

"Cooking with Cook Easy is TaStY, EaSY, IntEReStiNg"

ДОКУМЕНТАЦИЯ



Уеб приложение за споделяне, менажиране и коментиране на широк набор от рецепти

Разработчик: Киара Лазарова

Имейл: kival1020@yahoo.com

LinkedIn профил: <https://linkedin.com/in/kiara-lazarova-5b4147270>

GitHub хранилище: <https://github.com/M0r64na/CookEasy>

"Cooking with Cook Easy is TaStY, EaSY, InTEReStiNg"

1	Увод.....	5
1.1	Цели и задачи	5
1.2	Обхват	5
2	Използвани технологии	5
2.1	IntelliJ IDEA Ultimate.....	5
2.2	Spring Framework	7
2.3	Java	9
2.4	Spring MVC.....	10
2.5	Spring Data	11
2.6	MySQL.....	12
2.7	Hibernate ORM.....	12
2.8	ModelMapper.....	13
2.9	Spring Security.....	13
2.10	Pbkdf2PasswordEncoder.....	13
2.11	Bean Validation API 2.0	14
2.12	Hibernate Validator.....	15
2.13	Project Lombok.....	16
2.14	Spring Test.....	16
2.15	JUnit.....	17
2.16	JUnit Jupiter.....	17
2.17	Mockito.....	18
2.18	HTML	18
2.19	CSS	19
2.20	JavaScript.....	19
2.21	MDB Bootstrap.....	19
2.22	Thymeleaf.....	20
2.23	Thymeleaf Security.....	20
3	Архитектура.....	21
3.1	Архитектура на базата данни.....	22
3.1.1	Таблица <i>users</i>	23
3.1.2	Таблица genders.....	23
3.1.3	Таблица levels.....	24
3.1.4	Таблица roles.....	24
3.1.5	Таблица users_roles	25
3.1.6	Таблица recipes	25
3.1.7	Таблица categories	26

"Cooking with Cook Easy is TaStY, EaSY, IntEReStiNg"

3.1.8	Таблица comments.....	27
3.1.9	Таблица users_liked_recipes	28
3.1.10	Таблица users_saved_recipes	28
3.1.11	Таблица users_cooked_recipes	29
3.2	Архитектура на приложението	29
3.2.1	Пакет config	30
3.2.2	Пакет model	30
3.2.3	Пакет repository.....	31
3.2.4	Пакет service	31
3.2.5	Пакет util	32
3.2.6	Пакет web	32
4	Разработка на приложението.....	33
4.1	ApplicationSecurityConfiguration.java.....	33
4.2	Model	34
4.2.1	Binding.....	34
4.2.2	Service	36
4.2.3	Entity.....	36
4.2.4	View	37
4.3	Repository	38
4.4	Service	39
4.5	Util	40
4.5.1	Валидатори.....	40
4.5.2	Анотации	41
4.6	Web	42
4.6.1	Интерсептори.....	42
4.6.2	Контролери.....	43
4.7	Test	44
4.7.1	Компонентни тестове	44
4.7.2	Интеграционни тестове	45
5	Ръководство на потребителя	46
5.1	Стартиране на приложението	46
5.2	Изгледи.....	48
5.2.1	Начална страница за guest потребители	48
5.2.2	Форма за регистрация на потребител	49
5.2.3	Форма за влизане в потребителски акаунт	50

"Cooking with Cook Easy is TaStY, EaSY, IntEReStiNg"

5.2.4 Администраторски изглед за одобряване и архивиране на създадени коментари
към дадена рецепта 51

"Cooking with Cook Easy is TaStY, EaSY, IntEReStiNg"

1 Увод

Приготвянето на ястия е част от ежедневието на всяка домакиня, но съставянето на разнообразно и качествено меню е трудно. За подпомагането на тази сложна задача е необходима програма, чрез която да се запазят както традиционни, така и не толкова традиционни рецепти. „Онлайн система за споделяне, менажиране и коментиране на рецепти“ е подобна програма, чиято идея е да стимулира желанието на хората да консумират повече домашно пригответа храна.

1.1 Цели и задачи

С помощта на „Онлайн система за споделяне, менажиране и коментиране на рецепти“ потребителите имат възможност да създават, преглеждат, коментират, запазват и харесват готварски рецепти, след като са се регистрирали. Интернет приложението може да се администрира с помощта на администраторско меню, което позволява менажиране на потребители (изтриване на даден потребител или обновяване на неговата роля) и коментари (одобряване на даден коментар или архивиране на такъв). Архивираните от администраторите коментари се изтриват автоматично от системата на всеки 24 часа.

1.2 Обхват

Софтуерът се състои от вход за потребители, като всеки потребител има определена роля – потребител или администратор. Всички потребители в приложението имат възможност да редактират основната информация в своя профил, както и създадените от тях рецепти в уеб системата. Администраторите в приложението могат да управляват информация за регистрираните потребители и създадените от тях коментари.

2 Използвани технологии

2.1 IntelliJ IDEA Ultimate

IntelliJ IDEA е популярна интегрирана среда за разработка (IDE) за JVM-базирани програмни езици като Java, Kotlin, Groovy и други. Разработена е от JetBrains и се отличава с ергономичен дизайн. Предоставя широк набор от функции за подобряване на производителността на разработчиците, като това включва: анализ и довършване на код, интелигентно рефакториране, вграден дебъгер и инструменти за отстраняване на грешки и т.н. IntelliJ IDEA предлага интеграция с множество популярни и необходими в

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

практиката на един разработчик инструменти, а именно – системи за контрол на версии (например: GitHub, GitLab SVN), инструменти за съвместна разработка (Collaborative Development), инструменти за управление на бази данни, инструменти за профилиране (IntelliJ Profiler) и други. В допълнение IntelliJ IDEA Ultimate разполага с допълнителни възможности като поддръжка на Spring Framework, Selenium, Tomcat, Docker, Kubernetes и подобни. Предлага се като търговски продукт с безплатен 30-дневен пробен период и се използва широко от разработчици и организации по целия свят за създаване на софтуер.

IntelliJ IDEA Ultimate има конкурент на пазара, наречен Eclipse. Приложената долу таблица сравнява двете горепосочени среди за разработка, като изтъква предимствата на IntelliJ IDEA Ultimate.

Характеристика	IntelliJ IDEA Ultimate	Eclipse
Подпомагане на кода	Разширени функции за допълване и анализ на код. Интелигентни проверки на кода и откриване на грешки. Използва своя собствена вътрешна машина за анализ, наречена IntelliJ IDEA Analysis.	Базово допълване на код.
Рефакториране	Голям набор от разширени опции за рефакториране.	Представя ограничени опции за рефакториране.
Интеграция	Предлага отлична интеграция с множество инструменти и технологии (системи за контрол на версии, системи за профилиране и мониторинг, инструменти за виртуализация и др.).	Не предлага вградена интеграция със системи за контрол на версии.

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

Производителност	Отлична производителност за големи кодови бази.	Добра производителност за малки до средно големи кодови бази.
Персонализация	Предоставя разнообразни плъгини за персонализация, които допълнително спомагат работата на разработчика.	Въпреки наличните плъгини, разработката на софтуер не се улеснява особено поради многото инструменти, които не се поддържат по подразбиране.
Потребителски интерфейс	Ергономичен интерфейс, проектиран за висока производителност.	Неинтуитивен за ориентиране интерфейс.

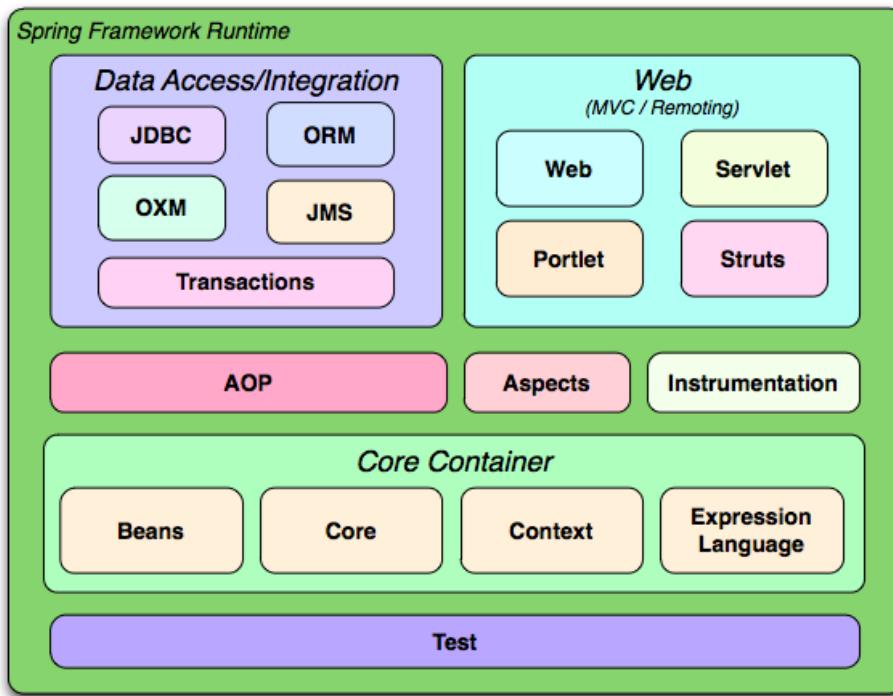
Фигура 1. Таблица, сравняваща IntelliJ IDEA Ultimate и Eclipse

2.2 Spring Framework

Spring Framework е популярен JVM фреймуърк с отворен код, предназначен да улесни разработването на приложения на корпоративно ниво. Той предоставя цялостен модел за програмиране и конфигуриране, позволяващ на разработчиците да изграждат стабилни, мащабируеми и лесни за поддръжка приложения. Предлага широк набор от функции и възможности, включително инверсия на контрола (IoC), аспектно-ориентирано програмиране (AOP), управление на транзакции и др.

Едно от основните предимства на Spring е неговата модулна архитектура, която позволява на разработчиците да използват само необходимите за разработвания софтуер компоненти, без да се налага да включват ненужни библиотеки или функционалности. Фреймуъркът се състои от приблизително 20 отделни модула, групирани в няколко категории, както е показано на следната схема:

"Cooking with Cook Easy is TaSTY, EaSY, InTEReStiNg"



Фигура 2. Схема на модулната архитектура на Spring Framework

Всяка група от модули има конкретно предназначение в цялостната архитектура на Spring, както следва:

- Core Container – осигурява основната функционалност на Spring Framework, включително инверсия на контрола (IoC) и инжектиране на зависимости (Dependency Injection);
- Data Access / Integration – осигурява поддръжка за работа с бази данни и други източници на данни, включително ORM, JDBC (Java Database Connectivity) и JPA (Java Persistence API);
- Web – осигурява поддръжка за изграждане на уеб приложения, включително уеб-базирани потребителски интерфейси, RESTful API и WebSocket комуникация;
- AOP (Аспектно-ориентирано програмиране) – осигурява поддръжка за AOP, позволявайки на разработчиците да повишат модулността на разработвания софтуер чрез изолиране на пресичащи се функционални дялове;
- Instrumentation – осигурява поддръжка за наблюдение и управление на производителността на Spring приложенията
- Test – осигурява поддръжка за тестване на Spring приложения, включително компонентно и интеграционно тестване.

"Cooking with Cook Easy is TaSTY, EaSY, InTEResting"

В заключение Spring Framework е мощен и гъвкав фреймуър, подходящ за разработване на сложни и мащабируеми JVM приложения. Неговата модулна архитектура, изчерпателен набор от функции и лекота на използване го правят ценен инструмент за разработчици, които се стремят да създават висококачествен софтуер.

2.3 Java

Java е популярен обектно-ориентиран език за програмиране на високо ниво, разработен от Sun Microsystems през 1995 година. Една от ключовите характеристики на Java е философията при нейното проектиране, гласяща следното: „пиши веднъж, изпълнявай навсякъде“. Това означава, че Java програмите, компилирани в байт код, могат да работят на всяка машина, която има инсталрирана Java Virtual Machine (JVM), независимо от основния хардуер и операционна система.

Синтаксисът на Java се базира на програмните езици C и C++, но за разлика от тях не може да функционира на ниско ниво. Java също така предлага широк набор от библиотеки и инструменти, които улесняват изграждането на сложни приложения.

Намира приложение в разработването на настолни приложения, уеб-базирани приложения, мобилни приложения и други. Езикът е известен със своята надеждност и функции за сигурност, поради което се използва за изграждане на критични системи като банков и финансов софтуер, здравни приложения и отбранителни системи.

Към 2022 г. според статистиката на GitHub Java е третият най-използван програмен език след JavaScript и Python.

Както е добре познато, Java има известни сходства със C++. Приложената долу таблица има за цел да сравни тези два програмни езика и да посочи силните страни на Java.

