ДИПЛОМЕН ПРОЕКТ ЗА ДЪРЖАВЕН ЗРЕЛОСТЕН ИЗПИТ

по професия код 481030 „Приложен програмист“

специалност код 4810301 Приложно програмиране“

Тема: „Козметичен мениджър“

Автор:

Мариета Петкова Стойчева, клас XII Г

Ръководител:

Михаела Михайлова

Бургас

СЪДЪРЖАНИЕ

[1 Увод 2](#_Toc163066315)

[2 Цели и обхват на софтуерното приложение 6](#_Toc163066316)

[3 Анализ на решението 7](#_Toc163066317)

[3.1 Потребителски изисквания и работен процес 9](#_Toc163066318)

[3.2 Примерен потребителски интерфейс 12](#_Toc163066319)

[3.3 Диаграми на анализа 16](#_Toc163066320)

[3.4 Модел на съдържанието / данните 16](#_Toc163066321)

[4 Дизайн 21](#_Toc163066322)

[4.1 Реализация на архитектурата на приложението 22](#_Toc163066323)

[4.2 Описание на слоевете, предназначението им, библиотеки и методи включени в съответния слой. 23](#_Toc163066324)

[4.3 Организация и код на заявките към база от данни 28](#_Toc163066325)

[4.4 Наличие на потребителски интерфейс (конзолен, графичен, уеб) 30](#_Toc163066326)

[5 Ефективност и бързодействие на решението 32](#_Toc163066327)

[6 Тестване 33](#_Toc163066328)

[7 Заключение и възможно бъдещо развитие 37](#_Toc163066329)

[8 Използвани литературни източници и Уеб сайтове 38](#_Toc163066330)

[9 Приложения 40](#_Toc163066331)

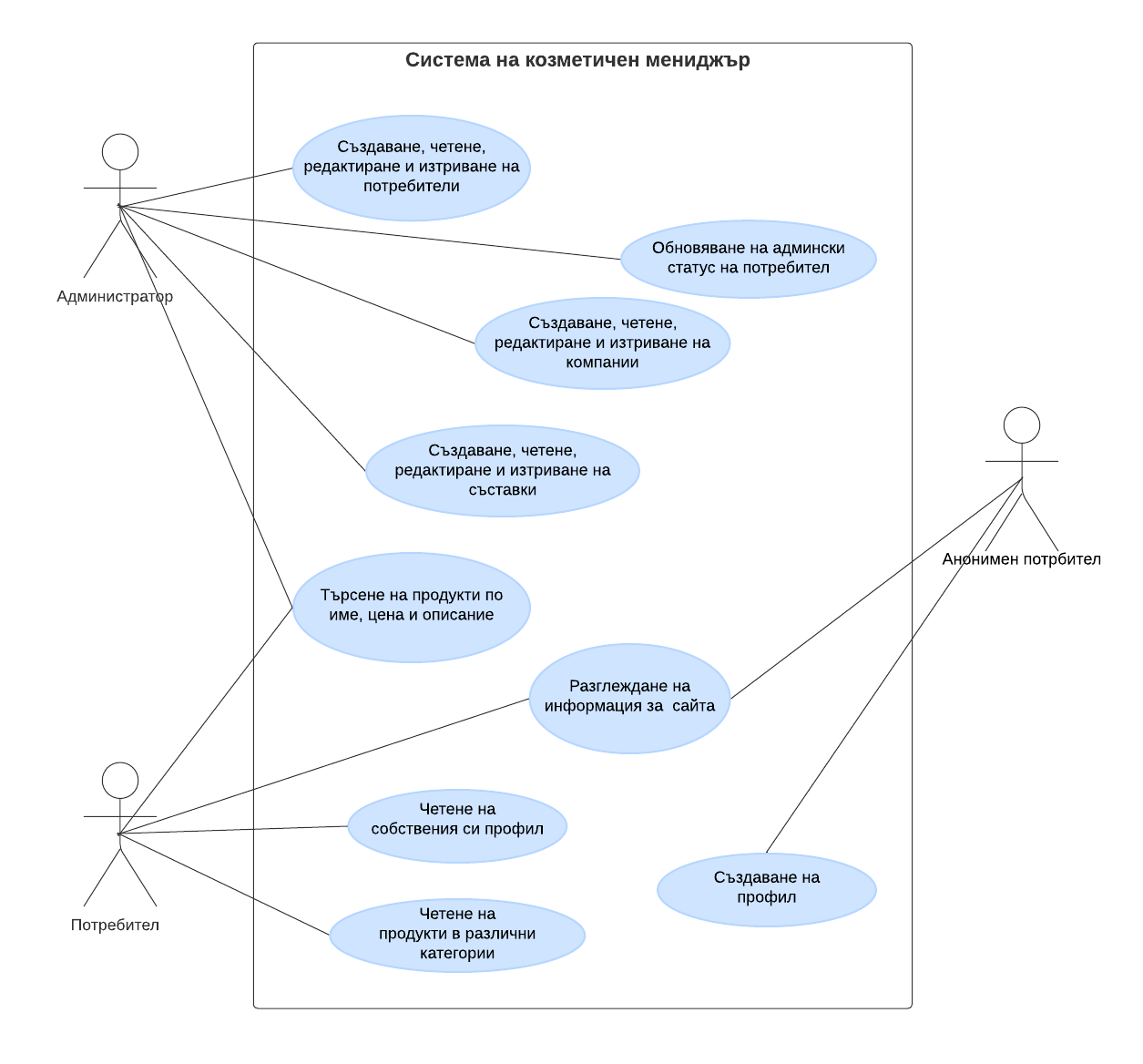
# Увод

През последното десетилетие индустрията на козметиката е преживяла значителен напредък, като нови технологии, иновации във формулите и нарастващата осведоменост на потребителите са допринесли за разнообразието и качеството на продуктите.

Въпреки това, в този разцвет все още липсва пълното разбиране и внимание към индивидуалните нужди и специфики на всеки човек. Козметиката се ориентира към масово производство и стандартизирани решения, които не винаги отговарят на уникалните изисквания на всеки клиент.

В този контекст се вписва проектът, който се стреми да попълни тази празнота и да предложи персонализиран подход към грижата за кожата и косата. Целта на проекта е да изгради иновативна платформа, която комбинира технологични решения и експертно знание в областта на козметиката, за да предостави на потребителите индивидуализирани грижи, съобразени с техните уникални нужди и предпочитания.

