УНИВЕРЗИТЕТ ,,СВ. КИРИЛ И МЕТОДИЈ” – СКОПЈЕ

ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Напреден веб дизајн

Spoon – апликација за пребарување на рецепти по состојки

Изработиле:

Луна Делевска 211116

Марија Зографска 211111

Ментор:

проф. д-р Бобан Јоксимоски

Датум: 19.09.2024

Вовед

Spoon е веб апликација развиена како дел од курсот за Напреден веб дизајн, дизајнирана да го олесни планирањето на оброци, помагајќи им на корисниците да најдат рецепти врз основа на состојките што ги имаат. Апликацијата користи современи веб технологии, при што Spring Boot е користен за backend, а React Bootstrap за frontend, овозможувајќи непречено и ефикасно корисничко искуство.

Главната функционалност на Spoon е пребарувањето на рецепти според состојки. Корисниците едноставно ги внесуваат состојките што ги имаат дома, а Spoon брзо враќа листа на рецепти што можат да се направат со тие состојки. Без разлика дали се обидувате да ги искористите преостанатите намирници или експериментирате со нови јадења, Spoon го поедноставува процесот на наоѓање соодветни рецепти, заштедува време и помага во намалување на фрлањето храна.

На backend-от, Spoon користи Spring Boot, моќен фрејмворк за градење робусни и скалабилни веб апликации. Ова овозможува апликацијата ефикасно да ги обработува податоците, да ги повлекува рецептите од сеопфатна база на податоци и да обезбеди точни резултати на корисниците во реално време. Backend-от исто така управува со корисничките барања, комуницира со извори на податоци за рецепти и обезбедува непречено функционирање на главните функции на апликацијата.

Frontend-от, изграден со React Bootstrap, обезбедува кориснички пријателски и одговорен интерфејс. Со чист дизајн и интуитивен распоред, корисниците лесно можат да се движат низ апликацијата, да внесуваат состојки и да ги прегледуваат предложените рецепти. React Bootstrap гарантира дека апликацијата работи непречено на сите уреди, нудејќи конзистентно искуство без разлика дали корисниците пристапуваат од десктоп, таблет или мобилен телефон.

Spoon има за цел да обезбеди едноставно, но ефективно решение за секојдневното прашање: „Што можам да зготвам со состојките што ги имам?“ Со комбинирање на моќна технологија и фокус на корисничката погодност, Spoon го подобрува подготвувањето на оброци и инспирира кулинарска креативност.

Frontend

Почетна страница (Home Page)

Home Page или почетната страна е првата страна која корисникот ќе ја види кога ќе ја отвори апликацијата. Има за задача да го информира корисникот за функционалностите на апликацијата, начинот на употреба и идејата зад Спун. . Користењето на React-

Bootstrap библиотеката овозможува респонзивен дизајн со елементи како што се Navbar, Button, Container и Carousel, обезбедувајќи модерен и кориснички пријателски интерфејс. Еве преглед на структурата и функционалноста на кодот:

- **Распоред и структура**

HomePage започнува со увезување на основни компоненти од React, Bootstrap и react-router-dom. Користењето на Bootstrap осигурува дека страната е респонзивна и одржува конзистентен дизајн на различни големини на екрани. Ова исто така го поедноставува процесот на развој со обезбедување готови компоненти.

Страницата е поделена на три главни делови:

1. Навигациска лента на врвот
2. Содржински дел во средината
3. Footer на дното.

- **Навигациска лента**

Navbar служи како горна навигација на апликацијата. Таа содржи Spoon лого кое води до почетната страница и три навигациски линкови: Home, Search и Premium. Овие линкови овозможуваат лесна навигација низ апликацијата користејќи useNavigate hook од react-router-dom. Логото исто така функционира како копче кое при клик води назад на почетната страница. Premium копчето е стилизирано со variant="outline-light" за чист и елегантен изглед.

- **Позадина и Overlay**

Првиот дел под навигациската лента вклучува целосна позадина со overlay текст, кој ја промовира основната порака на апликацијата: "Готви со она што го имаш!". Во овој дел има и "Get Started" копче кое го носи корисникот на страната за пребарување, овозможувајќи им на корисниците да истражуваат рецепти со внесување состојки.

- **Содржински дел**

Овој дел дава краток вовед за Spoon, нагласувајќи ја нејзината цел да го отклучи потенцијалот на кујнските залихи на корисниците преку генерирање персонализирани рецепти врз основа на достапните состојки. Содржи колони користејќи го Bootstrap системот на мрежа, кои прикажуваат слики и описи за тоа како функционира апликацијата. Секој ред содржи икона и опис на клучните функции на Spoon апликацијата.