Характеристика	Java	C++
Платформена независимост	Платформено независим програмен език. Компилира се веднъж в байт код и може да се изпълни на всяка машина (<i>Write once, run anywhere.</i>)	Платформено зависим програмен език. За да се изпълни на отделна машина, трябва да се компилира отново (<i>Write once, compile anywhere.</i>)

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

Компилиране и интерпретиране	Може както да се компилира, така и да се интерпретира.	Може само да се компилира.
Управление на паметта	Контролира се от <i>Java Virtual Machine (JVM)</i> . Включва <i>garbage collector</i> , който автоматично управлява разпределението и освобождаването на паметта.	Контролира се ръчно и е отговорност на разработчика. Това осигурява възможност за по-добър контрол на употребата на памет, но също така създава рискове за възникване на грешки, свързани с паметта.
Работа с нишки (<i>threads</i>)	Има вградена поддръжка на работа с нишки, което позволява на разработчиците да създават многонишкови приложения.	Поддържа се едва от C++11, пуснат в употреба през 2011 година.
Откриване на грешки по време на изпълнение	Контролира се от вграден механизъм за откриване на грешки по време на изпълнение в <i>Java Virtual Machine (JVM)</i> .	Отговорност е на разработчика.

Фигура 3. Таблица, сравняваща Java и C++

2.4 Spring MVC

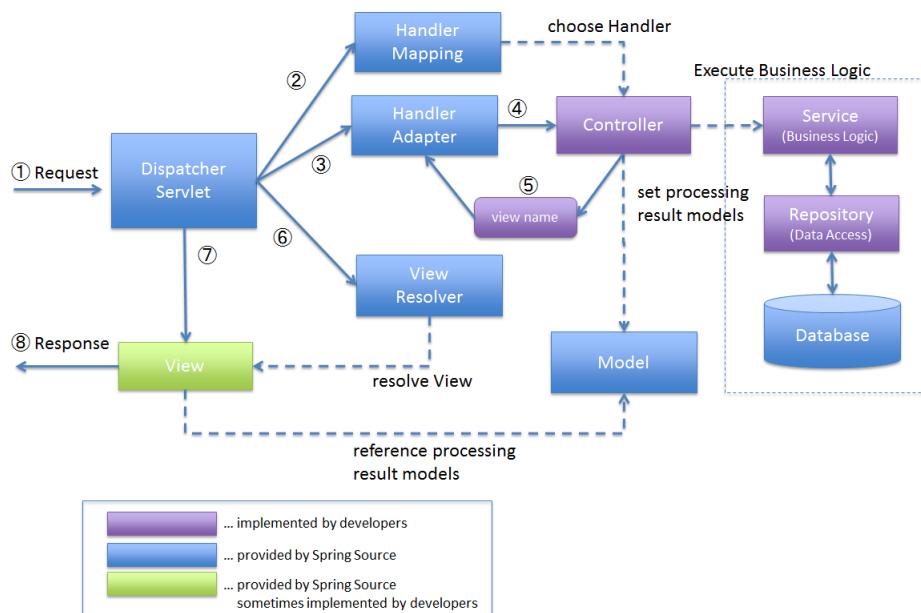
Spring MVC е модул в Spring Framework, който предоставя възможност за имплементиране на *Model–View–Controller (MVC)* архитектурния модел за изграждане на уеб приложения. Моделите на данните представляват основната информация, необходима за работата на дадения софтуер. Потребителите на съответното приложение взаимодействат с наличната информация посредством отделни изгледи. Изгледите, от своя страна, комуникират с контролерите, действащи като интерфейс между моделите на данни и съответстващите изгледи. Основната задача на контролерите е да обработят

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

входящите заявки чрез подходящи методи и да върнат изгледи спрямо извършените от потребителя на системата действия.

Този механизъм на действие на MVC архитектурния модел в контекста на Spring MVC модула се осъществява с помощта на *DispatcherServlet*. *DispatcherServlet* е сървлет клас, чиято основна цел е да обработва входящи уеб заявки, които съответстват на конфигурирания URL модел. Той взема входящия Uniform Resource Identifier (URI) и намира съответният контролер, който да обработи по подходящ начин постъпилата заявка. В този смисъл *DispatcherServlet* действа като основен контролер, или наречен още *front controller*, насочващ отделните заявки към техния пред назначен контролер.

Приложената схема показва как се осъществява архитектурния модел на Spring MVC:



Фигура 4. Схема, показваща архитектурата на Spring MVC

Spring MVC е силно конфигурируем и разширяем, което позволява на разработчиците да персонализират поведението на модула в зависимост от целите на разработвания софтуер. Използва се широко в разработката на уеб приложения и е известен със своята лекота и гъвкавост.

2.5 Spring Data

Spring Data е модул в Spring Framework, който предоставя последователен и опростен програмен модел за достъп до съхраняваните данни, като същевременно

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

запазва отличителните характеристики на използваното хранилище на данни. Хранилищата, които могат да бъдат използвани съвместно със Spring Data, са разнообразни – от релационни бази данни до NoSQL такива. Също така Spring Data осигурява на разработчиците набор от услуги и абстракции за работа с данни, включително хранилища (repositories), методи за заявки и управление на транзакции.

2.6 MySQL

MySQL е система за управление на релационни бази данни (RDBMS) с отворен код, която използва *Structured Query Language* (SQL) за управление на големи количества данни. Намира широко приложение в уеб приложения и други софтуерни системи като бек-енд база данни за съхранение, извлечане и обработване на данни. Поддържа набор от функции, включително репликация, управление на процедури и транзакции и други. MySQL е бърз, надежден и машабируем поради уникалната си архитектура на системата за съхранение. Предоставя резултати с много висока производителност в сравнение с други бази данни, без да губи съществена функционалност на софтуера. MySQL е написан на C и C++ и е съвместим с множество операционни системи (например: ArcaOS, Oracle Solaris, SunOS, Linux, Windows, macOS и т.н.).

2.7 Hibernate ORM

Hibernate ORM, или накратко Hibernate, е фреймуърк за *Object-Relational Mapping* (ORM), който опростява разработването на приложения, управявани от релационни бази данни. Той автоматизира мапирането на обектно-ориентирани модели към релационни бази данни, като същевременно намалява количеството шаблонен код, който разработчиците иначе трябва да напишат. Hibernate предоставя редица разширени функции като кеширане, отложено зареждане и оптимизиране на заявки, които помагат за подобряване на производителността и машабируемостта на приложението. Например механизъмът за кеширане на Hibernate действа като отделен слой между приложението и използваната база данни и намалява необходимото време за извлечане на желаните данни. От друга страна, механизъмът за отложено зареждане зарежда само необходимите данни, подобрявайки използването на паметта на приложението. Hibernate поддържа различни стратегии за мапиране, което включва Java анотации и XML конфигурации, което дава на разработчиците гъвкавост в това как те мапират своите обекти към таблици. С Hibernate разработчиците могат да се съсредоточат върху бизнес логиката на

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

своите приложения и да оставят фреймуърка да управлява слоя за достъп до данни, правейки процеса на разработка по-бърз и по-ефективен.

2.8 ModelMapper

ModelMapper е Java библиотека, която помага за опростяване на преобразуването на един тип обект към друг тип. Тя позволява на разработчиците да дефинират сложни правила за мапиране между различни обектни модели, като намалява количеството шаблонен код, необходим иначе за мапиране на обекти. ModelMapper също така предоставя разширени функции като автоматично преобразуване на типове, адаптивни стратегии за мапиране и други. С ModelMapper разработчиците могат лесно да мапират обекти от един домейн в друг, без да се притесняват за детайлите на ниско ниво, свързани с процеса на мапиране.

2.9 Spring Security

Spring Security е модул в Spring Framework, който осигурява автентикация, авторизация и защита срещу често срещани заплахи за сигурността като *Cross-Site Scripting* (XSS) атаки и *Cross-Site Request Forgery* (CSRF) атаки. С помощта на Spring Security разработчиците имат възможността лесно да конфигурират политиките за сигурност в разработваното приложение въз основа на техните специфични изисквания. Също така модулът може да бъде интегриран с други модули на Spring като Spring Data и Spring MVC, за да се изградят сигурни и надеждни приложения на корпоративно ниво. Spring Security предлага широк набор от функции за сигурност, включително поддръжка на множество методи за автентикация и авторизация, криптиране на парола, управление на сесии, защитена комуникация през HTTPS и т.н. В заключение Spring Security е основен инструмент за изграждане на сигурни и надеждни JVM-базирани приложения.

2.10 Pbkdf2PasswordEncoder

Pbkdf2PasswordEncoder е енкодер на пароли, предоставен от Spring Security. Той имплементира алгоритъма *Password-Based Key Derivation Function 2* (PBKDF2). PBKDF2 е широко прилаган криптографски алгоритъм, който се използва за генериране на защитени криптографски ключове от паролата на кой да е потребител. Pbkdf2PasswordEncoder приема паролата на потребителя като вход заедно с произволно генериран *salt*, след което итеративно прилага PBKDF2 алгоритъма, за да генерира защитен хеш на съответната парола. Полученият хеш се съхранява сигурно в базата

"Cooking with Cook Easy is TaSTY, EaSY, InTEReStiNg"

данни и може да се използва за последваща верификация на потребителската парола по време на автентикация. Използването на PBKDF2 и salt-базирано хеширане е една от най-добрите практики за съхранение на пароли, тъй като осигурява висока степен на защита срещу често срещани атаки, свързани с потребителската парола, като *Dictionary* атаки и *Rainbow Table* атаки.

Salt е произволна стойност, която се добавя към паролата на определен потребител, преди тя да бъде хеширана. Целта на salt-а е да добави допълнителна произволност към хеша на паролата, което прави по-трудно за злонамерени лица да използват предварително изчислени таблици за откриване на оригиналната парола. Чрез използването на уникален salt за всяка парола, дори потребители с една и съща парола ще имат различни хешове на техните пароли, съхранени в базата данни. Това добавя допълнителен слой сигурност. Стойността на salt-а обикновено се съхранява заедно с хеша на паролата в базата данни с цел да може да се използва за проверка на паролата на потребителя по време на автентикация.

2.11 Bean Validation API 2.0

Bean Validation API 2.0 е Java фреймуърк (познат още като JSR 380), който предоставя набор от анотации и интерфейси за валидиране на Java обекти, наречени Java Beans. Той позволява на разработчиците да дефинират ограничения върху свойствата и полетата на обекта и предоставя механизъм за валидиране на тези ограничения по време на изпълнение.

Bean Validation API се основава на спецификацията за проверка на Java API и се изпълнява от различни фреймуърци и библиотеки на Java като Hibernate Validator, Apache BVal и Spring Validation.

Използвайки Bean Validation API, разработчиците могат лесно да дефинират ограничения като минимални и максимални стойности, регулярни изрази и персонализирана логика за валидиране. Тези ограничения могат да се добавят към свойствата на обекта и полетата с помощта на прости Java анотации като `@NotNull`, `@Size` и `@Min`.

По време на изпълнение Bean Validation API предоставя прост и последователен начин за валидиране на обекти и докладване на всякакви нарушения на ограниченията. Разработчиците могат също така да персонализират процеса на валидиране, като

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

дефинират свои собствени групи за валидиране, съобщения за грешки и последователности за валидиране.

Bean Validation API се използва широко в уеб приложения, където валидирането на входа и целостта на данните са критични. Използва се и в други контексти, като настолни приложения, пакетни задания и микроуслуги.

Като цяло Bean Validation API опростява задачата за валидиране на Java обекти, насърчава повторната употреба и поддръжка на кода и помага на разработчиците да пишат по-стабилни и надеждни приложения.

2.12 Hibernate Validator

Hibernate Validator е мощна и гъвкава референтна реализация на Bean Validation API 2.0. Hibernate Validator дефинира набор от анотации, ограничения и API за валидиране на данни. Предоставя широк набор от вградени ограничения, включително общи правила за валидиране като формат на имейл, дължина на низ и числови диапазони. Той също така позволява на разработчиците да дефинират персонализирани ограничения, интерполатори на съобщения (процесът на създаване на съобщения за грешка при нарушаване на ограниченията за валидация на Java Bean обекти) и групи за валидиране.

Една от ключовите характеристики на Hibernate Validator е способността да се извърши проверка на различни етапи от жизнения цикъл на обекта, като например преди запазване, преди актуализиране или преди връщане на отговор на клиент. Това гарантира, че данните винаги са последователни и отговарят на изискваните критерии. Hibernate Validator може също да се интегрира с други рамки и технологии, като Spring, CDI и RESTful уеб услуги.

Hibernate Validator също поддържа интернационализация и локализация, позволяйки на разработчиците да предоставят съобщения на множество езици и да персонализират форматите на съобщенията. Той също така поддържа програмно валидиране, където ограниченията се дефинират с помощта на код вместо анотации, и позволява създаването на персонализирани валидатори на ограничения.