Създаването на този проект е отговор на нарастващата нужда от персонализирани подходи в сферата на козметиката. Той се основава на доказването, че кожата и косата на всяка индивидуалност изискват уникални грижи, които не винаги могат да бъдат задоволени от стандартни продукти и процедури.



Диаграмата показва система на козметичен мениджър, като на нея може да видим, различните нива на достъп.

Потребителите на ниво администратор и нормален потребител притежават достъп до всички продукти, тяхното описание като основната им съставка, цената им и специфичния сезон, към който принадлежат. Анонимните потребители могат само да създадат акаунт и да разгледат информацията за системата.

Работата на администраторите ще се състои в това да преглеждат информацията за всеки потребител и да дават съответните права, ако е необходимо. Трябва да преглеждат също така и съобщенията, които са им изпратени от вече регистрираните потребители. Имат задължението да вкарват нови продукти като информацията за тях трябва да е точна и да кореспондира на реалността. Продукт, подходящ за зимата и се слага на устните, трябва да бъде поставен в съответната си категория. Като стане объркване е длъжен да редактира или ако е нужно да изтрие определен продукт. Администраторът притежава и правата да създават, четат, редактират и изтриват и компании, и съставки.

До системата имат достъп потребителите, които се информират за най-подходящите за тях козметични продукти, администраторите, които управляват козметиката и системата.

След като потребителите се регистрират в козметичния мениджър им се прави акаунт с номер, потребителско име(username), двете им имена, имейл и парола, като администраторът има информация до всичко без паролата.

# Цели и обхват на софтуерното приложение

Идеята за създаването на софтуерното приложение "Козметичен мениджър" произтича от осъзнаването на липсата на индивидуализирани решения в индустрията на козметиката. В наши дни, козметичните продукти се произвеждат масово и често се предлагат стандартизирани решения, които не винаги отговарят на уникалните нужди на всеки човек. Тази липса на персонализация може да доведе до неприятни реакции като акне или алергични реакции при използване на неподходящи продукти. Целта на "Козметичен мениджър" е да предостави индивидуализирано внимание към нуждите на всеки клиент, като гарантира, че те се чувстват красиви и уверени без риск от неприятности.

Със своята важност в обществото, приложението отговаря на нарастващата нужда от грижа за себе си и кожата. Във време, когато визията става все по-важен фактор за самочувствие и социалност, "Козметичен мениджър" се насочва към правилната аудитория, като предлага решения, които не само правят потребителите по-привлекателни, но и ги предпазват от възможни неприятности.

В ролята си, администраторите на приложението са упълномощени да вписват и редактират данни за потребителите, гарантирайки сигурността и точността на предоставената информация. Това осигурява на потребителите достъп до надеждни и персонализирани козметични решения, които отговарят на техните индивидуални изисквания и предпочитания.

# Анализ на решението

В системата на козметичния мениджър, есенциално е качеството и достоверността на информацията, която се предоставя на потребителите. Това е от критично значение, защото козметичните продукти пряко взаимодействат с кожата и косата на хората, и качеството им може да има значително влияние върху здравето и външния вид. Всички организации и експерти, които предоставят информация, свързана с козметиката, трябва да бъдат изключително компетентни и имащи надеждност в своите данни и препоръки.

Приложението работи в тясно сътрудничество с такива организации и експерти, за да осигури, че всички предлагани продукти са подложени на стриктни проверки на експертно ниво. Тези проверки включват оценка на съставките, клинични изпитвания за ефективност и безопасност, както и съобразяване със съответните регулации и стандарти за качество.

Важно е също така, козметичният мениджър да поддържа актуална информация за всички предлагани продукти, включително данни за съставките, инструкции за употреба и всякакви предупреждения относно възможни алергени или нежелани реакции. Това осигурява на потребителите достъп до пълна и надеждна информация, което им позволява да вземат информирани решения относно грижата за кожата и косата си.

В същото време, козметичният мениджър приложението поддържа открита комуникация с потребителите, като им предоставя възможност за оценка и обратна връзка относно предлаганите продукти. Това помага да се осигури постоянно подобрение на услугите и продуктите, като се отчитат реалните нужди и предпочитания на клиентите. Потребителите трябва да са напълно наясно с това какво се случва с тяхната информация и тази на продуктите.

Като заключение, за да се осигури най-добрата електронна информация чрез приложението, е от съществено значение да се придържаме към най-високите стандарти на професионална практика в индустрията на козметиката. Това включва непрекъснато следене на правилата и регламентите, свързани с качеството и безопасността на козметичните продукти.

Освен това, е от съществено значение да се гарантира сигурността и защитата на личните данни на потребителите. Приложението трябва да спазва всички законови изисквания за защита на данните и да прилага най-съвременните методи за криптиране и защита на информацията на потребителите.

За да се осигури достъпност и удобство за потребителите, е важно да се изберат правилните методи за събиране и анализ на информацията за кожата и косата на потребителите. Това може да включва използването на сензорни технологии или възможността за въвеждане на данни ръчно от потребителите.

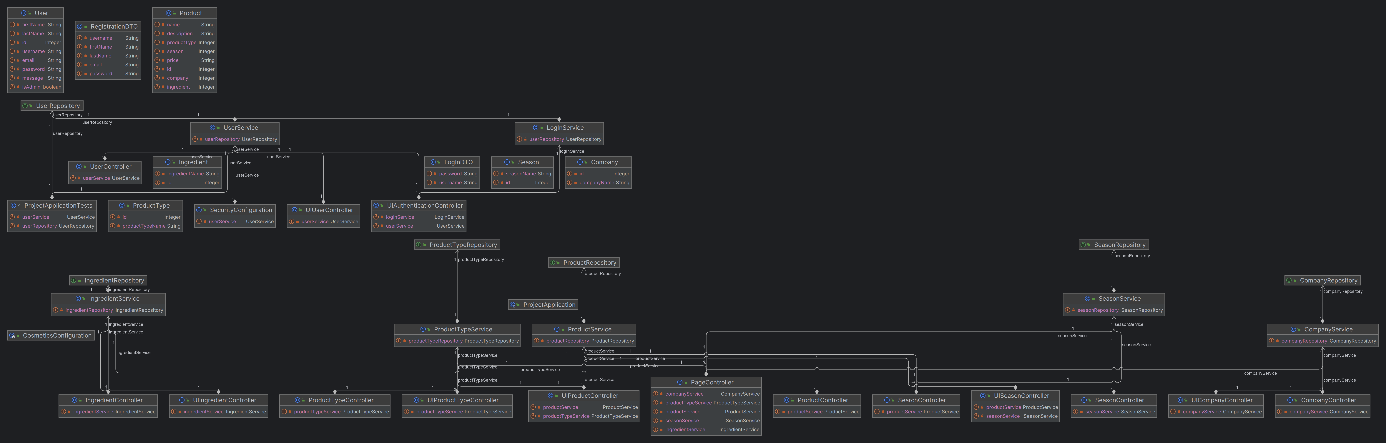
Накрая, е от съществено значение да се осигури удобен и сигурен начин за доставка на козметичната информация до потребителите. Приложението трябва да предостави ясни и лесно разбираеми инструкции и съвети за грижа за кожата и косата, като същевременно осигурява възможност за персонализирани препоръки и продукти, които са съобразени с индивидуалните нужди на всеки потребител. Така се гарантира, че потребителите получават надеждна и качествена козметична информация, която им помага да се грижат за здравето и красотата си с увереност.