Carousel под текстот обезбедува визуелна претстава на вкусни јадења кои можат да се направат со обични состојки како јајца, тестенини и гулаш. Ротира слики на секои 2 секунди, додавајќи динамична интерактивност на почетната страница.

- **Footer**

На дното на страницата е footer кој го содржи логото на Spoon, известување за авторски права и дополнителни навигациски линкови. Исто така, има икони за социјални медиуми за Facebook, Twitter и Instagram, промовирајќи интеракција со поширокатa онлајн заедница на Spoon.

Оваа почетна страница ефикасно комбинира дизајн и функционалност. Со користење на React-Bootstrap, кодот креира професионален и чист UI, додека одржува лесна навигација и интерактивност преку useNavigate hook. Комбинацијата на визуелни елементи како carousel, overlay и икони го зголемува ангажманот на корисниците, правејќи ја апликацијата функционална и визуелно привлечна.

<u>Страница за пребарување (Search Page)</u>

Страницата за пребарување или Search Page е клучна за апликацијата Спун бидејќи преку нејзе корисникот ги внесува состојките и добива рецепти од истите. На страницата се користи React-Bootstrap библиотека за да вклучи респонзивен дизајн со елементи како што се Navbar, Button, Container и Card, обезбедувајќи модерен и кориснички пријатен интерфејс. Еве преглед на структурата и функционалноста на овој код:

- **Распоред и Структура**

Search компонентот започнува со импорт на основните компоненти од React, Bootstrap, и react-router-dom. Употребата на Bootstrap осигурува дека страната е респонзивна и одржува конзистентен дизајн низ различни големини на екрани. Ова, исто така, го поедноставува процесот на развој, со обезбедување на готови за употреба компоненти.

Страната е поделена на три главни секции:

1. Навигациска лента на врвот
2. Содржинска секција во средина
3. Footer на дното

- **Навигациска лента**

Navbar е наменета како главна навигација за апликацијата. Таа го содржи логото на Spoon, кое го враќа корисникот на почетната страна, и три навигациски линкови: Home, Search, и Premium. Овие линкови овозможуваат мазна навигација низ апликацијата со помош на useNavigate hook од react-router-dom. Логото исто така функционира како копче кое го враќа корисникот на почетната страна. Premium копчето е стилизирано со variant="outline-light", што дава чист и елегантен изглед.

- **Позадина и Overlay**

Првиот дел под Navbar вклучува позадина на цел екран со overlay текст, промовирајќи ја основната порака на апликацијата: „Искористете ги вашите состојки!“ Оваа секција исто така содржи поле за внес на состојки, каде корисниците можат да пребаруваат рецепти. Копчето "Search" ги иницира барањата за рецепти и ги прикажува резултатите врз основа на внесените состојки.

- **Содржински дел**

Оваа секција вклучува краток вовед во Spoon, потенцирајќи ја неговата цел за креирање персонализирани рецепти според состојките што корисниците ги имаат. Користи Bootstrap grid систем за да подели содржина на колони, кои вклучуваат слики и текстуални описи за тоа како работи апликацијата. Покрај тоа, резултатите од пребарувањето на рецепти се прикажуваат динамично кога корисникот ќе внесе состојки.

- **Footer**

На дното на страната, footer-от е едноставен, но информативен, содржи логото и линкови до различни делови од апликацијата, вклучувајќи социјални мрежи како Facebook, Twitter и Instagram, кои се прикажани како икони.

Страната за пребарување е суштинска за добрата работа на апликацијата. Преку интерактивниот карактер на страницата, таа ги задоволува стандардите на добра прегледност и привлечен изглед на веб страната.