В обобщение, Hibernate Validator предоставя цялостно и гъвкаво решение за внедряване на валидиране в Java приложения, като помага да се гарантира, че данните са последователни, валидни и отговарят на изискваните критерии.

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

2.13 Project Lombok

Project Lombok е популярна библиотека с отворен код, която има за цел да намали шаблонния код в Java класовете. Тя предоставя набор от Java анотации, които могат да се добавят към Java класовете, за да се генерира автоматично код като методи за четене и промяна на стойността на полетата в даден клас, конструктори и други. Това може значително да намали количеството на кода, който иначе трябва да се напише, като направи кодовата база по-сбита и по-лесна за четене.

Някои от основните характеристики на Project Lombok включват:

- Автоматично генериране на методи за четене и промяна на стойността на полетата в даден клас, конструктори и други често използвани методи с помощта на Java анотации;
- Опростяване на създаването на шаблони за конструиране на обекти;
- Намаляване на многословието при логинг (logging) чрез предоставяне на пристапа анотация `@Log` за автоматично генериране на поле за логинг;
- Предоставяне на редица други анотации като `@NonNull` за генериране на проверки за `null` стойности и `@Synchronized` за генериране на синхронизирани блокове.

2.14 Spring Test

Spring Test е фреймуърк за тестване, представляващ цялостен модул в Spring Framework, който се използва за тестване на Spring-базирани приложения. Той предоставя набор от функции и помощни програми за тестване на компоненти на Spring в изолация (mocking) или по интегриран начин.

Spring Test предоставя различни анотации за тестване, например `@RunWith`, `@ContextConfiguration`, `@Transactional`, които могат да се използват за конфигуриране и персонализиране на средата за тестване. Рамката също така предоставя различни тестови помощни програми като `MockMvc`, `TestRestTemplate`, `TestEntityManager` и други, които могат да се използват за тестване на различни части на даденото Spring приложение.

С помощта на Spring Test разработчиците могат да пишат както интеграционни, така и end-to-end тестове за своите Spring-базирани приложения. Рамката осигурява

"Cooking with Cook Easy is TaSTY, EaSY, InTEReStiNg"

последователна и надеждна среда за тестване, която помага да се гарантира качеството на кода на приложението.

2.15 JUnit

JUnit е популярен фреймуърк за компонентно тестване с отворен код за програмния език Java. Той предоставя пристапа и лесна за използване платформа за писане и изпълнение на повтарящи се автоматизирани тестове. JUnit е проектиран така, че да поддържа методологиите за разработване, основано на тестове (TDD), и за разработване, основано на поведение (BDD), като помага на разработчиците да подобрят качеството на кода, да откриват дефекти на ранен етап и да намалят разходите за разработване на софтуер. Фреймуъркът включва анатации за определяне на тестови методи, методи, които се изпълняват преди и след тестовете, и методи, които се изпълняват еднократно преди и след набор от тестове.

JUnit позволява на разработчиците да пишат тестови случаи, които потвърждават очакваното поведение на тествания код. Той също така предоставя богат набор от *assertions*, които могат да се използват за проверка на коректността на получения изход на тествания код. Тестовете на JUnit могат да бъдат изпълнявани автоматично като част от *build* процеса, което осигурява незабавна обратна връзка за качеството на кода. Това го прави основен инструмент за разработчиците, които искат да гарантират, че кодът им работи по предназначение.

В заключение JUnit е мощен инструмент за разработчиците на софтуер, който им позволява да пишат компонентни тестове, гарантиращи коректността на техния код. Той помага за подобряване на качеството на кода, увеличаване на производителността и намаляване на разходите за разработване на софтуер, което го прави незаменим инструмент за съвременната разработка на софтуер.

2.16 JUnit Jupiter

JUnit Jupiter е следващото поколение на фреймуърка за компонентно тестване JUnit и е част от JUnit 5. Предоставя няколко нови функции като например поддръжка на параметризиранi тестове, динамични тестове и вложени тестове. Програмният модел на Jupiter е проектиран така, че да бъде по-modерен и гъвкав, което улеснява писането и поддръжането на тестове. В допълнение поддръжката на функциите на Java 8 като ламбда и потоци е подобрена. Jupiter предоставя нов модел на разширение, който

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

позволява на разработчиците да включват допълнителна функционалност, свързана с тестовете, включително потребителски анотации, lifecycle callbacks и инжектиране на зависимости.

2.17 Mockito

Mockito е популярен фреймуърк с отворен код за изолиране на зависимости (mocking) в Java, който се използва за компонентно тестване. Той позволява на разработчиците да създават и използват имитирани (mock) обекти, за да симулират поведението на реални обекти по контролиран начин, което може да бъде полезно за тестване на конкретни части от приложението.

Mockito предоставя прост и лесен за използване API за създаване на имитирани обекти и дефиниране на тяхното поведение, което улеснява писането на кратки и разбираеми тестови случаи. Той поддържа широк набор от функции, например проверка на извикването на методи, съвпадение на аргументи и т.н., което позволява на разработчиците да създават силно персонализирани моделиращи обекти.

Едно от основните предимства на Mockito е неговата гъвкавост и възможност за интегриране с други рамки за тестване като JUnit и TestNG. Той може да се използва и с популярни Java IDE, включително Eclipse и IntelliJ IDEA. Използва се за компонентно тестване в различни Java приложения като уеб приложения, микросървиси и корпоративни приложения.

2.18 HTML

HTML, съкратено от HyperText Markup Language, е език за маркиране, който се използва за създаване на структурата и съдържанието на уеб страници. Той предоставя стандартен начин за описание на елементите и структурата на дадена уеб страница, включително текст, изображения, връзки и други. HTML се използва за създаване и показване на съдържание в уеб браузърите.

HTML използва тагове за дефиниране на елементи като заглавия, параграфи, списъци и връзки. Тези тагове са затворени в ъглови скоби (< >) и могат да включват атрибути, които предоставят допълнителна информация за елемента, например размера или цвета на шрифта. Уеб разработчиците използват HTML, за да създават уеб страници, а полученият HTML код се интерпретира от уеб браузърите с цел показване на съдържанието на потребителите.

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

2.19 CSS

CSS е съкращение от Cascading Style Sheets. Това е език за стилизиране, който се използва за определяне на оформлението на уеб страниците в дадено уеб-базирано приложение. CSS се използва за контрол на външния вид на HTML елементите в уеб страницата. Това включва цветове, шрифтове, оформление, разстояния и т.н. Чрез разделяне на стилизацията на определена уеб страница от нейното съдържание, CSS позволява на разработчиците да създават последователни и атрактивни уеб страници, които са по-лесни за поддържане и актуализиране. CSS може да се прилага към HTML документи по три начина: външно (external), вътрешно (internal) или вградено (inline). CSS е ключов компонент на съвременната уеб разработка и често се използва в комбинация с HTML и JavaScript за създаване на богати, интерактивни уеб приложения.

2.20 JavaScript

JavaScript (често съкращаван като JS) е динамичен интерпретиран език за програмиране от високо ниво, който се използва за изграждане на адаптивни и визуално привлекателни уеб страници и онлайн програми като например видеоигри. Създаден е през 1995 г. от Брэндън Айх по време на работата му в Netscape Communications Corporation. JavaScript често се използва за добавяне на поведение към HTML и CSS уеб страници, като например създаване на динамични ефекти, валидиране на потребителския вход, манипулиране на DOM (Document Object Model) дървото и извършване на асинхронни заявки към сървъри. Може да се изпълнява както от страна на клиента (в даден уеб браузър), така и от страна на сървъра (с помощта на Node.js).

2.21 MDB Bootstrap

MDB Bootstrap е популярна библиотека за потребителски интерфейс за разработване на уеб сайтове, базирана на Bootstrap и Google Material Design. Тя включва допълнителни компоненти и функции, които подобряват дизайна и функционалността на уеб страниците. Проектирана е така, че да бъде лесна за използване и персонализиране, като има на разположение голям набор от предварително създадени шаблони и теми. Библиотеката е изградена с помощта на HTML, CSS и JavaScript и съдържа много елементи на потребителския интерфейс за многократна употреба като бутони, форми, модални елементи, карти, навигационни панели и много други. MDB Bootstrap включва и допълнителни плъгини на JavaScript, например библиотеки за

"Cooking with Cook Easy is TaSTY, EaSY, InTEReStiNg"

графики и диаграми и валидиране на формуляри, които могат лесно да бъдат интегрирани в уеб приложения.

2.22 Thymeleaf

Thymeleaf е сървърен шаблонен engine на Java, който позволява на разработчиците да изграждат уеб приложения с възможност за създаване на динамично съдържание. Това е библиотека с отворен код, която се интегрира със Spring Framework, позволявайки на разработчиците да разработват уеб приложения, използващи възможностите на Java и HTML.

Thymeleaf използва синтаксис, подобен на XML/HTML, и улеснява създаването на шаблони, които могат да бъдат визуализирани от страна на сървъра. Той поддържа използването на условни изрази, цикли и динамично обвързване на съдържанието.

Една от ключовите характеристики на Thymeleaf е способността му да се справя с интернационализацията и локализацията. Thymeleaf позволява на разработчиците да създават шаблони, които могат да се визуализират на няколко езика, без да е необходимо да се използват отделни шаблони.

Thymeleaf също така предоставя възможност за създаване на персонализирани тагове и изрази, което, от своя страна, дава на разработчиците гъвкавост за създаване на собствена функционалност и разширяване на възможностите на шаблонния engine.

Thymeleaf е лек, бърз и лесен за използване, което го прави популярен избор за разработчиците, изграждащи MVC уеб приложения със Spring Framework.

2.23 Thymeleaf Security

Thymeleaf Security е интеграционна библиотека, която съчетава функциите на шаблонния engine Thymeleaf с модула Spring Security. Тя предоставя набор от помощни класове и потребителски диалекти, които позволяват на разработчиците да създават сигурни уеб приложения с шаблони Thymeleaf. Thymeleaf Security опростява интеграцията на Spring Security с Thymeleaf, като предоставя редица полезни функции.

Пример за подобни функции са следните:

- Контрол на достъпа: възможност за лесно показване и скриване на части от даден шаблон въз основа на ролята или статуса на удостоверяване на потребителя;

"Cooking with Cook Easy is TaStY, EaSY, IntEReStiNg"

- CSRF защита: Thymeleaf Security осигурява вграден механизъм за CSRF защита, който работи безпроблемно със Spring Security;
- Контекст на сигурността: възможност за получаване на достъп и използване на контекста на Spring Security в създадените шаблони с цел извлечане на информация за текущия потребител или за проверка на състоянието на автентификацията;
- Връзки за влизане/излизане: възможност за използване на вградените атрибути за влизане и излизане на Thymeleaf Security с цел създаване на връзки за влизане и излизане в създадените шаблони;
- Генериране на URL адреси: Thymeleaf Security предоставя помощна програма за генериране на URL адреси, която взема предвид статуса на автентификацията и генерира подходящ URL адрес;
- Превод на съобщения: Thymeleaf Security предоставя механизъм за превод на съобщения, свързани със сигурността, като например съобщения за грешка, които се показват, когато потребителят въведе невалидни идентификационни данни.

