context) {
  if(product.image != null){

  }
  return GestureDetector(
   onTap: () {
    //print(product.id.toString());
    Utilities.data.add(product);
    Navigator.pushNamed(context, ProductPage.routeName,
      arguments: ProductDetailsArguments(product: product));
   },
   child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Image.asset(product.image, fit: BoxFit.fill,),
      Row(
        children: [
          Expanded(child: Text(product.title)),
          Container(
            padding: EdgeInsets.all(2),
            decoration: BoxDecoration(
              border: Border.all(color: Colors.white),
              borderRadius: BorderRadius.circular(2),
              color: Colors.green
            ),
            child: Text(product.price.toString(), style: TextStyle(
              color: Colors.white, fontWeight: FontWeight.bold),)),
        ],
      )
    ],),);}}
```

Bài Thực hành số 6

Mục tiêu : Xây dựng chức năng tìm kiếm

SearchPage



```
class SearchPage extends StatelessWidget {
  static String routeName = "/search_screen";
  @override
  Widget build(BuildContext context) {
    return Body();
  }
}
```

Body

```
import 'package:flutter/material.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter_foodnow_app/homepage/components/fragment/favorite_fragment.dart';
import 'package:flutter_foodnow_app/model/products.dart';
import 'package:flutter_foodnow_app/model/utilities.dart';
import 'package:flutter_tags/flutter_tags.dart';


class Body extends StatefulWidget {

  List<Products> dataProduct = new List<Products>();
// Body({this.dataProduct});
  @override
  _BodyState createState() => _BodyState();
}

class _BodyState extends State<Body> {

  final GlobalKey<ScaffoldState> _scaffoldKey = new GlobalKey<ScaffoldState>();

  List<String> _tags=[];
  List<Products> products = Products.init();
  List<Products> productsResult = new List<Products>();
```

```
TextEditingController textEditingController;

@override
void initState() {
  // TODO: implement initState
  super.initState();
  _tags.addAll(['food', 'categories','bread']);
  textEditingController = new TextEditingController();
}


Widget buildTag(BuildContext context){
  return Container(
    width: MediaQuery.of(context).size.width,
    color: Colors.white,
    child: Column(
      mainAxisSize: MainAxisSize.min,
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text('Recommend'),
        SizedBox(height: 10,),
        Tags(
          itemCount: _tags.length,
          itemBuilder: (index){
            // print(index.toString());
            return ItemTags(
              index: index,
              title: _tags[index],
              onPressed: (item)  {
                setState(() {
                  widget.dataProduct.clear();
                  widget.dataProduct.addAll(Utilities().find(item.title));
                });

              },
            );
          },
        ),
      ],
    ),
  );
}


@override
Widget build(BuildContext context) {
  return Scaffold(
    key: _scaffoldKey,
    appBar: AppBar(
      automaticallyImplyLeading: false,
      title: buildRow(),
    ),
    body: buildContainer(context)
  );
}

Widget buildRow() {
  return Row(
     children: [
      Expanded(
       child: TextField(
         controller: textEditingController,
         decoration: InputDecoration(
            filled: true,
            fillColor: Colors.white,
            hintText: "Search product",
            prefixIcon: Icon(Icons.search)
         ),
         onChanged: (value){
            setState(() {
             if(value.isEmpty){
              widget.dataProduct = new List<Products>();
              return;
             }
             widget.dataProduct.clear();
             widget.dataProduct.addAll(Utilities().find(value));
            });
         },
        ),
      ),

     ],
    );
 }

 Widget buildContainer(BuildContext context) {
  return Container(
    width: MediaQuery.of(context).size.width,
    height: MediaQuery.of(context).size.height,
    child: Column(
     mainAxisSize: MainAxisSize.max,
     children: [
      buildTag(context),
      if ( widget.dataProduct.length == 0)
       Expanded(child: Center(
          child: Text('No item')))
      else Expanded(child: ListView.builder(
         itemCount:  widget.dataProduct.length ,
         itemBuilder: (context, index){
          return ProductItemList(product:   widget.dataProduct[index],);
         }))

     ],
    ),

  );
 }
}
```

Bài thực hành số 7

Mục tiêu : Xây dựng chức năng sao lưu