<u>Страница за рецепт (RecipeDetail)</u>

Страницата за рецепт дава информации за состојките во рецептот и начинот на подготовка. Апликацијата користи React-Bootstrap библиотека за да создаде модерен и прилагодлив дизајн, обезбедувајќи беспрекорно корисничко искуство преку различни уреди. Овој код демонстрира сложен пристап кон прикажувањето на рецепти, интегрирајќи навигација, детални информации за рецепти и кориснички прилагодени компоненти. Еве анализа на структурата и функционалноста на кодот:

- **Распоред и структура**

Компонентата RecipeDetail користи управување со состојби на React (useState) и капацитети за рутирање (useNavigate) за динамички да преземе и прикаже детали за рецепт врз основа на ID-то на рецептот од URL параметрите. Компонентите на Bootstrap, вклучувајќи Navbar, Container, Button, Row, Col, и Card, придонесуваат за прилагодлив и визуелно привлекателен изглед. Страницата е организирана во три главни секции:

1. Навигациска лента на врвот

2. Секција за содржина во средина
3. Footer

- **Навигациска лента**

Навигациската бар се наоѓа на врвот на страницата и им обезбедува на корисниците лесен пристап до различни делови на апликацијата. Содржи лого на Spoon и три опции за навигација: Home, Search, и Premium. Хуком за навигација (useNavigate) овозможува глатки транзиции помеѓу овие страници, подобрувајќи го корисничкото искуство. Копчето Premium е стилизирано со variant="outline-light", што му дава истакнат и привлечен изглед, што ги охрабрува корисниците да истражуваат премиум содржини.

- **Функција за детали за рецепти**

Основната функција на оваа страница е деталниот преглед на еден рецепт. Хукот useEffect презема податоци за рецептот од API врз основа на ID параметарот и ја ажурира состојбата. Информациите за рецептот потоа се прикажуваат во добро организиран изглед. Горната секција ја содржи целосната ширина слика и наслов на рецептот, што обезбедува привлекувачко воведување. Под оваа, сликата на рецептот и состојките се прикажани едностранo, со состојките наведени во Card компонент за јасност. Инструкциите се прикажани во посебна секција за содржина, нудејќи упатства чекор по чекор.

- **Секција за содржина**

Секцијата за содржина е поделена во два столба. Првиот столб го прикажува сликата на рецептот, додека вториот столб го претставува компонентот Card кој ги детализира состојките. Овој распоред обезбедува дека корисниците лесно можат да ја видат и визуелната и текстуалната информација за рецептот. Додатните компоненти Card на дното на секцијата за содржина ги содржат препораките од различни готвачи, додавајќи кредибилитет и професионален увид во квалитетот на рецептот.

- **Footer**

Подножјето обезбедува важни информации и навигациски врски, вклучувајќи брендирање, контакт детали и икони за социјални медиуми. Оваа секција е дизајнирана да одржи професионален изглед, додека обезбедува корисниците со дополнителни информации и можност за поврзување со апликацијата преку социјални медиуми платформи.

Компонентата RecipeDetail е ефективно комбинирање на навигациски елементи, детално прикажување на рецепти и професионални препораки за создавање кориснички пријателско искуство. Користењето на компоненти на React-Bootstrap обезбедува

прилагодлив и привлекателен изглед, додека интеграцијата на динамички податоци и управување со состојби овозможува глатко и интерактивно корисничко искуство.

Страница за премиум членство (Premium page)

Страницата за премиум членство му овозможува на корисникот да се преплати на еден од понудените пакети со цел да му понуди помош при организирање на прослави. Дадениот React код претставува страница за цени на веб апликација која нуди различни опции за планирање оброци. Дизајнот акцентира чиста, модерна естетика користејќи React-Bootstrap за одзивен распоред и безпрекорно навигација. Погледнете ја прегледот на тоа како е структуриран кодот и неговата функционалност:

- **Распоред и Структура**

Компонентата Pricing е дизајнирана со фокус на корисничкото искуство и визуелната привлекателност. Таа користи React хукс (useNavigate) за навигација и компоненти на React-Bootstrap за распоред и стилизирање. Страницата е поделена на неколку различни секции:

1. Навигациска лента
2. Секции за Содржина
3. Footer

- **Навигациска лента**

Компонентата Navbar, лоцирана на врвот на страницата, обезбедува стилско и функционално искуство за навигација. Вклучува лого кое, кога ќе се кликне, насочува корисниците кон почетната страница. Navbar-от исто така вклучува линкови за навигација кон почетната страница, страницата за пребарување, делот „за нас“ и страницата за контакт, сите стилизирани да се истакнат на темната позадина. Дополнително, копчето „Premium“ е истакнато, поттикнувајќи ги корисниците да ги истражат премиум опциите. Ова копче е стилизирано со variant="outline-light" за да се осигура дека привлекува внимание и добро се вклопува со целокупниот дизајн.