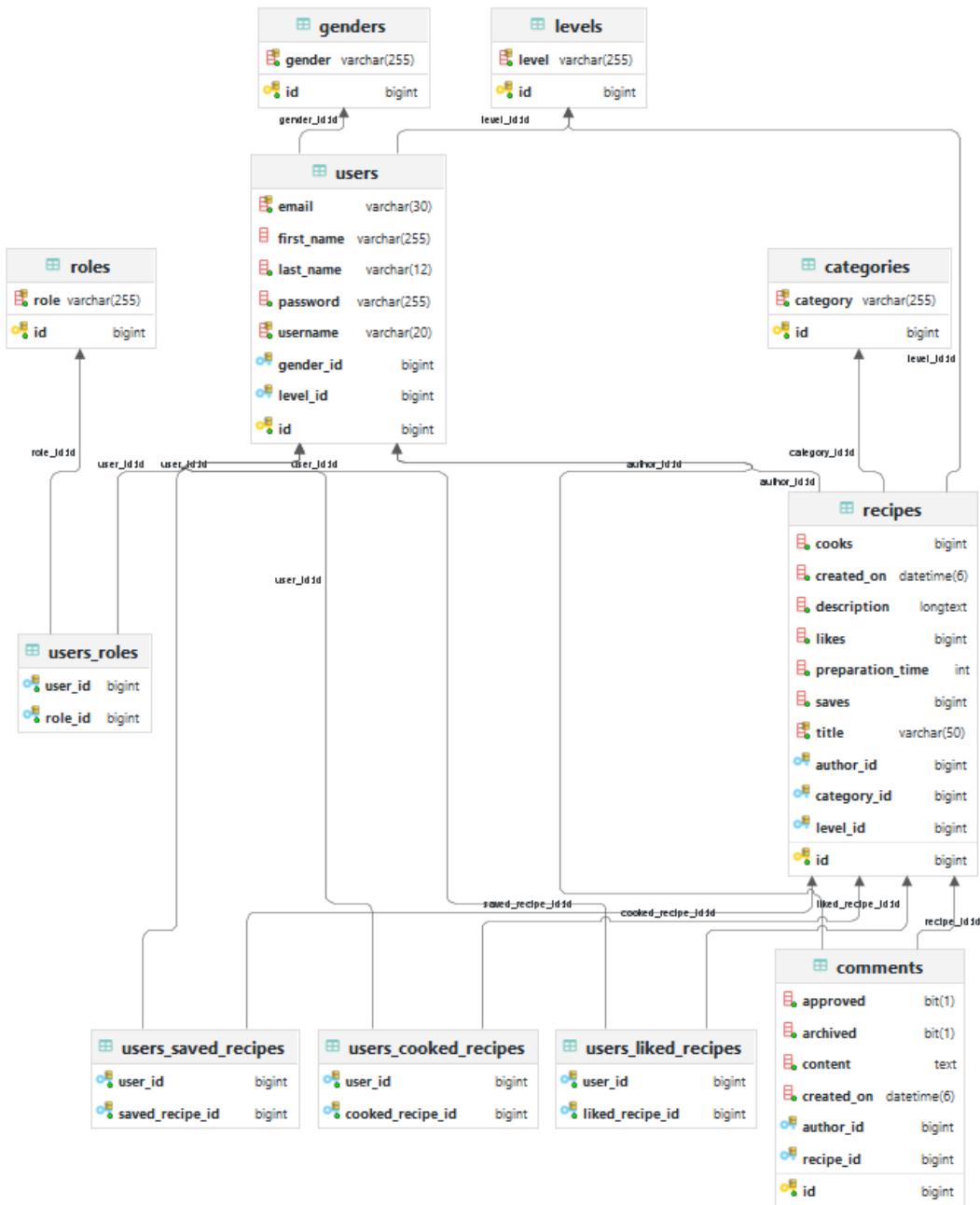
В заключение Thymeleaf Security спомага за опростяване на интеграцията между Thymeleaf и Spring Security, като улеснява добавянето на свързани със сигурността функции към разработваното уеб приложение.

3 Архитектура

В тази част на документацията върху дадената дипломна работа ще бъде разгледана основната концепция на работата на системата и технологиите, свързани с разработката ѝ.

"Cooking with Cook Easy is TaSTY, EaSY, InTEReStInG"

3.1 Архитектура на базата данни



Фигура 5. EER диаграми на таблициите в БД

"Cooking with Cook Easy is TaStY, EaSY, InTEReStiNg"

3.1.1 Таблица *users*

Име на колоната	Тип на колоната	Ограничения на колоната
id	BIGINT	Primary Key Not Null Auto Increment
username	VARCHAR(20)	Not Null Unique
first_name	VARCHAR(255)	-
last_name	VARCHAR(12)	Not Null
password	VARCHAR(255)	Not Null
email	VARCHAR(30)	Not Null Unique
gender_id	BIGINT	Foreign Key
level_id	BIGINT	Foreign Key

Фигура 6. Таблица *users*

Таблицата *users* съдържа основната информация за регистрираните потребители в системата. Полето *id* служи като първичен ключ на таблицата, представлява уникален идентификатор за всеки отделен запис в таблицата. Полето *username* съхранява потребителското име на потребителя. Полето *first_name* съхранява собственото име на потребителя. Полето *last_name* съхранява фамилията на потребителя. Полето *password* съхранява паролата на потребителя. Полето *email* съхранява електронната поща на потребителя. Полето *gender_id* е външен ключ към таблицата *genders* и съхранява пола на потребителя. Полето *level_id* е външен ключ към таблицата *levels* и съхранява категорията ниво на напредналост към която спада потребителят.

3.1.2 Таблица *genders*

Име на колоната	Тип на колоната	Ограничения на колоната
id	BIGINT	Primary Key Not Null

"Cooking with Cook Easy is TaSTY, EaSY, InTEReStiNg"

		Auto Increment
gender	VARCHAR(255)	Not Null Unique

Фигура 7. Таблица genders

Таблицата genders съдържа основната информация за наличните опции за пол на регистрираните потребители в системата. Полето id служи като първичен ключ на таблицата, представлява уникален идентификатор за всеки отделен запис в таблицата. Полето gender съдържа възможните опции за избор на пол на потребителите в уеб приложението. Полето gender съхранява точно 2 уникални стойности – MALE и FEMALE.

3.1.3 Таблица levels

Име на колоната	Тип на Колоната	Ограничения на колоната
id	BIGINT	Primary Key Not Null Auto Increment
level	VARCHAR(255)	Not Null Unique

Фигура 8. Таблица levels

Таблицата levels съдържа основната информация за наличните опции за категория ниво на напредналост към която спадат както регистрираните потребители в системата, така и създадените рецепти в уеб приложението. Полето id служи като първичен ключ на таблицата, представлява уникален идентификатор за всеки отделен запис в таблицата. Полето level съдържа възможните опции за избор на категория ниво на напредналост на потребителите и рецептите в уеб приложението. Полето level съхранява точно 3 уникални стойности – BEGINNER, INTERMEDIATE и ADVANCED.

3.1.4 Таблица roles

Име на колоната	Тип на колоната	Ограничения на колоната
id	BIGINT	Primary Key

"Cooking with Cook Easy is TaStY, EaSY, InTEReStInG"

		Not Null Auto Increment
level	VARCHAR(255)	Not Null Unique

Фигура 9. Таблица roles

Таблицата levels съдържа основната информация за наличните опции за роли на регистрираните потребители в уеб приложението. Полето id служи като първичен ключ на таблицата, представлява уникален идентификатор за всеки отделен запис в таблицата. Полето role съдържа възможните опции за избор на роля на отделните потребители в уеб приложението. Полето role съхранява точно 2 уникални стойности – USER и ADMIN.

3.1.5 Таблица users_roles

Име на колоната	Тип на колоната	Ограничения на колоната
user_id	BIGINT	Foreign Key
role_id	BIGINT	Foreign Key

Фигура 10. Таблица users_roles

Таблицата users_roles съдържа основната информация за ролите на регистрираните потребители в уеб приложението. Полето user_id е външен ключ към таблицата users и съхранява уникалния идентификатор на потребителя. Полето role_id е външен ключ към таблицата roles и съхранява уникалния идентификатор на присъщата роля на дадения потребител.

3.1.6 Таблица recipes

Име на колоната	Тип на колоната	Ограничения на колоната
id	BIGINT	Primary Key Not Null Auto Increment
created_on	DATETIME(6)	Not Null
title	VARCHAR(50)	Not Null Unique

"Cooking with Cook Easy is TaSTY, EaSY, InTEReStiNg"

preparation_time	INT	Not Null
description	LONGTEXT	Not Null
likes	BIGINT	Not Null
saves	BIGINT	Not Null
cooks	BIGINT	Not Null
category_id	BIGINT	Foreign Key
level_id	BIGINT	Foreign Key
author_id	BIGINT	Foreign Key

Фигура 11. Таблица recipes

Таблицата recipes съдържа основната информация за добавените рецепти в системата. Полето id служи като първичен ключ на таблицата, представлява уникален идентификатор за всеки отделен запис в таблицата. Полето username съхранява потребителското име на потребителя. Полето created_on съхранява датата и часа на създаване на рецептата. Полето title съхранява наименованието на рецептата. Полето preparation_time съхранява необходимото време за приготвяне на рецептата в минути. Полето description съхранява описание на рецептата. Полето likes съхранява броя на харесванията на рецептата, който при създаване на дадената рецепт винаги е 0. Полето saves съхранява броя на запазванията на рецептата, който при създаване на дадената рецепт винаги е 0. Полето cooks съхранява колко пъти е изпробвана рецептата, като този брой пъти при създаване на дадената рецепт винаги е 0. Полето category_id е външен ключ към таблицата categories и съхранява категорията, към която спада рецептата. Полето level_id е външен ключ към таблицата levels и съхранява категорията ниво на напредналост към която спада рецептата. Полето author_id е външен ключ към таблицата users и съхранява уникалния идентификатор на потребителя, който е автор на съответната рецепт.

3.1.7 Таблица categories

Име на колоната	Тип на колоната	Ограничения на колоната
id	BIGINT	Primary Key Not Null Auto Increment

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

category	VARCHAR(255)	Not Null Unique
----------	--------------	--------------------

Фигура 12. Таблица categories

Таблицата categories съдържа основната информация за наличните опции за категории на добавените рецепти в уеб приложението. Полето id служи като първичен ключ на таблицата, представлява уникален идентификатор за всеки отделен запис в таблицата. Полето category съдържа възможните опции за избор на категория на отделните рецепти в уеб приложението. Полето category съхранява точно 10 уникални стойности – SALAD, STARTER, SANDWICH, SOUP, MEAT, PASTA, BREAD, DRINK, DESSERT и OTHER.

3.1.8 Таблица comments

Име на колоната	Тип на колоната	Ограничения на колоната
id	BIGINT	Primary Key Not Null Auto Increment
approved	BIT(1)	Not Null
archived	BIT(1)	Not Null
created_on	DATETIME(6)	Not Null
content	TEXT	Not Null
author_id	BIGINT	Foreign Key
recipe_id	BIGINT	Foreign Key

Фигура 13. Таблица comments

Таблицата comments съдържа основната информация за създадените коментари към публикуваните рецепти в системата. Полето id служи като първичен ключ на таблицата, представлява уникален идентификатор за всеки отделен запис в таблицата. Полето approved указва дали дадения коментар е одобрен от администратор в уеб приложението, като само одобрените коментари се публикуват към съответната рецепта. Полето archived указва дали дадената рецепта е архивирана от администратор в уеб приложението, като всички архивирани коментари се изтриват автоматично от системата на всеки 24 часа. Полето created_on съхранява датата и часа на създаване на коментара.

"Cooking with Cook Easy is TaStY, EaSY, InTEReStiNg"

Полето content съхранява съдържанието на коментара. Полето author_id е външен ключ към таблицата users и съхранява уникалния идентификатор на потребителя, който е автор на съответния коментар. Полето recipe_id е външен ключ към таблицата recipes и съхранява уникалния идентификатор на рецептата, към която се отнася даденият коментар.

3.1.9 Таблица users_liked_recipes

Име на колоната	Тип на колоната	Ограничения на колоната
user_id	BIGINT	Foreign Key
liked_recipe_id	BIGINT	Foreign Key

Фигура 14. Таблица users_liked_recipes

Таблицата users_liked_recipes съдържа основната информация за харесаните рецепти от регистрираните потребители в уеб приложението. Полето user_id е външен ключ към таблицата users и съхранява уникалния идентификатор на потребителя. Полето liked_recipe_id е външен ключ към таблицата recipes и съхранява уникалния идентификатор на харесаната рецептa от дадения потребител.

3.1.10 Таблица users_saved_recipes

Име на колоната	Тип на колоната	Ограничения на колоната
user_id	BIGINT	Foreign Key
saved_recipe_id	BIGINT	Foreign Key

Фигура 15. Таблица users_saved_recipes

Таблицата users_saved_recipes съдържа основната информация за запазените рецепти от регистрираните потребители в уеб приложението. Полето user_id е външен ключ към таблицата users и съхранява уникалния идентификатор на потребителя. Полето saved_recipe_id е външен ключ към таблицата recipes и съхранява уникалния идентификатор на запазената рецептa от дадения потребител.

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

3.1.11 Таблица users_cooked_recipes

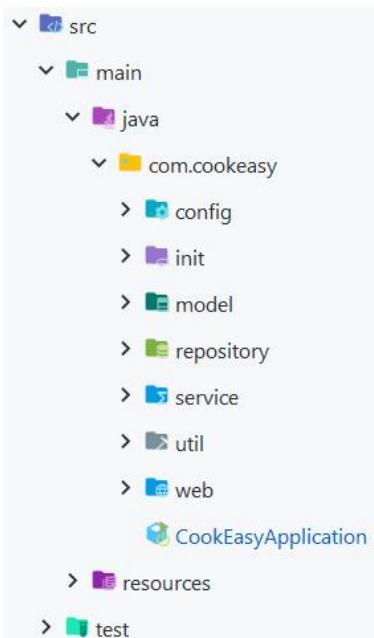
Име на колоната	Тип на колоната	Ограничения на колоната
user_id	BIGINT	Foreign Key
cooked_recipe_id	BIGINT	Foreign Key

Фигура 16. Таблица users_cooked_recipes

Таблицата users_cooked_recipes съдържа основната информация за отбелязаните като изпробвани рецепти от регистрираните потребители в уеб приложението. Полето user_id е външен ключ към таблицата users и съхранява уникалния идентификатор на потребителя. Полето cooked_recipe_id е външен ключ към таблицата recipes и съхранява уникалния идентификатор на отбелязаната като изпробвана рецепта от дадения потребител.