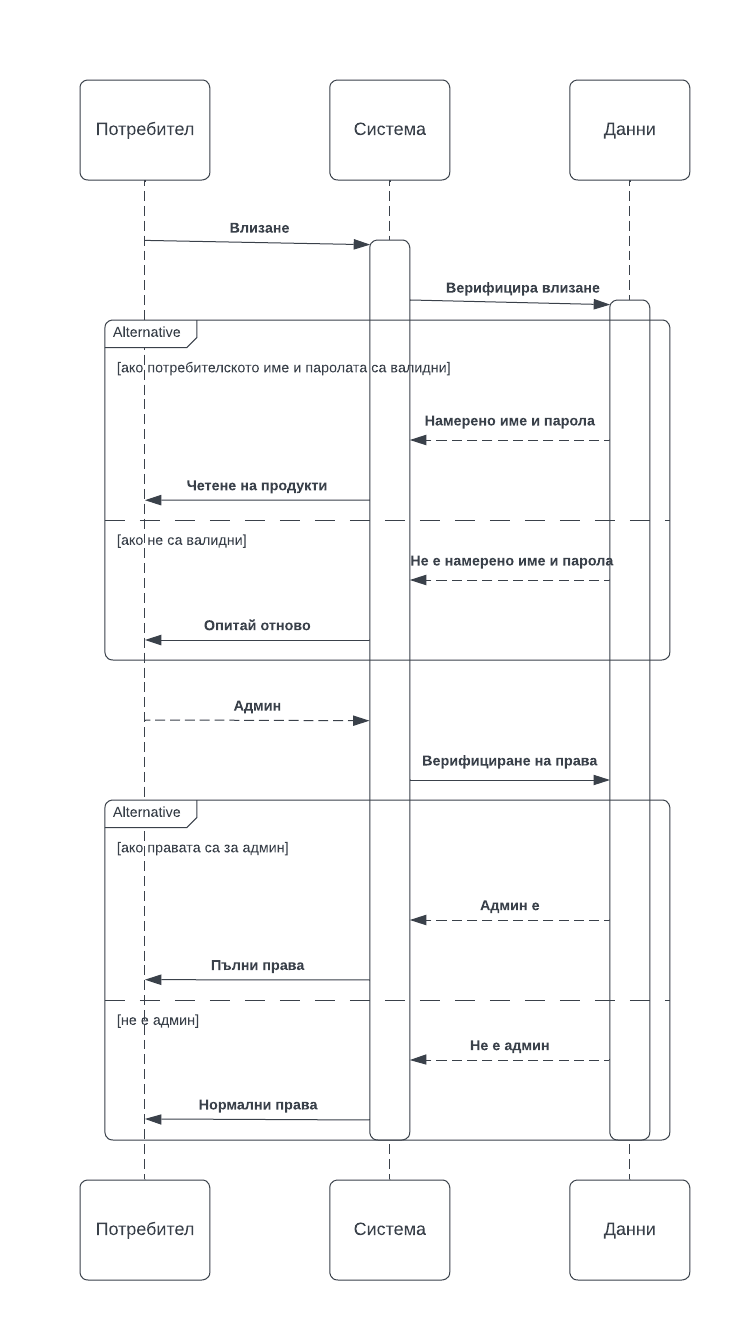
## Потребителски изисквания и работен процес

Входното съдържание за козметичния мениджър включва информация, свързана с козметични продукти, потребители и техните предпочитания. Тази информация може да бъде получена от различни източници, включително от администраторите на системата, потребителите и външни доставчици на данни за продуктите. Например, данните за нови козметични продукти могат да бъдат въведени от администраторите, докато предпочитанията на потребителите могат да бъдат събрани чрез анкети или интеракции в приложението.

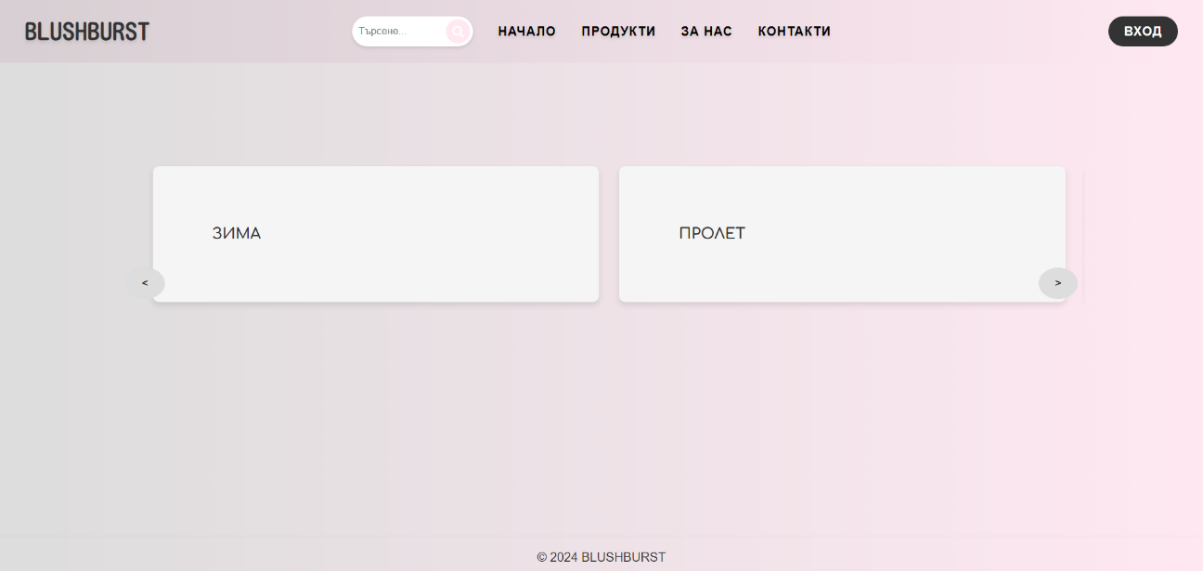
След като входната информация е събрана, тя се обработва и запазва в системата на козметичния мениджър. Обработката може да включва валидация на данните, извличане на съществена информация и свързване на данните със съответните обекти в системата. Например, информацията за нов продукт може да бъде валидирана за съответствие със зададените стандарти за качество и безопасност, след което да бъде свързана със съответната компания и категория на продукта. След обработката, данните се запазват в базата данни на системата за по-нататъшно използване.