- **Секции за Содржина**

Страницата за цени е поделена на две главни секции за содржина:

Секција со Прекриен Текст: Поставена на врвот, оваа секција обезбедува привлечна воведна порака за достапните планови за плаќање. Прекриениот текст е стилизиран да биде смел и истакнат, користејќи големи фонт големини и силна типографија за веднаш да ја привлече вниманието на корисниците.

Секција со Картички за Планови: Оваа секција претставува три различни опции за планирање оброци: Small Party, Standard Party и Grande Party. Секоја опција е прикажана во Card компонента, која вклучува детално опишување, цена и копче за акција. Картичките се стилизирани со граници, сенки и облоги за да создадат визуелно привлечна и кориснички пријателска презентација. Планот Small Party е погоден за интимни собири, планот Standard Party за средни групи, а планот Grande Party за поголеми настани. Секоја картичка е дизајнирана да ги истакне карактеристиките и придобивките на соодветниот план, што им овозможува на корисниците лесно да ги споредат и изберат најдобрата опција за нивните потреби.

- **Footer**

Компонентата Footer на дното на страницата обезбедува дополнителни навигациски и брендирани елементи. Вклучува помала верзија на логото и информации за авторски права, обезбедувајќи дека корисниците имаат пристап до брендирањето и кога ќе стигнат до крајот на страницата. Подножјето исто така вклучува линкови до различни секции како што се „Featured“, „About“ и „Image License Info“, како и икони за социјални медиуми за Facebook, Twitter и Instagram. Овој распоред не само што ја зајакнува идентификацијата на брендот туку и обезбедува брз пристап до важни информации и канали за социјални медиуми.

Општо, страницата за цени е дизајнирана да понуди јасна и привлечна презентација на опции за планирање оброци. Комбинацијата на добро структурирана навигациска трака, информативни секции за содржина и функционално подножје создава кохерентно корисничко искуство. Користењето на React-Bootstrap осигурува одзивен дизајн, правејќи ја страницата визуелно привлечна на различни уреди. Овој распоред ефективно комуницира достапните опции и поттикнува корисниците да ги истражат премиум плановите, подобрувајќи и леснотијата на употреба и естетската привлечност.

Backend

Користени технологии

Апликацијата е изградена користејќи Spring Boot за backend и React за frontend, формирајќи моќно и ефикасно full-stack решение.

1. **Spring Boot**
    - Java Framework: Spring Boot е рамка за изградба на веб апликации во Java. Тој го поедноставува setup-от и развојниот процес, обезбедувајќи претходно

конфигурирани поставки и функции, што им овозможува на програмерите да се фокусираат на пишувањето код, а не на конфигурирање.

- RESTful APIs: Со вградената поддршка за создавање на RESTful веб услуги, Spring Boot овозможува беспрекорна комуникација помеѓу клиентот и серверот. Тој се придржува до принципите на REST, овозможувајќи на frontend-от да взаимодејствува со backend услугите преку стандардизирани HTTP методи.

2. **React**

   - JavaScript Library: React е широко користена библиотека за изградба на кориснички интерфејси, особено за едностранични апликации (SPAs) каде динамичното корисничко искуство е клучно. Неговата архитектура базирана на компоненти овозможува развој на повторно употребливи UI компоненти, што ја подобрува одржливоста и скалабилноста.
   - Декларативна Синтакса: Декларативниот пристап на React ја прави лесна дизајнирањето на интерактивни UI, овозможувајќи на програмерите да специфицираат како UI треба да изгледа врз основа на моменталната состојба на апликацијата.

Слоевита Архитектура Апликацијата следи слоевита архитектура која се состои од три основни слоеви: презентациски слој, сервисен слој и слој за персистенција. Овој дизајн промовира поделба на задачите, правејќи ја апликацијата полесна за управување и проширување.

1. **Презентациски Слој(Presentation Layer):** Ова е местото каде корисниците взаимодејствуваат со апликацијата. Изградена со React, вклучува различни компоненти како RecipeList, RecipeDetail и рутирање управувано од react-router-dom. Овој слој е одговорен за рендерирање на UI и прифаќање на кориснички внесувања.
2. **Сервисен Слој(Service Layer):** Овој слој содржи деловна логика и сервисни класи. Тој управува со протокот на податоци помеѓу презентацискиот слој и Persistence слојот. Интерфејсот RecipeService и неговата имплементација (RecipeServiceImpl) се тука. Сервисниот слој функционира како посредник, осигурувајќи дека презентациски слој не се поврзува директно со изворите на податоци.
3. **Persistence Слој**: Овој слој е одговорен за складирање и добивање на податоци. Во оваа апликација, тој комуницира со надворешни APIs, конкретно Spoonacular API, за да ги добие податоците за рецепти врз основа на корисничкиот input. Тој ја апстрахира сложеноста на добивањето податоци, овозможувајќи на сервисниот слој да се фокусира на деловната логика.