3.2 Архитектура на приложението

Логиката на приложението е изградено от 6 основни пакета, наречени още packages, разпределени в директорията src.main.java: config, model, repository, service, util и web. Изгледите, или още views, съответстващи на функционалността на приложението, се съдържат в директорията src.main.resources. Компонентните и интеграционните тестовете на кода на уеб приложението се намират съответно в директорията src.test.

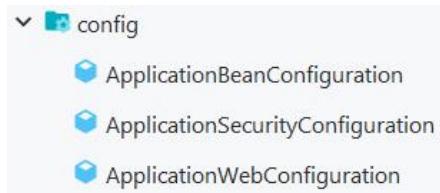


Фигура 17. Архитектура на приложението

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

3.2.1 Пакет config

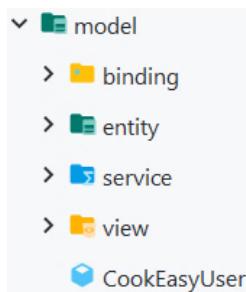
Пакетът config съдържа конфигурациите на уеб приложението, като това включва: конфигурацията на отделните bean обекти; конфигурацията на Spring Security, необходима за правилната работа на приложението; конфигурацията на интерсептора, който се използва за така наречената интернационализация, позната още като i18n.



Фигура 18. Пакет config

3.2.2 Пакет model

Пакетът model съдържа всички модели, необходими за предлаганите от уеб приложението услуги. Моделите биват няколко вида и съответно са разпределени в няколко подпакета. Binding моделите служат за извличане на въведената в дадена форма информация в браузъра, която информация се валидира по подходящ начин. След валидиране на тази информацията binding моделите се преобразуват към service модели, благодарение на които въведената в браузъра информация се предава към бизнес логиката на приложението. Entity моделите представляват класове, описващи таблиците в MySQL базата данни в контекста на разработваното уеб приложение, а view моделите спомагат за конкретизиране на информацията, която да се визуализира пред потребителите след това посредством отделните изгледи.



Фигура 19. Пакет model

"Cooking with Cook Easy is TaStY, EaSY, IntEReStiNg"

3.2.3 Пакет repository

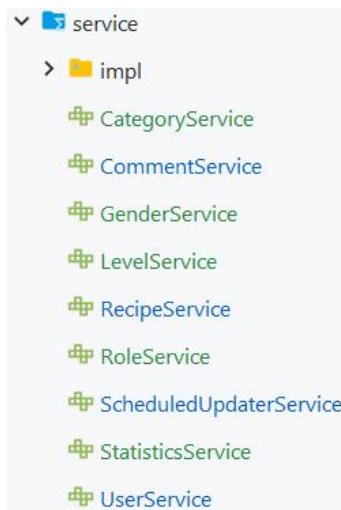
Пакетът repository съдържа всички нужни репозиторита, служещи за извличане на информация от базата данни чрез специализирани JPQL (Java Persistence Query Language) заявки.



Фигура 20. Пакет repository

3.2.4 Пакет service

Пакетът service съдържа интерфейсите и техните съответни имплементации, съхраняващи бизнес логиката на уеб приложението, която описва предлаганата функционалност на системата.

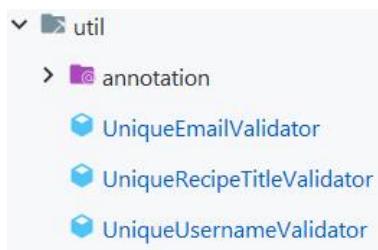


Фигура 21. Пакет service

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

3.2.5 Пакет util

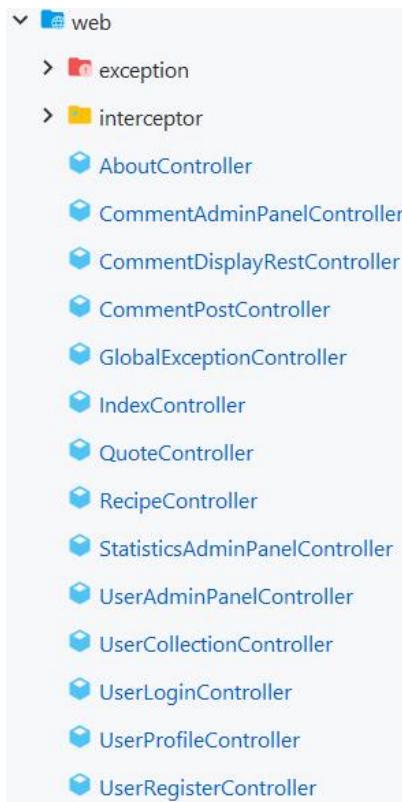
Пакетът util съдържа персонализирани валидатори и съответстващи анотации, които служат за валидиране на определена информация в уеб приложението като потребителски имейл, уникално потребителско име и т.н.



Фигура 22. Пакет util

3.2.6 Пакет web

Пакетът web съдържа всички контролери, персонализирани изключения и интерсептори в контекста на разработваното уеб приложение.



Фигура 23. Пакет web

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

4 Разработка на приложението

В тази точка се представят главните класове и функционалности на уеб приложението.

4.1 ApplicationSecurityConfiguration.java

ApplicationSecurityConfiguration е Java клас, който наследява Spring Boot класа WebSecurityConfigurerAdapter, който осигурява възможност за персонализиране на сигурността на уеб приложението.

Класът ApplicationSecurityConfiguration настройва следните конфигурации за сигурност:

- Разрешаване на достъп до определени URL адреси без необходимост от автентикация: страница за вход в системата, страница за регистрация и страница с повече информация за разработваното приложение;
- Изискване на автентикация за всички останали URL адреси;
- Конфигуриране на страницата за вход в системата и параметрите, необходими за вход в системата;
- Конфигуриране на функционалността *Remember Me* с уникален ключ и период на валидност от една седмица;
- Конфигуриране на URL адреса за излизане от системата и пренасочване към началната страница след излизане от системата;
- Конфигуриране на специфичната имплементация на *UserDetailsService* и енкодера на пароли *Pbkdf2PasswordEncoder*.

```
@Configuration
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class ApplicationSecurityConfiguration extends WebSecurityConfigurerAdapter {
    2 usages
    private final UserDetailsService userDetailsService;
    2 usages
    private final PasswordEncoder passwordEncoder;

    no usages  ↗ Kiara Lazarova
    @Autowired
    public ApplicationSecurityConfiguration(UserDetailsService userDetailsService, PasswordEncoder passwordEncoder) {
        this.userDetailsService = userDetailsService;
        this.passwordEncoder = passwordEncoder;
    }
}
```

Фигура 24. ApplicationSecurityConfiguration.java

"Cooking with Cook Easy is TaSTY, EaSY, InTEReStiNg"

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
        .authorizeRequests() ExpressionUrlAuthorizationConfigurer<...>.ExpressionInterceptUrlRegistry
            .requestMatchers(PathRequest.toStaticResources().atCommonLocations()).permitAll()
            .antMatchers(...antPatterns: "/", "/users/register", "/users/login", "/about").permitAll()
            .antMatchers(...antPatterns: "/**").authenticated()
        .and() HttpSecurity
            .formLogin() FormLoginConfigurer<HttpSecurity>
                .loginPage("/users/login")
                .usernameParameter(UsernamePasswordAuthenticationFilter.SPRING_SECURITY_FORM_USERNAME_KEY)
                .passwordParameter(UsernamePasswordAuthenticationFilter.SPRING_SECURITY_FORM_PASSWORD_KEY)
                .defaultSuccessUrl("/")
                .failureForwardUrl("/users/login-error")
        .and() HttpSecurity
            .rememberMe() RememberMeConfigurer<HttpSecurity>
                .key("uniqueKey1231@#")
                .tokenValiditySeconds(604800)
        .and() HttpSecurity
            .logout() LogoutConfigurer<HttpSecurity>
                .logoutUrl("/users/logout")
                .logoutSuccessUrl("/")
                .invalidateHttpSession(true)
                .deleteCookies(...cookieNamesToClear: "remember-me", "JSESSIONID");
}

// Kiara Lazarova
@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.userDetailsService(this.userDetailsService).passwordEncoder(this.passwordEncoder);
}
```

Фигура 25. ApplicationSecurityConfiguration.java

4.2 Model

Model класовете се делят смислово на няколко различни вида, както следва: binding, service, entity и view.

4.2.1 Binding

Binding класовете представляват програмна конструкция, използвана в обектно-ориентираното програмиране за определяне на връзката между два или повече обекта, при която един обект зависи от друг обект, за да функционира правилно. Binding класовете описват как обектите са свързани и взаимодействат помежду си и често се използва в програмирането на графични потребителски интерфейси (GUI). В контекста на разработваното уеб приложение binding класовете служат за обвързване на компонентите на потребителския интерфейс (например формите за регистрация на потребител и формата за редактиране на вече добавена рецепта) с данните, съхранявани

"Cooking with Cook Easy is TaSTY, EaSY, InTEReStiNg"

в базата данни. В допълнение този вид класове в контекста на разглежданото уеб приложение се използват и за валидиране от страна на сървъра на потребителския вход, преди той да бъде обработен с цел да се гарантира, че данните, получени от съответния клиент, са валидни и отговарят на изискваните критерии. Подобна валидация чрез употреба на binding класове е позната като server-side техника за валидация на грешки. Това е важна мярка за сигурност, която помага за предотвратяване на злонамерени атаки и допринася за подобряване на цялостната надеждност на приложението. Показаният клас UserRegisterBindingModel обвързва формата за регистрация на потребители с MySQL базата данни.

```
@Data
@NoArgsConstructor
public class UserRegisterBindingModel {
    no usages
    private String firstName;
    no usages
    @NotBlank
    @Size(min = 5, max = 12)
    private String lastName;
    no usages
    @NotBlank
    @UniqueUsername
    @Size(min = 5, max = 20)
    private String username;
    no usages
    @NotBlank
    @Email
    @UniqueEmail
    @Size(min = 5, max = 30)
    private String email;
    no usages
    @NotBlank
    @Size(min = 5, max = 20)
    private String password;
    no usages
    @NotNull
    private GenderNameEnum genderNameEnum;
    no usages
    @NotNull
    private LevelNameEnum levelNameEnum;
}
```

Фигура 26. UserRegisterBindingModel.java

"Cooking with Cook Easy is TaSTY, EaSY, InTEReStiNg"

4.2.2 Service

Service класовете представлява слой на абстракция между binding класовете и entity класовете, използван в бизнес логиката. При този тип разделяне на model класовете binding класовете представляват обектите за трансфер на данни (DTO), които се използват за прехвърляне на данни между потребителския интерфейс и слоя с бизнес логиката. От друга страна, entity класовете представляват модела на данните, който се съхранява в базата данни. Поради тази причина service класовете действат като междинен слой между binding класовете и entity класовете. За преобразуване на binding класовете към service такива разработваното приложение използва ModelMapper библиотеката в имплементацията на бизнес логиката. Показаният клас UserRegisterServiceModel представлява абстракция между UserRegisterBindingModel класа и UserEntity класа и се използва в имплементацията на интерфейса UserService.