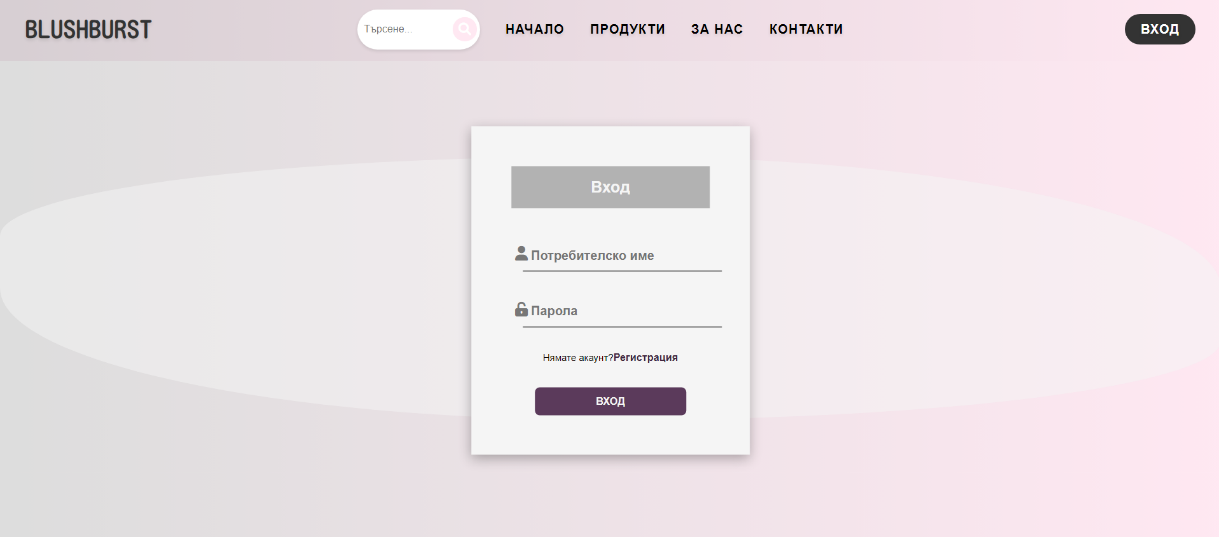
Изходът от работния процес на козметичния мениджър включва предоставянето на персонализирани препоръки и информация за козметични продукти на потребителите. Тази информация може да бъде предоставена през уеб интерфейса на приложението или чрез мобилно приложение. Потребителите могат да използват тази информация за вземане на информирани решения относно покупките си на козметични продукти и за подобряване на грижата за кожата и косата си. Освен това, системата може да предоставя и аналитични данни и отчети за администраторите, които да им помогнат в управлението и оптимизацията на процесите в системата.

Този работен процес осигурява ефективно управление на информацията за козметичните продукти и предоставя персонализирани решения и препоръки на потребителите, като така допринася за подобряване на опита им с приложението.