Преглед на Код

Основните модели, услуги и контролери формираат основа на апликацијата. Подолу следи детална анализа на секој компонент, вклучувајќи кодни фрагменти за илустрација на нивната функционалност.

Модели

1. **Recipe Model**:
    - Цел: Претставува рецепт со атрибути за наслов, ID, состојки, URL на слика и упатства за готвење. Овој модел служи како план за создавање на објекти за рецепти и ги енкапсулира податоците поврзани со рецепт.
    - Методи: Гетери и сетери за секој атрибут овозможуваат енкапсулација, овозможувајќи контролирано пристапување до својствата на класата Recipe.
2. **Ingredient Model**:
    - Цел: Дефинира индивидуални состојки со нивното име, количина и единица за мерење. Овој модел ѝ овозможува на апликацијата да претстави сложени податоци за состојки на структурирано ниво.
    - Методи: Слично на Recipe моделот, содржи гетери и сетери, осигурувајќи дека податоците остануваат енкапсулирани и можат лесно да се манипулираат.

Сервисен Слој

1. **RecipeService Интерфејс**:
    - Цел: Специфицира рамка за сервисниот слој, специфицирајќи методи за добивање рецепти и мапирање податоци од надворешниот API. Овој интерфејс промовира loose coupling, овозможувајќи различни имплементации на сервисниот слој.

```
7 usages  1 implementation  unknown
public interface RecipeService {
    2 usages  1 implementation  unknown
    List<Recipe> getRecipes(String ingredients);

    2 usages  1 implementation  unknown
    Map<String, Object> getRecipeInfo(int id);

    1 usage  1 implementation  unknown
    Recipe mapToRecipeModel(LinkedHashMap<String, Object> recipeData);

    1 usage  1 implementation  unknown
    List<Ingredient> mapIngredients(List<LinkedHashMap<String, Object>> ingredientsData);
}
```

2. **RecipeServiceImpl Имплементација**:
    - Цел: Имплементира логика за добивање рецепти од Spoonacular API врз основа на внесот од корисникот. RestTemplate се користи за правење HTTP барања и

управување со одговори. Имплементацијата исто така содржи управување со грешки за да се осигура стабилност.

- Мапирање на рецепти:
  - Цел: Претвора одговор од API во структуриран Recipe објект, правејќи го полесно за frontend-от да ги консумира. Овој метод ја апстрахира сложеноста на управувањето со податоци од API.
- Мапирање на состојки:
  - Цел: Мапира список на состојки од одговорот од API во список на Ingredient објекти. Овој метод осигурува дека сите податоци за состојки се структурирани конзистентно за лесна консумација со frontend-от.

Преглед на Класата

Оваа класа е имплементација на сервис во Spring Boot апликација дизајнирана да комуницира со Spoonacular API за да добие рецепти врз основа на состојките внесени од корисникот. Таа дефинира методи за добивање рецепти и детални информации за специфични рецепти.

Клучни Компоненти

1. **Анотации и Конфигурација**:
   - @Service: Оваа анотација ја маркира класата како сервисен компонент во Spring, овозможувајќи автоматско откривање за време на скенирање на компоненти.
   - @Value("${spoonacular.api.key}"): Оваа анотација инјектира API клуч од апликациските својства, овозможувајќи безбедно управување со чувствителни информации.
2. **Константи**:
   - API_KEY: Тврдокодиран API клуч.
   - URL: Основен URL за добивање рецепти по состојки.
   - RECIPE_INFO_URL: URL шаблон за добивање детални информации за конкретен рецепт.
3. **Метод: getRecipes(String ingredients)**

```
2 usages  unknown
public List<Recipe> getRecipes(String ingredients) {
    List<Recipe> recipeList = new ArrayList<>();
    RestTemplate restTemplate = new RestTemplate();
    String url = URL + ingredients;

    ResponseEntity<List> response = restTemplate.getForEntity(url, List.class);
    List<LinkedHashMap<String, Object>> recipes = response.getBody();

    if (recipes != null) {
        for (LinkedHashMap<String, Object> recipeData : recipes) {
            Recipe recipe = mapToRecipeModel(recipeData);
            if (recipe != null) {
                recipeList.add(recipe);
            }
        }
    }
    return recipeList;
}
```

- Цел: Добива рецепти врз основа на обезбедени состојки.
- Процес:
  - Конструира целосен URL користејќи ги внесените состојки.
  - Користи RestTemplate за правење HTTP GET барање до Spoonacular API.
  - Мапира одговор (список на рецепти) на Recipe модел објекти.
  - Враќа список на Recipe објекти.
- Придонес: Овој метод директно ја исполнува основната функционалност на апликацијата, добивајќи рецепти врз основа на внесите на корисникот.