```
@Data  
@NoArgsConstructor  
public class UserRegisterServiceModel {  
    no usages  
    private String firstName;  
    no usages  
    private String lastName;  
    no usages  
    private String username;  
    no usages  
    private String email;  
    no usages  
    private String password;  
    no usages  
    private GenderNameEnum genderNameEnum;  
    no usages  
    private LevelNameEnum levelNameEnum;  
}
```

Фигура 27. UserRegisterServiceModel.java

4.2.3 Entity

Entity класовете описват отделните обекти, съхранявани в базата данни, и присъщите за тях свойства и характеристики. Приложената част от кода на клас UserEntity показва наличните свойства на дадения клас (и съответно колони в таблицата users) и демонстрира как се задават ограничения на тези свойства / колони в таблицата users в MySQL базата данни. UserEntity класът наследява BaseEntity класа, който бива наследяван и от останалите entity обекти в приложението.

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

```
@Entity
@Table(name = "users")
public class UserEntity extends BaseEntity {
    2 usages
    private String username;
    2 usages
    private String firstName;
    2 usages
    private String lastName;
    2 usages
    private String password;
    2 usages
    private String email;
    2 usages
    private GenderEntity genderEntity;
    2 usages
    private LevelEntity levelEntity;
    2 usages
    private List<RoleEntity> roles = new ArrayList<>();
    2 usages
    private List<RecipeEntity> addedRecipes = new ArrayList<>();
    2 usages
    private List<RecipeEntity> likedRecipes = new ArrayList<>();
    2 usages
    private List<RecipeEntity> savedRecipes = new ArrayList<>();
    2 usages
    private List<RecipeEntity> cookedRecipes = new ArrayList<>();

    :: Kiara Lazarova
    public UserEntity() {
    }

    :: Kiara Lazarova
    @Column(name = "username", length = 20, nullable = false, unique = true)
    public String getUsername() { return this.username; }
}
```

Фигура 28. UserEntity.java

4.2.4 View

View класовете описват данните, които се визуализират в потребителския интерфейс. Тези класове се използват за разделяне на презентационната логика от бизнес логиката в уеб приложението, като осигуряват ниво на абстракция, улесняващо поддръжката и модифицирането на потребителския интерфейс. В заключение view класовете отговарят за извличането на данни от бизнес логиката и за трансформирането им във формат, който може лесно да бъде визуализиран от потребителския интерфейс. Показаният клас StatisticsViewModel описва каква информация относно изпратените заявки в приложението да бъде визуализирана пред потребителите с роля ADMIN.

"Cooking with Cook Easy is TaStY, EaSY, IntEReStiNg"

```
@Data  
@NoArgsConstructor  
public class StatisticsViewModel {  
    no usages  
    private int anonymousRequests;  
    no usages  
    private int authenticatedRequests;  
}
```

Фигура 29. *StatisticsViewModel.java*

4.3 Repository

Repository интерфейсите представляват механизъм, използван за съхраняване, извличане и управление на данни от източник на данни, обикновено база данни. Repository интерфейсите осигуряват абстракция над източника на данни, като енкапсулират логиката за достъп до данни. Spring предоставя няколко начина за имплементиране на repository-та, като в разработваното уеб приложение е използван ORM фреймуърка Hibernate. В Spring repository интерфейсите се дефинират като интерфейс, който наследява базов repository интерфейс. В контекста на разработваното приложение базовия repository интерфейс, използван от всички персонализирани repository интерфейсите в системата, е JpaRepository. Базовият repository интерфейс предоставя основни CRUD, а персонализираните repository интерфейси в уеб приложението дефинират допълнителни методи, специфични за областта на даденото приложението. Показаният интерфейс CommentRepository описва методите за управление на данните в базата данни, свързани с коментарите към публикуваните рецепти.

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

```
@Repository
public interface CommentRepository extends JpaRepository<CommentEntity, Long> {
    1 usage  ↗ Kiara Lazarova
    @Query(value = "SELECT COUNT(c) FROM CommentEntity AS c WHERE c.recipeEntity.id = :recipeId AND c.approved = TRUE AND c.archived = FALSE")
    long findAllByRecipeEntityCount(@Param(value = "recipeId") Long id);

    1 usage  ↗ Kiara Lazarova
    @Query(value = "SELECT c FROM CommentEntity AS c WHERE c.recipeEntity.id = :recipeId AND c.approved = TRUE AND c.archived = FALSE")
    List<CommentEntity> findAllByRecipeEntityAndApprovedTrue(@Param(value = "recipeId") Long recipeId);

    4 usages  ↗ Kiara Lazarova
    @Query(value = "SELECT c FROM CommentEntity AS c WHERE c.recipeEntity.id = :recipeId")
    List<CommentEntity> findAllByRecipeEntity(@Param(value = "recipeId") Long recipeId);

    1 usage  ↗ Kiara Lazarova
    @Query(value = "SELECT c FROM CommentEntity AS c WHERE c.author.id = :authorId")
    List<CommentEntity> findAllByAuthorEntity(@Param(value = "authorId") Long authorId);

    1 usage  ↗ Kiara Lazarova
    List<CommentEntity> findAllByArchivedTrue();
}
```

Фигура 30. CommentRepository.java

4.4 Service

Service класовете се използват за реализиране на бизнес логиката на уеб приложението. Тези класове осигуряват ниво на абстракция между уеб контролерите и repository-тата и отговаря за координирането на достъпа до данни, обработката и други специфични за приложението операции. Добра практика е service класовете да имплементират service интерфейси. Service интерфейсите представляват API, чието предназначение е да дефинира операциите, които съответните service класове трябва да имплементират. Дадените интерфейси осигуряват ясно разделение на отговорността (separation of concerns) между бизнес логиката на приложението и детайлите, свързани с имплементацията на определените service класове. Това означава, че компонентите от по-високо ниво могат да взаимодействат със даден service единствено въз основа на методите, дефинирани в съответния интерфейс, без да е необходима информация за конкретните детайли на имплементация на service-а. По този начин service класът може да бъде актуализиран, без това да се отрази на останалата част от приложението, стига да продължи да осигурява имплементация на наличните в интерфейса методи. Приложената част от кода на уеб приложението показва интерфейса ScheduledUpdaterService и класа ScheduledUpdaterServiceImpl, който представлява имплементацията на горепосочения интерфейс. Предназначенето на дадения интерфейс и неговата реализация е автоматично на всеки 24 часа да изтриват от системата архивираните коментари от потребители с роля ADMIN.

"Cooking with Cook Easy is TaSTY, EaSY, InTEReStiNg"

```
public interface ScheduledUpdaterService {  
    no usages 1 implementation ↗ Kiara Lazarova  
    void scheduleDatabaseUpdate();  
}
```

Фигура 31. *ScheduledUpdaterService.java*

```
@Service  
public class ScheduledUpdaterServiceImpl implements ScheduledUpdaterService {  
    2 usages  
    private final CommentService commentService;  
  
    no usages ↗ Kiara Lazarova  
    @Autowired  
    public ScheduledUpdaterServiceImpl(CommentService commentService) { this.commentService = commentService; }  
  
    no usages ↗ Kiara Lazarova  
    @Override  
    @PostConstruct  
    @Scheduled(cron = "0 0 12 * * ?")  
    public void scheduleDatabaseUpdate() { this.commentService.deleteArchivedComments(); }  
}
```

Фигура 32. *ScheduledUpdaterServiceImpl.java*

4.5 Util

Util (съкратено от utility) класовете съдържат методи, осигуряващи функционалност с общо предназначение в уеб приложението. Тези класове са предназначени да се използват като набор от инструменти или помощни средства за многократна употреба. Чрез енкапсулиране на често използвана функционалност в util класове разработчиците имат възможността да избегнат дублирането на код в цялото приложение и да насырчат преизползването на този код. Освен това util класове могат да помогнат за опростяване на сложните операции и да направят кода по-четим и лесен за поддръжка.

4.5.1 Валидатори

В контекста на разработваното уеб приложение util класовете биват валидатори, които отговарят за валидирането на коректността и последователността на входните данни. Всички валидатори в разработваното приложение имплементират интерфейса ConstraintValidator<A, T>, където A се отнася към типа на анотацията, чрез която да се окаже необходимостта от валидация на съответното свойство в кода, а T – към типа на

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

валидираното свойство. Показаният клас UniqueRecipeTitleValidator описва правилата за валидиране на наименованието на дадена рецепта.

```
@Component
public class UniqueRecipeTitleValidator implements ConstraintValidator<UniqueRecipeTitle, String> {
    2 usages
    private final RecipeService recipeService;

    no usages  ↗ Kiara Lazarova
    @Autowired
    public UniqueRecipeTitleValidator(RecipeService recipeService) { this.recipeService = recipeService; }

    ↗ Kiara Lazarova
    @Override
    public boolean isValid(String title, ConstraintValidatorContext context) {
        return !title.trim().equals("") && !this.recipeService.isTitleOccupied(title);
    }
}
```

Фигура 33. UniqueRecipeTitleValidator.java

4.5.2 Анотации

Персонализираните анотации за валидиране се използват за валидиране на данни в разработваното приложение. Чрез дефинирането на собствени анотации се осигурява декларативен начин за определяне на правилата за валидиране и улеснява четенето и поддръжката на кода. След дефиниране на персонализирана анотация и валидатор клас, анотацията може да бъде използвана, за да се анотират полета в даденото приложение. Когато се задейства процесът на валидиране, фреймуъркът за валидация използва валидатор класа, за да валидира анотираното поле и да върне всички грешки при валидиране, ако съществуват такива. Показаната анотация UniqueRecipeTitle се използва за валидиране, че полето title в класа RecipeAddBindingModel има уникална стойност в сравнение с наименованията на вече създадените рецепти.

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

```
@Retention(value = RetentionPolicy.RUNTIME)
@Target(value = ElementType.FIELD)
@Constraint(validatedBy = UniqueRecipeTitleValidator.class)
public @interface UniqueRecipeTitle {

    no usages   ↗ Kiara Lazarova
    String message() default "Recipe title must be unique.";
    ↗ Kiara Lazarova
    Class<?>[] groups() default {};
    no usages   ↗ Kiara Lazarova
    Class<? extends Payload>[] payload() default {};
}
```

Фигура 34. UniqueRecipeTitle.java

4.6 Web

4.6.1 Интерсептори

Spring Framework осигурява поддръжка за междинни програми (middleware), които са компоненти, прихващащи и обработващи заявки и отговори между клиента и сървъра. В Spring интерсепторите са вид middleware и представляват Java класове, които могат да прихващат HTTP заявки и отговори и да извършват допълнителна обработка преди и след обработката на заявката от дадения контролер. Интерсепторите могат да се използват за изпълнение на широк спектър от задачи, като например логинг (logging), кеширане и обработка на грешки и т.н. Те могат също така да променят заявката и отговора или да добавят персонализирани хедъри (headers) и атрибути. Показаният клас StatisticsInterceptor имплементира интерфейса HandlerInterceptor, като презаписва метода preHandle(), в чието тяло се извиква методът onRequest() от интерфейса StatisticsService. Горепосоченият интерсептор е предназначен за преброяване на заявките в приложението, направени преди и след автентикация.

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

```
@Component
public class StatisticsInterceptor implements HandlerInterceptor {
    2 usages
    private final StatisticsService statisticsService;

    no usages  ↗ Kiara Lazarova
    @Autowired
    public StatisticsInterceptor(StatisticsService statisticsService) { this.statisticsService = statisticsService; }

    no usages  ↗ Kiara Lazarova
    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {
        this.statisticsService.onRequest();
        return true;
    }
}
```

Фигура 35. *StatisticsInterceptor.java*

4.6.2 Контролери

MVC контролерите в Spring са отговорни за обработката на входящите HTTP заявки и за съответното актуализиране на модела и изгледа. Те получават заявки от клиента, взаимодействат с модела, за да изпълнят необходимата бизнес логика, и след това връщат отговор на клиента чрез съответния изглед. В Spring MVC контролерите се реализират като класове, които са анотирани с анотацията `@Controller`. Тези класове съдържат методи за обработка на заявки, които са анотирани с анотация спрямо целите на метода (`@GetMapping`, `@PostMapping` и т.н.). Тези анотации над методите се използват, за да се посочи URL мапинга на дадения метод за обработка на заявки. Spring MVC контролерите предлагат възможност за използване на анотации като `@ModelAttribute` и `@RequestParam` с цел извличане на данни от заявката и предаване на вече извлечените данни на метода за обработка на заявки. Показаният метод `editRecipe()` от класа `RecipeController` е пример за реализация на метод за обработка на `patch` заявка, която актуализира дадена рецепта спрямо уникалния идентификатор на определената рецепта. Анотацията `@PreAuthorize` служи за авторизация на текущия потребител, който се опитва да редактира съответната рецепта. В случая, ако този потребител не е автор на посочената рецепта, то тогава той няма достъп до нейното актуализиране.

"Cooking with Cook Easy is TaStY, EaSY, InTEReSTiNg"

```
@PreAuthorize("@recipeServiceImpl.isRecipeOwner(#principal.name, #id)")
@PatchMapping(value = "/recipes/{id}/details/edit")
public String editRecipe(@PathVariable(name = "id") Long id,
                        @Valid RecipeEditBindingModel recipeEditBindingModel,
                        BindingResult bindingResult,
                        RedirectAttributes redirectAttributes,
                        Principal principal) {
    if(bindingResult.hasErrors()) {
        redirectAttributes.addFlashAttribute("editBindingModel", recipeEditBindingModel);
        redirectAttributes.addFlashAttribute("org.springframework.validation.BindingResult.editBindingModel", bindingResult);

        return "redirect:/recipes/" + id + "/details/edit";
    }

    RecipeEditServiceModel recipeEditServiceModel = this.modelMapper.map(recipeEditBindingModel, RecipeEditServiceModel.class);

    this.recipeService.editRecipe(id, recipeEditServiceModel);

    return "redirect:/recipes/" + id + "/details";
}
```

Фигура 36. Метод `editRecipe()`, част от класа `RecipeController.java`

4.7 Test

4.7.1 Компонентни тестове

Компонентните тестове в разработваното приложение са реализирани с помощта на JUnit Jupiter и Mockito, като се спазва AAA (Arrange – Act – Assert) моделът в следната последователност: инициализация на необходимите данни за извършване на съответния тест и подготвяне на тестовия обект (Arrange), извикване на тествания метод или свойство (Act) и сравняване на получения резултат с очаквания такъв (Assert). Показаният метод в контекста на компонентното тестване `testUserByUsernameShouldReturnUserDetailsObj()` (част от класа `CookEasyUserServiceTest`) проверява дали методът `loadUserByUsername()` в интерфейса `CookEasyUserService` връща успешно желания `UserDetails` обект.

"Cooking with Cook Easy is TaSTY, EaSY, InTEReStiNg"