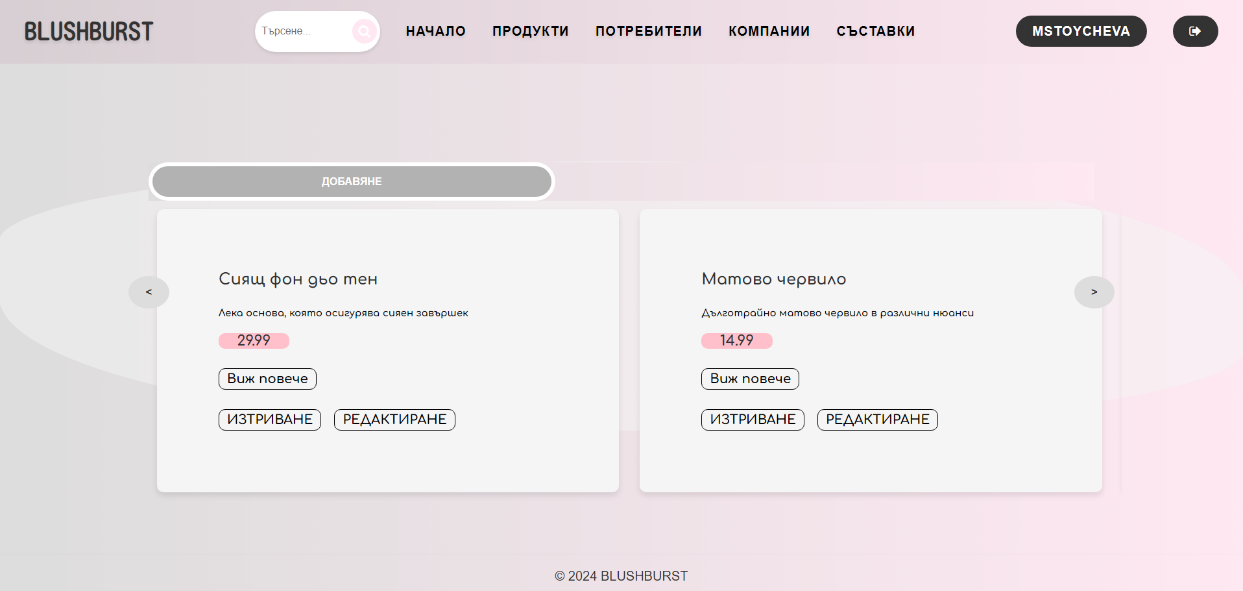
## Примерен потребителски интерфейс



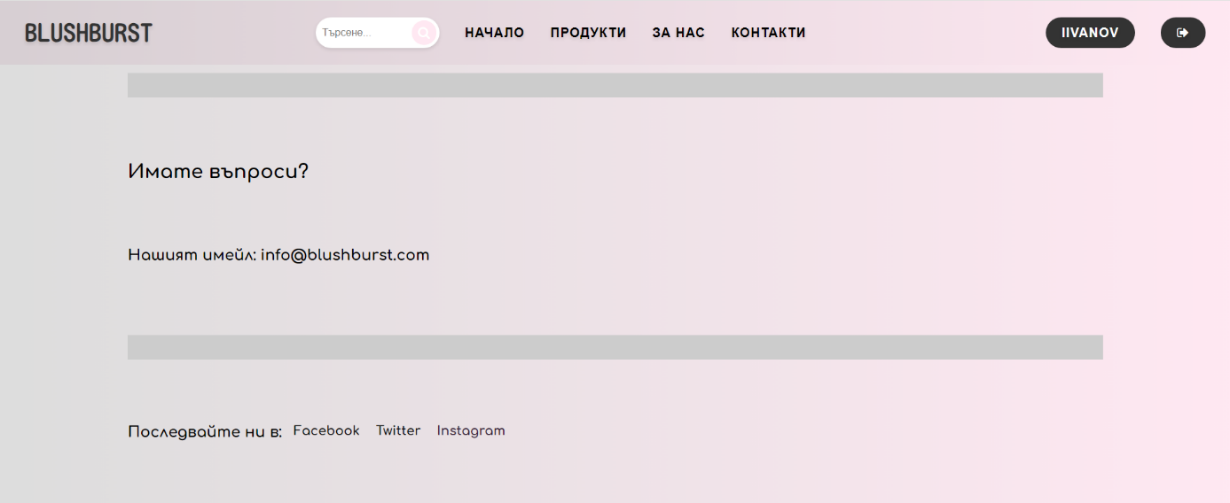


A screenshot of a computer

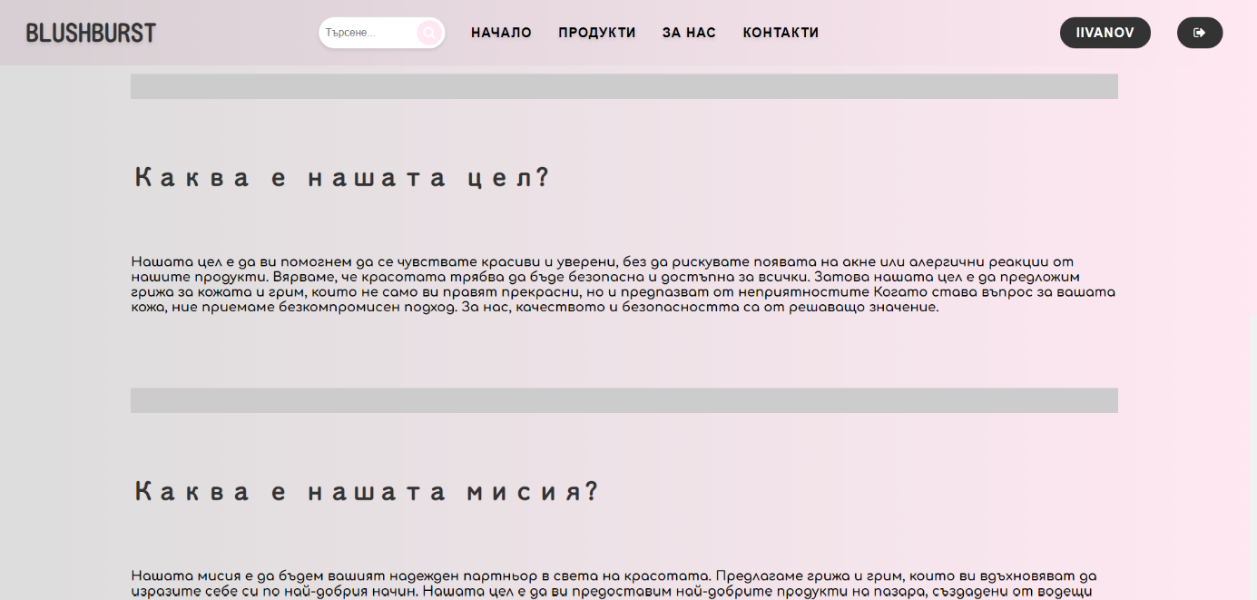
Description automatically generated



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Интерфейсът на проекта включва няколко страници, които предоставят информация и функционалности за различните типове потребители в системата.

На началната страница като потребител може да се видят сезоните в годината. За всеки един от тях има определени продукти и когато се натисне някой от сезоните излизат тези продукти. Може да се види и търсачката, която работи по име, описание и цена на продукта.

Другите страници са свързани с регистрацията на нови потребители или влизането им в системата. В тези страници, потребителите могат да въведат информация за себе си като нови записи и да я запазят в базата данни на системата.

За всеки от продуктите, съставките им и компаниите, съществува индивидуална страница със специално предназначена форма, която се попълва от администратора на системата. Тази форма позволява въвеждането на всички необходими данни за продукта, включително съставките, производителя и друга съществена информация. Това осигурява централизирано и структурирано съхранение на данните за всеки продукт, което в последствие допринася за по-доброто управление и поддръжка на базата данни.

Освен страниците за продуктите, съществува и специална страница, която дава възможност на потребителите да преглеждат информацията, свързана с техния профил и личните им данни. Тази страница предоставя на потребителите обобщена информация за техния акаунт, включително лични данни. Потребителите могат да използват тази страница за управление на своя профил.

Потребителите с администраторски права имат възможността да управляват списъка с потребители на системата, като могат да добавят нови или да изтриват съществуващи потребители. Тази функционалност е достъпна чрез специална страница, която е достъпна само за администратори.

Интерфейсът е проектиран да бъде интуитивен и лесен за използване, като предлага бърз и удобен начин за въвеждане на информация в системата. Това гарантира ефективност и точност на данните, като същевременно осигурява удобство за всички потребители при работа с приложението.