4. **Метод: getRecipeInfo(int id)**

```java
public Map<String, Object> getRecipeInfo(int id) {
    RestTemplate restTemplate = new RestTemplate();
    String url = RECIPE_INFO_URL.replace( target: "{id}", String.valueOf(id)).replace( target: "{apiKey}", apiKey);

    ResponseEntity<String> responseEntity = restTemplate.getForEntity(url, String.class);
    String responseBody = responseEntity.getBody();

    ObjectMapper objectMapper = new ObjectMapper();
    Map<String, Object> recipeInfoMap = new HashMap<>();
    try {
        JsonNode rootNode = objectMapper.readTree(responseBody);
        String title = rootNode.path( s: "title").asText();
        String instructions = rootNode.path( s: "instructions").asText();
        String imageUrl = rootNode.path( s: "image").asText();
        JsonNode ingredientsNode = rootNode.path( s: "extendedIngredients");
        List<Ingredient> ingredientsList = new ArrayList<>();
        for (JsonNode ingredientNode : ingredientsNode) {
            String name = ingredientNode.path( s: "name").asText();
            double amount = ingredientNode.path( s: "amount").asDouble();
            String unit = ingredientNode.path( s: "unit").asText();
            ingredientsList.add(new Ingredient(name, amount, unit));
        }
        recipeInfoMap.put("title", title);
        recipeInfoMap.put("instructions", instructions);
        recipeInfoMap.put("ingredients", ingredientsList);
        recipeInfoMap.put("imageUrl", imageUrl);
        return recipeInfoMap;
    } catch (IOException e) {
        e.printStackTrace();
        return Collections.singletonMap("error", "Error occurred while fetching recipe info");
    }
}
```

- **Цел**: Добива детални информации за специфичен рецепт користејќи го неговиот ID.
- **Процес**:
  - Конструира URL користејќи го ID на рецептот.
  - Прави GET барање и обработува JSON одговор користејќи ObjectMapper за парсирање на податоците во мапа.
  - Извлекува наслов, упатства, URL на слика и список на состојки.
  - Враќа мапа која содржи детали за рецептот.
- **Придонес**: Овој метод им овозможува на корисниците да гледаат повеќе информации за специфичен рецепт, подобрувајќи го корисничкото искуство.

5. **Метод: mapToRecipeModel(LinkedHashMap<String, Object> recipeData)**

```java
1 usage   unknown *
public Recipe mapToRecipeModel(LinkedHashMap<String, Object> recipeData) {
    int id = (int) recipeData.get("id");
    String title = (String) recipeData.get("title");
    String imageUrl = (String) recipeData.get("image");
    List<Ingredient> ingredients = mapIngredients((
            List<LinkedHashMap<String, Object>>) recipeData.get("usedIngredients"));

    return new Recipe(title, id, ingredients, imageUrl);
}
```

- **Цел**: Претвора податоци за рецепт од API во структуриран Recipe објект.
- **Процес**:
  - Извлекува ID на рецептот, наслов и URL на слика.

        - Повикува mapIngredients за да ги претвори состојките во Ingredient објекти.
    - **Придонес**: Овој метод помага да се одржи чиста одвоеност помеѓу податоци и модели на апликацијата.
6. **Метод: mapIngredients(List<LinkedHashMap<String, Object>> ingredientsData)**

```
1 usage   unknown
public List<Ingredient> mapIngredients(List<LinkedHashMap<String, Object>> ingredientsData) {
    List<Ingredient> ingredients = new ArrayList<>();
    if (ingredientsData != null) {
        for (LinkedHashMap<String, Object> ingredientData : ingredientsData) {
            String name = (String) ingredientData.get("name");
            double amount = (double) ingredientData.get("amount");
            String unit = (String) ingredientData.get("unit");
            ingredients.add(new Ingredient(name, amount, unit));
        }
    }
    return ingredients;
}

}
```

    - **Цел**: Претвора список на податоци за состојки во список на Ingredient објекти.
    - **Процес**: Итерира преку податоците за состојки, извлекувајќи име, количина и единица за мерење за секоја состојка.
    - **Придонес**: Овој метод осигурува дека податоците за состојки се правилно структурирани за употреба во апликацијата.