```
@Test
public void testLoadUserByUsernameShouldReturnUserDetailsObj() {
    Mockito
        .when(this.userRepository.findByUsername(this.user.getUsername()))
        .thenReturn(Optional.of(this.user));

    UserDetails actualUser = this.cookEasyUserService.loadUserByUsername(GlobalTestConstants.USERNAME);

    String actualRoles = actualUser
        .getAuthorities()
        .stream()
        .map(GrantedAuthority::getAuthority)
        .collect(Collectors.joining(", "));

    Assertions.assertEquals(this.user.getUsername(), actualUser.getUsername());
    Assertions.assertEquals(GlobalTestConstants.EXPECTED_ROLES, actualRoles);
}
```

Фигура 37. Метод `testUserByUsernameShouldReturnUserDetailsObj()`, част от класа `CookEasyUserServiceImplTest.java`

4.7.2 Интеграционни тестове

Интеграционните тестове в разработваното приложение са реализирани с помощта на Spring Boot Test, класа MockMvc (част от Spring Test) и класа SecurityMockMvcRequestPostProcessors (част от Spring Security). Благодарение на употребата на класа MockMvc позволява тестване на уеб слоя на приложението, без то да се стартиране на уеб сървъра. Така този клас осигурява начин за симулиране на HTTP заявки към приложението и проверка на получените отговори. Класът SecurityMockMvcRequestPostProcessors предоставя статични factory методи (такива, които създават обект и го връщат) за създаване на инстанции на интерфейса RequestPostProcessor с цел симулиране на различни тестови случаи на автентификация и авторизация в интеграционните тестове на уеб приложението. Тези инстанции на RequestPostProcessor могат да се използват заедно с MockMvc, за да се добавят допълнителни детайли за автентификация и авторизация към заявките, изпращани към сървъра по време на тестването. В тялото на приложения метод `testRegisterAndLoginUserShouldRegisterAndLoginUserWithValidCredentials()` (част от класа UserRegisterControllerTest) методът SecurityMockMvcRequestPostProcessors.csrf() създава RequestPostProcessor, добавящ CSRF жетон (token) към заявката, необходим за всяка POST заявка към защитена от Spring Security крайна точка (endpoint).

"Cooking with Cook Easy is TaSTY, EaSY, InTEReStiNg"

```
@Test
public void testRegisterAndLoginUserShouldRegisterAndLoginUserWithValidCredentials() throws Exception {
    this.mockMvc.
        perform(post( urlTemplate: "/users/register").
            param( name: "firstName", GlobalTestConstants.FIRST_NAME).
            param( name: "lastName", GlobalTestConstants.LAST_NAME).
            param( name: "username", GlobalTestConstants.USERNAME).
            param( name: "email", GlobalTestConstants.EMAIL).
            param( name: "password", GlobalTestConstants.PASSWORD).
            param( name: "confirmPassword", GlobalTestConstants.PASSWORD).
            param( name: "genderNameEnum", GlobalTestConstants.GENDER_NAME_ENUM).
            param( name: "levelNameEnum", GlobalTestConstants.LEVEL_NAME_ENUM).
            with(csrf()).
            contentType(MediaType.APPLICATION_FORM_URLENCODED)
        ).
        andExpect(status().is3xxRedirection());
}

Optional<UserEntity> optionalRegisteredAndLoggedUser = this.userRepository.findByUsername(GlobalTestConstants.USERNAME);

Assertions.assertTrue(optionalRegisteredAndLoggedUser.isPresent());

UserEntity registeredAndLoggedUser = optionalRegisteredAndLoggedUser.get();