## Диаграми на анализа

## Модел на съдържанието / данните

Моделът на съдържание/данни за управление на козметичния мениджър е създаден с цел да съхранява и управлява всичка съществена информация, свързана с козметичните продукти. Той включва разнообразни обекти, като продукти, компании, потребители и техните взаимоотношения помежду си. В модела се описват атрибутите на всяка една от тези същности, както и техните взаимовръзки, което позволява ефективно управление и използване на информацията от страна на приложението. Този модел е ключов инструмент за правилното функциониране на системата за управление на козметичния мениджър и осигурява цялостно и структурирано съхранение на всички необходими данни.

**Обекти:**

Потребител: Този обект представлява информация за потребителя, като имена, имейл, права и др.

Продукт: Този обект представлява информацията за продукта, като име, цена, описание и др.

Компания: Този обект представлява информацията за компаниите, които са производители на продуктите.

Съставка: Този обект представлява информацията за съставките, които са производители на продуктите

**Атрибути:**

Всяка една същност в системата разполага със свой набор от характеристики, които отразяват информацията, която трябва да бъде запазена и управлявана.

Тези атрибути представляват различни аспекти и детайли за всеки обект, като например наименование, описание, съставки, цена, производител и други съответни данни, които са съществени за съответната същност.

Те играят ролята на ключови елементи за идентификация и управление на информацията в системата за управление на козметичния мениджър.

Например обектът на потребител има атрибути като идентификационен номер, име и др. По същия начин обектът продукта има атрибути като идентификационен номер, име, цена и др.

**Взаимоотношения:**

Моделът съдържание/данни също дефинира връзките между обектите. Например, един продукт може да има само една компания производител.

По същия начин един потребител може да преглежда множество продукти на различни компании. Следователно продуктът може да бъде достъпван от различни потребители .

Системата за управление на козметичния мениджър е предназначена да предоставя цялостно и интелигентно управление на информацията, свързана с козметични продукти, потребители и техните предпочитания. Основната цел на системата е да подпомогне потребителите в избора и употребата на подходящите за тях козметични продукти, като предоставя персонализирани препоръки и съвети за грижа за кожата и косата.

Системата също така има за цел да улесни процесите на администриране и управление на информацията за козметичните продукти от страна на администраторите, като предоставя инструменти за добавяне, редактиране и изтриване на продукти и потребители.

Освен това, системата може да предоставя и аналитични данни и отчети за администраторите, които да им помогнат в управлението и оптимизацията на процесите в системата. Така козметичния мениджър не само предоставя информация и съвети за грижа за кожата и косата, но и допринася за подобряване на потребителския опит и ефективността на бизнес процесите в областта на козметиката.

В системата за управление на козметичния мениджър потребителите и администраторите имат различни нива на достъп и права в зависимост от техните роли.

**Потребител:**

Преглед на продукти: Потребителите могат да разглеждат информация за различните козметични продукти, включително описания, съставки, цени

Управление на профил: Потребителите могат да управляват своя профил

**Администратор:**

Управление на продукти: Администраторите имат право да добавят нови козметични продукти, да редактират вече съществуващи и да изтриват продукти от системата.

Управление на потребители: Администраторите могат да създават нови потребители, да редактират информацията за съществуващи потребители и да изтриват потребителски акаунти при необходимост.

Управление на компании: Администраторите могат да създават нови компании, да редактират информацията за съществуващи компании и да изтриват при необходимост.

Управление на съставки: Администраторите могат да създават нови съставки, да редактират информацията за съществуващи съставки и да изтриват при необходимост.

Тези различни нива на достъп и права осигуряват ефективно управление на системата и удовлетворение на потребителските нужди, като същевременно поддържат сигурността и интегритета на данните.

# Дизайн

Проектът е структуриран по модела MVC (Model-View-Controller), където моделът представлява данните и бизнес логиката, изгледът (view) е отговорен за представянето на данните към потребителя, а контролерът управлява взаимодействието между модела и изгледа.

За изграждане на базата данни е използван подход, който включва дефиниране на структурата и връзките между различните таблици. След създаването на базата данни, кодът е разработен съобразно нея, включително извличането, манипулирането и съхранението на данни в нея.

Програмирането на проекта е извършено с използването на програмния език Java, който осигурява сигурност, надеждност и гъвкавост при разработката на приложения. Сървърната част (back-end) на приложението е разработена с помощта на Spring Boot, което предоставя голямо удобство и бързина при създаването на уеб приложения.

За създаването на потребителския интерфейс (UI) са използвани технологиите HTML [7], CSS [8] и JavaScript [9]. HTML е използван за структурирането на уеб страниците, CSS за стилизирането им и JavaScript за интерактивността и функционалността на уеб приложението. Тези технологии създават приятен и лесен за използване интерфейс за потребителите на приложението.

## Реализация на архитектурата на приложението

Архитектурата на приложението за системата, базирана на модела MVC (Model-View-Controller), е отличен избор за управление на различните компоненти на приложението. Тази архитектура предоставя ясно разделение между данните, бизнес логиката и потребителския интерфейс, което улеснява разработката, поддръжката и разширяването на системата.

Моделът (Model): Това е компонентът, който се грижи за данните и бизнес логиката на приложението. В случая на системата на козметичен мениджър, моделът съдържа данните за продуктите, потребителите, компаниите. Моделът предоставя методи за достъп и манипулиране на тези данни.

Контролерите (Controller): Контролерите приемат заявки от потребителите и се грижат за обработката на тези заявки. Те извличат необходимата информация от модела и я предоставят на потребителя чрез изгледа. Контролерите също така се грижат за валидацията на данните, преди те да бъдат записани в модела.

Изгледът (View): Изгледът е отговорен за визуализирането на информацията, предоставена от контролерите, и за представянето ѝ на потребителя. Той предоставя потребителски интерфейс, с който потребителят може да взаимодейства. Изгледът също така предоставя формуляри за въвеждане на данни и други интерактивни елементи.

Тази архитектурна модел позволява гъвкавост, лесно разширяване и поддръжка на приложението, като в същото време осигурява ясно разделение на отговорностите между различните компоненти.

## Описание на слоевете, предназначението им, библиотеки и методи включени в съответния слой.

Системата се състои от следните слоеве:

Слой на контролерите (Controllers layer) играе ключова роля в управлението на заявките, постъпващи от потребителите към системата, и осигурява връщането на подходящи резултати към тях. Този слой често използва различни библиотеки и фреймуърки, като например Spring Web, за да улесни обработката на HTTP заявките и да осигури по-голяма гъвкавост и удобство при програмирането.