Вкупен Придонес на Апликацијата

- **Корисничка Интеракција**: Методот getRecipes им овозможува на корисниците да внесуваат состојки и да добиваат релевантни рецепти, што е основната функционалност на вашата апликација.
- **Управување со Податоци**: Со мапирање на одговорите од API на модели на апликацијата, сервисот одржува јасна и управлива структура на податоци, што го олеснува работењето во рамките на апликацијата.
- **Управување со Грешки**: Иако првично, сервисот вклучува управување со грешки во getRecipeInfo, што е клучно за стабилна апликација.

Контролери

1. **RecipeRestController**:
    - **Цел**: RESTful API за добивање рецепти во JSON формат. Ова му овозможува на React фронтендот да прави API повици за добивање податоци без потреба од освежување на страницата.

- **CORS Конфигурација**: Анотацијата @CrossOrigin дозволува фронтендот, кој работи на различна порта, да пристапува до API на backend-от, олеснувајќи развој и тестирање.

```java
LunaDelevska
@RestController
@RequestMapping("/api/recipes")
@CrossOrigin(origins = "http://localhost:3000")
public class RecipeRestController {

    3 usages
    private final RecipeService recipeService;

    LunaDelevska
    @Autowired
    public RecipeRestController(RecipeService recipeService) { this.recipeService = recipeService; }

    LunaDelevska
    @GetMapping
    public ResponseEntity<List<Recipe>> getRecipes(@RequestParam(required = false) String ingredients) {
        List<Recipe> recipes = recipeService.getRecipes(ingredients);
        return ResponseEntity.ok(recipes);
    }

    LunaDelevska
    @GetMapping("/{id}/info")
    public ResponseEntity<Map<String, Object>> getRecipeInfo(@PathVariable int id) {
        Map<String, Object> recipeInfo = recipeService.getRecipeInfo(id);
        return ResponseEntity.ok(recipeInfo);
    }
}
```

Преглед

Оваа класа е RESTful контролер во Spring Boot апликација која обработува барања поврзани со рецепти. Користи анотации за да дефинира точки на пристап и управува со зависности, што ја прави клучен компонент за функционалноста за пребарување рецепти.

Преглед на кодот

1. **Анотации**:
    - **@RestController**: Означува дека оваа класа е контролер каде секој метод враќа доменски објект наместо поглед. Ги комбинира @Controller и @ResponseBody.
    - **@RequestMapping("/api/recipes")**: Поставува основен URL за сите точки на пристап во овој контролер на /api/recipes.
    - **@CrossOrigin(origins = "http://localhost:3000")**: Дозволува крос-оригин барања од одредениот извор, што е корисно ако фронтендот (на пр. React апликација) работи на различна порта за време на развојот.

2. **Зависности**:
    - **private final RecipeService recipeService**: Ова ја декларира зависноста од класата RecipeService, која содржи деловна логика за добивање рецепти.
    - **@Autowired**: Оваа анотација се користи во конструкторот за инјектирање на RecipeService бинот кога контролерот се создава.
3. **Точки на Пристап**:
    - **@GetMapping (за пребарување рецепти)**:
        - **Патека**: /api/recipes
        - **Метод**: Обработува GET барања за добивање список на рецепти врз основа на состојки.
        - **Параметри**: **@RequestParam(required = false) String ingredients**: Ова му овозможува на клиентот да предаде состојки како параметар. Ако не се обезбедени состојки, ќе биде null.
        - **Одговор**: Го повикува recipeService.getRecipes(ingredients) за да ги добие рецептите и ги враќа обвиткани во ResponseEntity со статус 200 OK.
    - **@GetMapping("/{id}/info")**:
        - **Патека**: /api/recipes/{id}/info
        - **Метод**: Обработува GET барања за добивање детални информации за специфичен рецепт според неговиот ID.
        - **Параметри**: **@PathVariable int id**: Ова извлекува ID на рецептот од URL-то.
        - **Одговор**: Го повикува recipeService.getRecipeInfo(id) за да ги добие деталните информации и ги враќа на сличен начин како и претходниот метод.