Assertions.assertEquals(GlobalTestConstants.FIRST_NAME, registeredAndLoggedUser.getFirstName());
Assertions.assertEquals(GlobalTestConstants.LAST_NAME, registeredAndLoggedUser.getLastName());
Assertions.assertEquals(GlobalTestConstants.USERNAME, registeredAndLoggedUser.getUsername());
Assertions.assertEquals(GlobalTestConstants.EMAIL, registeredAndLoggedUser.getEmail());
Assertions.assertEquals(GlobalTestConstants.GENDER_NAME_ENUM, registeredAndLoggedUser.getGenderEntity().getGenderNameEnum().name());
Assertions.assertEquals(GlobalTestConstants.LEVEL_NAME_ENUM, registeredAndLoggedUser.getLevelEntity().getLevelNameEnum().name());
}
```

*Фигура 38. Метод
testRegisterAndLoginUserShouldRegisterAndLoginUserWithValidCredentials(), част от
класа UserRegisterControllerTest.java*

5 Ръководство на потребителя

5.1 Стартоване на приложението

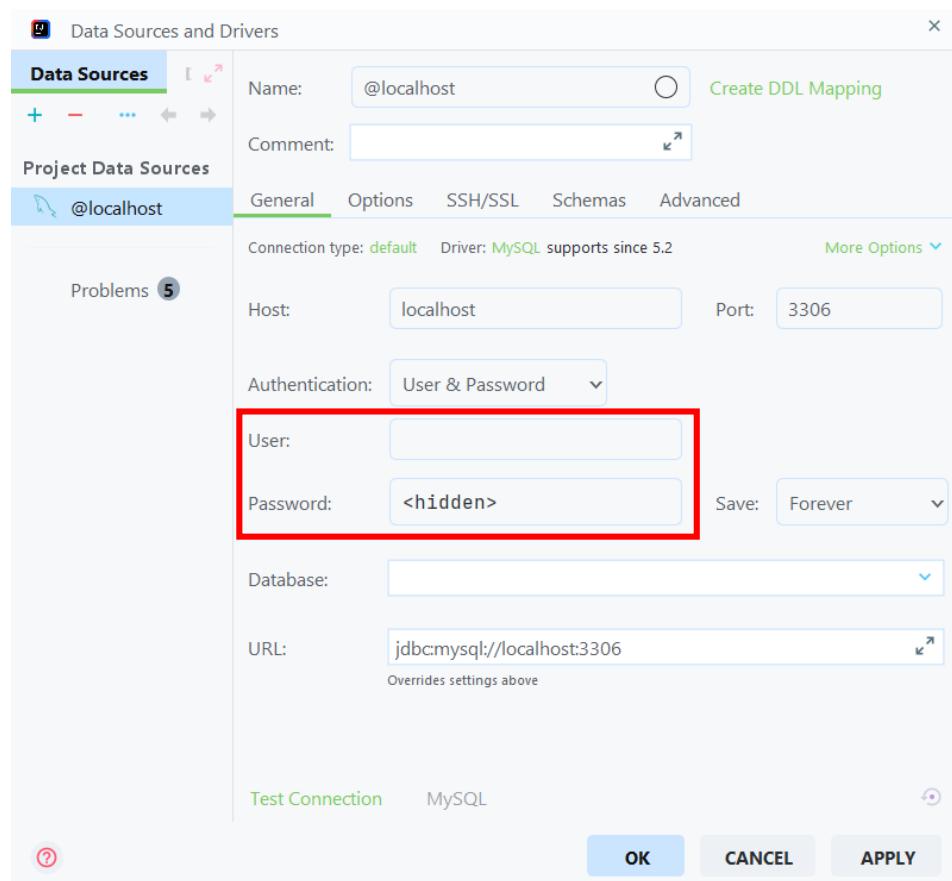
За да се стартира, приложението се нуждае от няколко предварително локално изтеглени технологии: Java 15 или по-нова версия, MySQL и IntelliJ IDEA Ultimate. След успешно инсталиране на горепосочените технологии следва да се изпълнят няколко необходими стъпки с цел стартиране на уеб приложението:

1. Във файла *application.properties*, намиращ се в директорията *src.main.resources*, трябва да бъде зададена подходяща стойност на свойствата *spring.datasource.username* и *spring.datasource.password* спрямо credential информацията на клиента, опитващ се да стартира приложението.

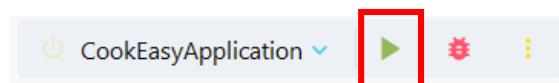
```
spring.datasource.username = root
spring.datasource.password = root
```

"Cooking with Cook Easy is TaSTY, EaSY, InTEReStInG"

2. В дясното странично поле в IntelliJ IDEA Ultimate е разположен бутона *Database*. След натискане на посочения бутона се отваря меню с различни опции. В горния ляв ъгъл има друг бутона със знак „+“. След натискане на бутона със знак „+“ се появява dropdown меню, от което се избира опцията *Data Source => MySQL* и отново се въвеждат потребителското име и паролата спрямо credential информацията на клиента, опитващ се да стартира приложението.



3. След успешно изпълняване на стъпка №1 и №2 от навигационната лента се натиска зеленият триъгълен бутона или вместо това се изпълнява клавишината комбинация Shift + F10.



Изчаква се докато в конзолата в IntelliJ IDEA Ultimate се изпише *Started CookEasyApplication*. При успешно стартиране на уеб приложението трябва да се покаже началната страница на приложението на адрес <http://localhost:8080/>.

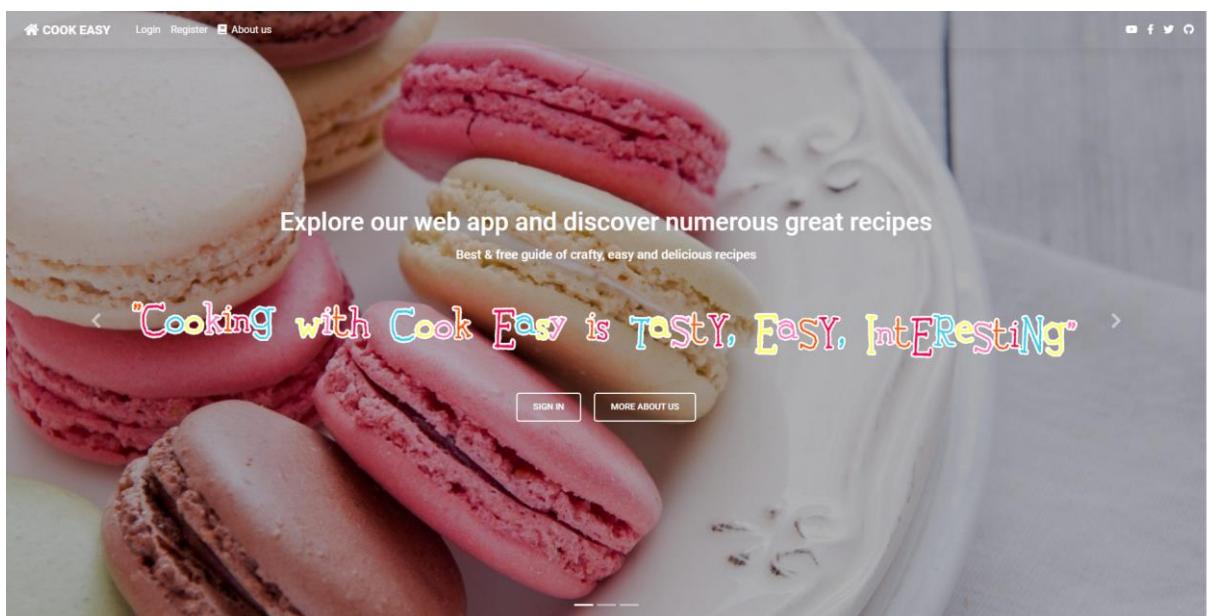
"Cooking with Cook Easy is TaStY, EaSY, IntEReSTiNg"

5.2 Изгледи

В разработваното уеб приложение за визуализиране на информацията пред потребителя на помощ идват така наречените изгледи, или още view-та. Част от тези изгледи са следните:

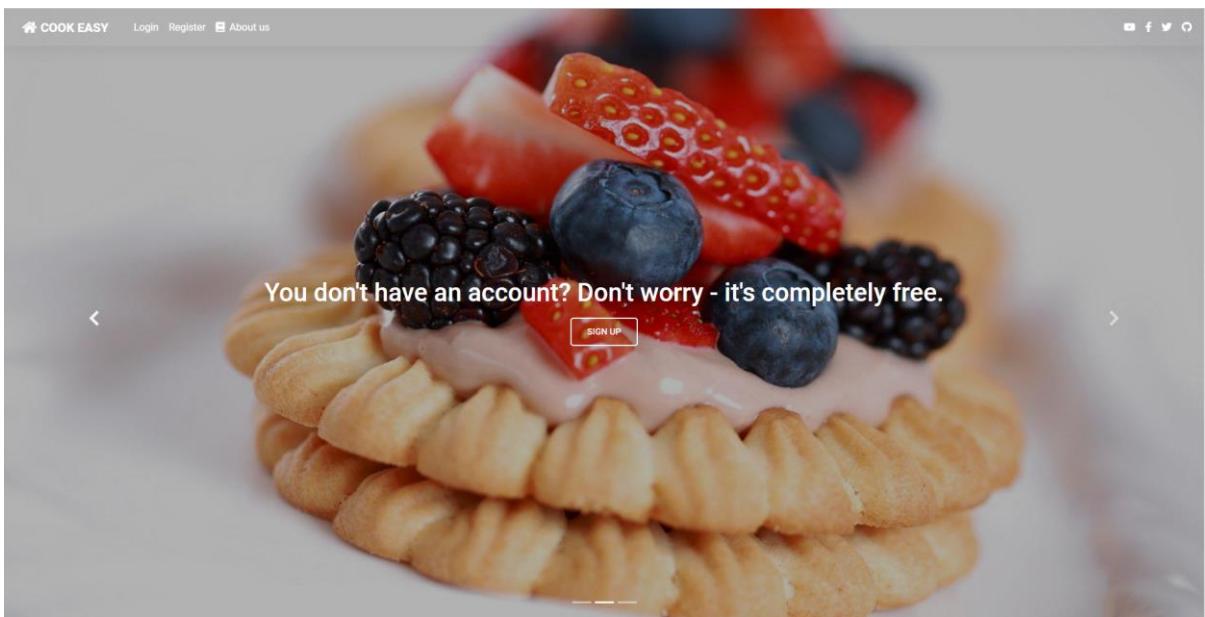
5.2.1 Начална страница за guest потребители

В разработваното приложение началната страница се различава за потребители, преминали през успешна автентикация и такива, които съответно не са. Приложената снимка показва какво представлява началната страница за guest потребители – т.е. тези потребители, които не са влезнали в системата.

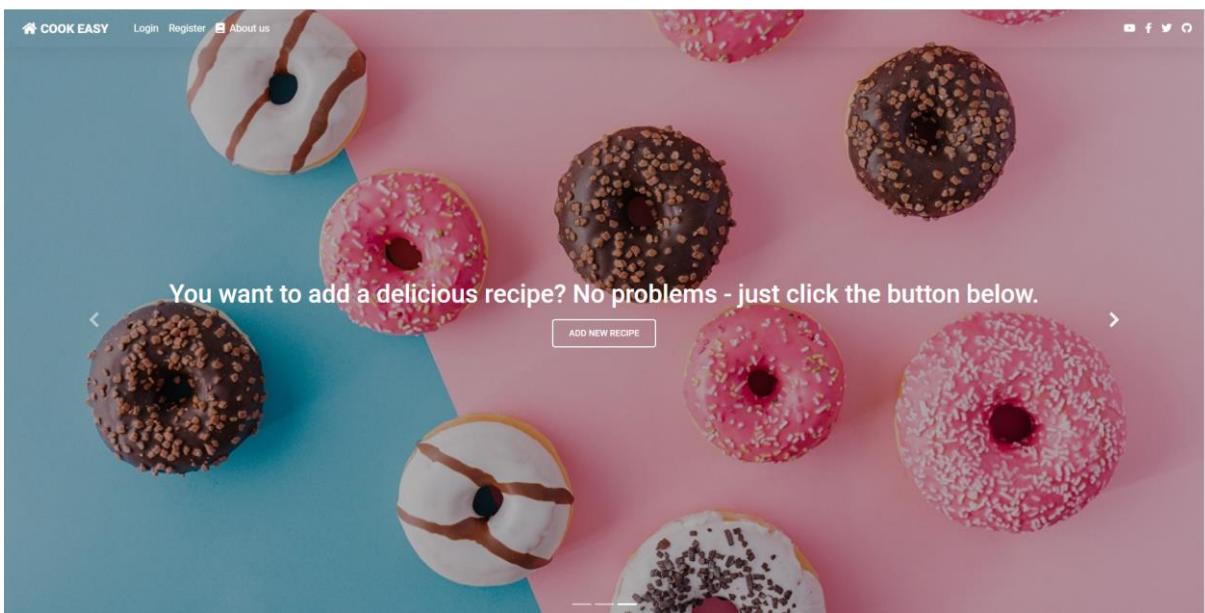


Фигура 39. Начална страница за guest потребители - слайд №1

"Cooking with Cook Easy is TaStY, EaSY, InTEReSTiNg"



Фигура 40. Начална страница за guest потребители - слайд №2



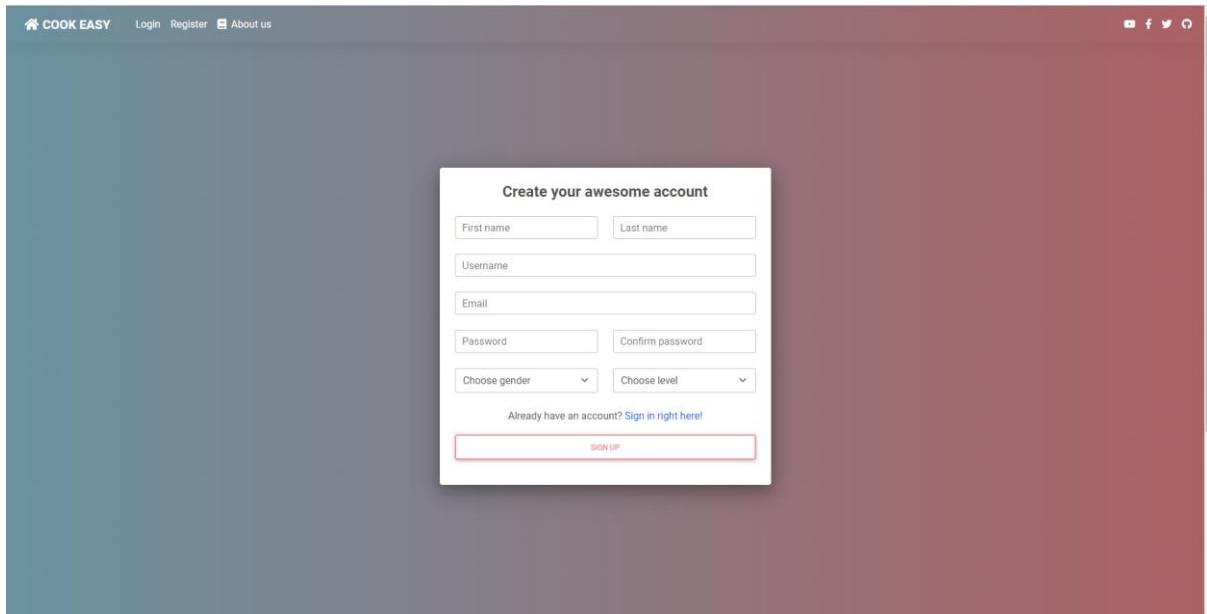
Фигура 41. Начална страница за guest потребители - слайд №3

5.2.2 Форма за регистрация на потребител

Формата за регистрация на потребител съдържа различни полета от различни типове. За да се регистрира успешно в системата даденият потребител, е необходимо да се въведе стойност във всяко едно поле, като обаче полето First Name не е задължително. Въведените от потребителя стойности преминават през валидация преди да бъдат изпратени към сървъра, като съответно в случай на невалидна информация се визуализира подходящо съобщения за допуснатата грешка във формата на въведена

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

стойност. При регистриране на нов потребител, системата автоматично логва в уеб приложението даденият потребител.

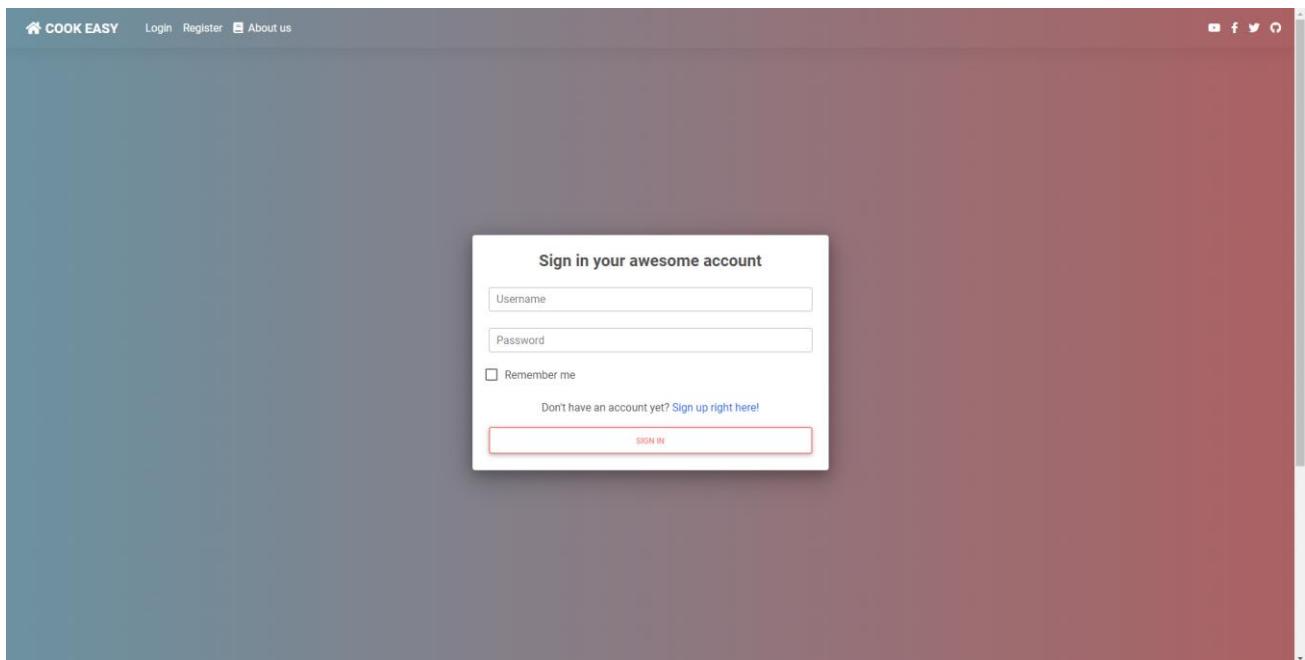


Фигура 40. Форма за регистрация на потребител

5.2.3 Форма за влизане в потребителски акаунт

Подобно на формата за регистрация на потребител, така и формата за вход в системата съдържа различни полета, които обаче са от един и същи тип. Двете полета биват username и password. За да влезе даденият потребител успешно в системата, е необходимо да се въведе правилна credential информация за съответния потребител, преминал първо успешно през регистрация) и в двете полета – username и password.

"Cooking with Cook Easy is TaStY, EaSY, IntEReStiNg"



Фигура 41. Форма за влизане в потребителски акаунт

5.2.4 Администраторски изглед за одобряване и архивиране на създадени коментари към дадена рецепта

Администраторският изглед в разработваното уеб приложение предоставя 3 различни опции: редактиране на потребителите в системата, редактиране на коментарите към дадена рецепти и преглед и преглед на статистиката относно направените заявки в системата. Приложената снимка показва какво представлява изгледът за редактиране на коментарите към дадена рецепта. Разбира се, достъп до администраторския изглед и опции, които той предлага, имат единствено администраторите в системата.

"Cooking with Cook Easy is TaSTY, EaSY, IntEReStiNg"

The screenshot shows the Cook Easy website's administrative section. At the top, there is a navigation bar with links for 'Add new recipe', 'View collections', 'Profile', 'Admin', 'About us', and 'Quote'. A welcome message 'WELCOME, MS. LAZAROVA!' is displayed. On the right side of the header, there are social media icons for YouTube, Facebook, Twitter, and LinkedIn. The main content area is titled 'Edit comments' and contains a message: 'You want to update user roles or delete users? No problem - just click the button below.' Below this is a 'EDIT USERS' button. At the bottom of the page, there is a yellow 'IMPORTANT' note: 'Archived comments are automatically deleted on every 24 hours.'

AUTHOR USERNAME	RECIPE TITLE	CREATED ON	CONTENT		
@ivan_lazarov	Ivan's Homemade Hot Chocolate	2023-04-20	My recipe is AMAZING 🍫!!! The result will amaze you. This homemade hot chocolate is super delicious and it is absolutely perfect for Christmas! You should definitely try it - it is super easy to make! Happy holidays ️!!! X O X O 🎅	<button>APPROVE</button>	<button>ARCHIVE</button>
@daniel_lazarov	Daniel's Perfect Scrambled Eggs	2023-04-20	My recipe is AMAZING 🍳!!! The result will amaze you. These scrambled eggs are super delicious and they are absolutely perfect for your breakfast! You should definitely try them - they are not so hard to make! Happy holidays ️!!! X O X O 🎅	<button>APPROVE</button>	<button>ARCHIVE</button>
			My recipe is AMAZING 🍰!!! The result will amaze you. This vanilla cake		

Фигура 42. Администраторски изглед за одобряване и архивиране на коментари към дадена рецепта