В контролерите се съдържат методи, които се използват за конкретната обработка на заявките от потребителите. Някои от тях в този слой включват:

loginUser(): Метод за обработка на заявка за влизане в системата (login) от страна на потребителя.

registerUser(): Метод за обработка на заявка за регистрация на нов потребител в системата.

homeView(): Метод за предоставяне на началната страница на приложението, която обикновено съдържа обобщение на основната информация или функционалност на системата.

seasonView(): Метод за предоставяне на информация или функционалност, свързана с текущия сезон или период.

Освен тези методи, в контролерите са добавени и други, специфични за приложението функции, които обслужват конкретни заявки от потребителите.

Всеки контролер в този слой обикновено съдържа логика за обработка на заявките, валидация на данните и подготовка на отговорите към клиентската страна. Това улеснява разделението на отговорностите и подобрява поддръжката на приложението.

Този слой е от съществено значение за всяко уеб приложение, тъй като осигурява връзката между потребителите и бизнес логиката на приложението, като в същото време гарантира правилното функциониране на цялата система.

Слой на услугите (Services layer) играе критична роля в управлението на бизнес логиката на приложението и е отговорен за свързването между контролерите и репозиториите. Той е средната стъпка между слоя на контролерите и слоя на данните (репозиториите или хранилищата). В този слой обикновено се съдържат различни сервизи или услуги, които предоставят високо ниво на абстракция над операциите с данните и бизнес логиката.

Някои от ключовите компоненти и практики, които могат да бъдат използвани в слоя на услугите, включват:

UserService: Сервиз, който предоставя функционалности, свързани с потребителските операции, като например създаване, редактиране, изтриване и т.н.

Слой на услугите осигурява високо ниво на абстракция над данните и репозиториите, като в същото време гарантира, че бизнес логиката на приложението е добре структурирана и поддържаема. Този слой е ключов за изолирането на бизнес логиката от детайлите на имплементацията на слоя на данните, което улеснява разработката, тестването и поддръжката на приложението.

Разширяването на бизнес логиката и въвеждането на промени става по-лесно благодарение на абстракцията, предоставена от слоя на услугите. Този слой предоставя възможност за високо ниво на гъвкавост на приложението.

Слой на репозиториите (Repositories layer) играе важна роля в управлението на връзката с базата данни и осигурява достъп до нужната информация от нея. Този слой се грижи за комуникацията със системата за управление на бази данни (SQL Server), изпълняването на заявки и манипулирането на данните.

В слоя на репозиториите често се използва библиотеката Spring Data JPA, която предоставя удобни абстракции над операциите с базата данни чрез Java Persistence API (JPA). Това позволява лесно създаване на репозитории, които използват стандартни JPA анотации за мапиране на обекти от Java към таблици в базата данни.

Примери за интерфейси на репозиторита, които могат да бъдат използвани в този слой, включват:

UserRepository: Интерфейс, който дефинира методи за достъп и манипулиране на данни за потребителите в системата.

ProductRepository: Интерфейс, който дефинира методи за работа с продуктите в системата.

Като има и други репозиторита, но това са двете основни.

Осигурява изолация на бизнес логиката от детайлите на имплементацията на базата данни и улеснява тестването и поддръжката на приложението. Благодарение на Spring Data JPA и абстракцията, която тя предоставя, лесно се извършват операции с базата данни без да се използва директни SQL заявки.

Този слой е от съществено значение за постигането на добра структура и управление на данните в приложението и играе ключова роля за неговата функционалност и ефективност.

Слой на моделите (Models layer) играе ключова роля в приложението, тъй като в него се дефинират структурата и характеристиките на данните, които се използват в системата. Този слой съдържа моделите на приложението, които отразяват обектите или същностите в приложението и представят таблиците в базата данни. Моделите са отговорни за съхраняването и обработката на данните, като осигуряват структура и методи за манипулиране на информацията.

В този слой обикновено не се използват конкретни библиотеки, но се използват анотации като @Entity, @Table, @Column, @Id, които се използват от ORM (Object-Relational Mapping) библиотеки като Hibernate или Spring Data JPA, за да се мапират Java класове към таблиците в базата данни. Тези анотации помагат за дефинирането на мапинга между полетата на класовете и колоните на таблиците, както и за управлението на основни аспекти на съхранението на данните.

Примери за класове модели, които могат да се срещнат в този слой, включват:

Потребител: Модел, който представя потребителите в системата и съдържа информация за тях като потребителско име, парола, имейл и други данни.

Продукт: Модел, който представя продуктите или артикулите в системата и съдържа информация за тях като име, описание, цена и други характеристики.

Компания: Модел, който представя компаниите или фирмите, които управляват продуктите или предоставят услугите в системата.

Съставка: Модел, който представя съставките на продуктите или рецептите в системата и съдържа информация за тях като наименование, количество и други характеристики.

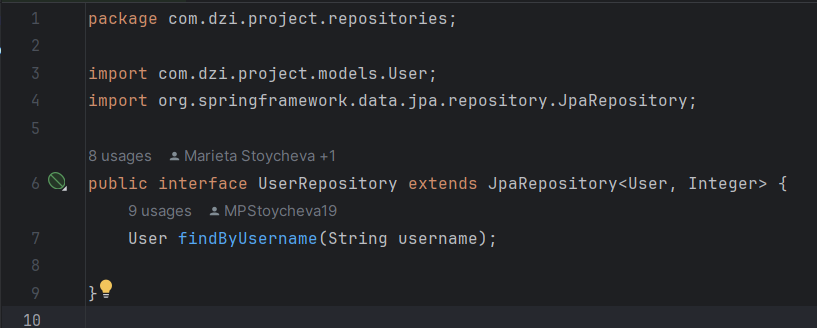
Сезон: Модел, който представя сезоните в системата и съдържа информация за тях като наименование, дата на начало и край и други характеристики.

Тип: Модел, който представя типовете продукти или услуги в системата и съдържа информация за тях като категория, описание и други характеристики.

Този слой е от съществено значение за правилното функциониране на приложението, тъй като дефинира структурата на данните и осигурява начин за манипулирането на информацията. Освен това, той улеснява разработката и поддръжката на приложението, като осигурява ясен и структуриран начин за работа с данните.