Цел и Придонес на Апликацијата

- **Скалабилност**: RESTful дизајнот овозможува лесно да додавате нови точки на пристап во иднина (на пр. додавање, ажурирање или бришење рецепти) без да влијае на постоечката функционалност.
- **Поддршка за Крос-Оргин**: Анотацијата @CrossOrigin му дозволува на фронтенд апликацијата да прави барања до овој API, олеснувајќи развој и тестирање во различни средини.
- **Управување со Грешки и Одговори**: Со користење на ResponseEntity, можете поефективно да управувате со HTTP статус кодовите и хедерите, што е корисно за управување со грешки и обезбедување дополнителни информации до клиентите.

Поврзување помеѓу Backend и React Апликацијата

```
5+ usages  LunaDelevska +1
const App = () => {
    return (
        <Router>
            <Routes>
                <Route path="/" element={<Home />} /> {/* Default home page */}
                <Route path="/search" element={<Search />} /> {/* Search page */}
                <Route path="/recipes/:id/info" element={<RecipeDetail />} />
                <Route path="/pricing" element={<Pricing />} />
            </Routes>
        </Router>
    );
};
4 usages  LunaDelevska
export default App;
```

React фронтендот комуницира со Spring Boot backend преку RESTful API, што овозможува флексибилен и ефикасен начин на размена на податоци. Оваа врска се воспоставува преку HTTP барања кои овозможуваат беспрекорен проток на податоци.

Како Фронтендот се Поврзува со Backend-от

1. **Обработка на Внес на Состојки**:
    - Корисниците внесуваат состојки во текстуално поле. Состојката ingredients ја држи оваа информација, која се ажурира преку handleInputChange.
2. **Функционалност за Пребарување**:
    - Кога корисникот кликне на копчето за пребарување (<Button>), се активира функцијата handleSearchClick, која повикува fetchRecipes.
3. **Добивање Податоци**:
    - Функцијата fetchRecipes користи fetch API за да направи GET барање до Spring Boot backend:

        fetch(`/api/recipes?ingredients=${ingredients}`)

    - Оваа конструкција создава URL кој ги испраќа состојките како параметар до точката на пристап што ја дефинираме во Spring Boot апликацијата.
4. **Обработка на Backend**:
    - Spring Boot апликацијата ја прима нарачката на /api/recipes точката на пристап.

- Таа ги обработува состојките, ги добива одговарачките рецепти од базата на податоци или друг извор, и испраќа JSON одговор назад.

5. **Обработка на Одговорот**:
   - Ако одговорот е успешен, JSON податоците (низа на рецепти) се парсираат и се чуваат во recipes.
6. **Ажурирање на UI**:
   - Откако се добиени рецептите, UI-от се ажурира:
   - Ако searchInitiated е true, компонентата ги мапира рецептите и ги прикажува насловите како линкови:

```
recipes.map((recipe) => (
  <div key={recipe.id}>
    <h3>
      <ul>
        <li><Link to={`/recipes/${recipe.id}/info`} className="recipe-links">{recipe.title}</Link></li>
      </ul>
    </h3>
  </div>
))
```

7. **Динамичко Патување за Детали за Рецептите**:
   - Кога корисникот кликне на насловот на рецепт, се навигира на детална верзија на тој рецепт преку динамичкиот пат (/recipes/info), што може да извлече повеќе детали врз основа на ID на рецептот.

Чекори на Поврзување

1. Корисник Внесува Состојки: Внесот се ажурира во состојбата.
2. Корисник Започнува Пребарување: Кликнувањето на копчето за пребарување активира fetch нарачка.
3. Испратена fetch Нарачка: Фронтендот прави GET нарачка до backend API со состојките.
4. Backend Обработува Нарачка: Spring Boot контролерот ја обработува нарачката, барајќи одговор - рецепти во базата на податоци.

5. Вратен Одговор: Backend-от испраќа JSON одговор назад до фронтендот.
6. Фронтендот Ажурира Состојка: Рецептите се чуваат во состојката и се прикажуваат на корисникот.
7. Взаемодејство на Корисниците со Рецептите: Корисниците можат да кликнат на рецепт за да видат повеќе детали.

Анотацијата @CrossOrigin му овозможува на Spring Boot backend-от да прифаќа нарачки од React фронтендот кој работи на http://localhost:3000. Ова е суштински важно за овозможување на крос-оригин нарачки, што е значајно во развојна средина каде фронтендот и backend-от можат да работат на различни порти.