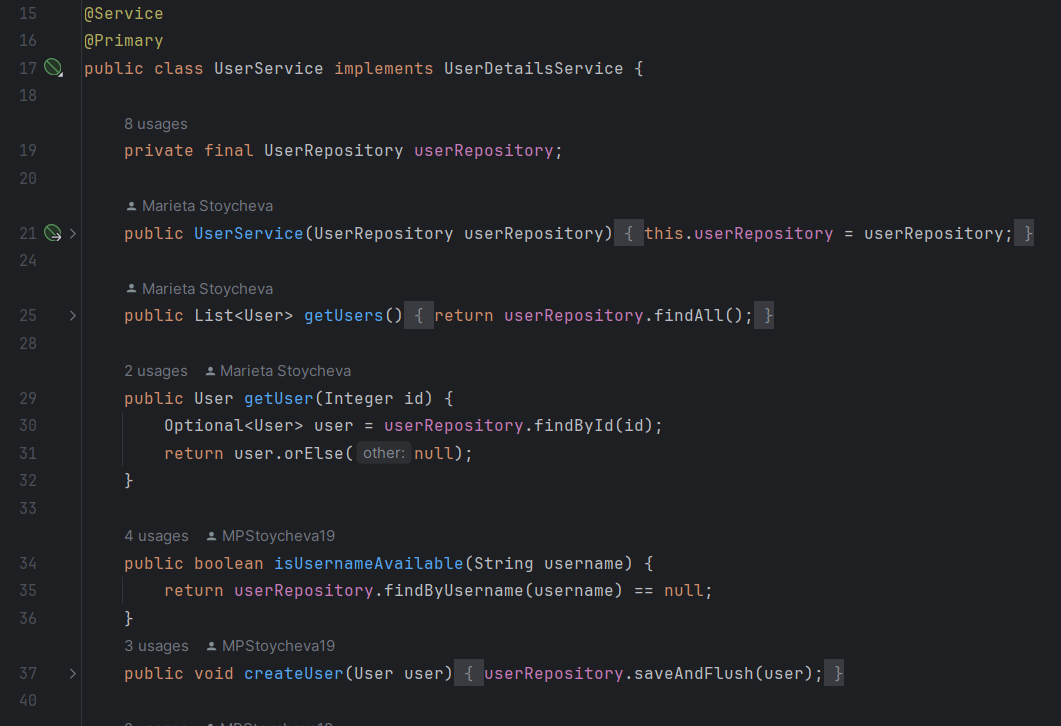
Тези слоеве работят заедно, за да осигурят пълната функционалност на системата.

## Организация и код на заявките към база от данни



A screen shot of a computer program

Description automatically generated



Използването на Spring Data JPA в организацията на заявките към базата данни в системата е отличен подход за управление на данните. Spring Data JPA предоставя удобен начин за дефиниране на репозитории за моделите на данните и изпълнение на различни операции като търсене, създаване, обновяване и изтриване.

В предоставения код се използва интерфейсът JpaRepository<User, Integer> и предоставя допълнителни методи за работа с базата данни. Този интерфейс предоставя базови методи като findAll(), findById(), save(), delete() и други, които са генерирани автоматично от Spring Data JPA въз основа на името на метода. Това осигурява удобство и бързина при разработката, като намалява необходимостта от ръчно писане на SQL заявки.

Следователно, в този слой от системата на козметичния мениджър не се използват конкретни библиотеки за работа с базата данни, тъй като това е реализирано от Spring Data JPA. Този фреймуърк автоматично генерира SQL заявки и извършва необходимите операции с базата данни, което значително опростява и ускорява разработката на приложението. Такъв подход осигурява висока абстракция и гъвкавост на кода, като същевременно гарантира ефективността на извършваните операции с данните.

## Наличие на потребителски интерфейс (конзолен, графичен, уеб)

Проектът има уеб потребителски интерфейс. Това означава, че потребителите могат да получат достъп до функционалността на приложението чрез браузър, като използват графичен интерфейс. На началната страница като потребител може да се видят сезоните в годината. За всеки един от тях има определени продукти и когато се натисне някой от сезоните излизат тези продукти. Може да се види и търсачката, която работи по име, описание и цена на продукта.

Страница за регистрация на нов потребител, където администраторите могат да въведат информацията за потребителя, включително имена, права, имейл, парола, потребителско име и други подробности. Освен администратора и потребител сам може да се регистрира, но не може да си даде по-големи права от това да са нормални потребители.

Страница за регистрация на нова компания, където администраторите могат да въведат информацията за компанията, включително информация за името й.

Страница за регистрация на нов продукт, където администраторите могат да въведат информацията за продукта, включително информация за името, цената, описанието и др.

Страница за регистрация на нова съставка, където администраторите могат да въведат информацията за съставката, включително информация за името й.

Страница с детайлна информация за потребител, където можете да видите всички подробности за този потребител, включително имена, имейл, контакти и др.

Страници с детайлна информация за продукти, където можете да видите всички подробности за продуктите, като във всяка една от тях са разделени по различен критерий, по които потребителя може да избира.

Администраторите имат права да добавят нови потребители и да изтриват съществуващи потребители от системата.

Така, за регистрацията на нови потребители и продукти е нужно администраторите да влязат в системата и да използват съответните страници за това. Също така, администраторите имат достъп до страницата за регистрация на нови компании и могат да добавят нови съставки към системата. Всички потребители могат да използват страниците за детайлна информация за да видят всички подробности за съответния продукт, от който се нуждаят.

# Ефективност и бързодействие на решението

За да се осигури висока ефективност и бързодействие на системата, се прилагат различни методи и стратегии. Тези методи включват оптимизации на различни нива на архитектурата и настройките на системата.

Оптимизирани заявки към базата данни: Използването на индекси и оптимизация на конфигурациите на базите данни може да подобри бързодействието на системата, като се избягва изпълнението на неефективни заявки.

Паралелна обработка на заявки: Разделянето на заявките на парчета, които могат да се обработват паралелно, може да увеличи общата производителност на системата, като намали времето за изпълнение на заявките.

Оптимизация на кода: Анализирането и оптимизацията на кода на приложението може да намали времето за зареждане на приложението и да подобри отговорността му на заявките на потребителите.

Следене на производителността и тестване: Редовното следене на производителността на системата и извършването на тестове и анализи помагат за идентифицирането на проблеми и предприемането на подходящи мерки за подобряване на бързодействието.

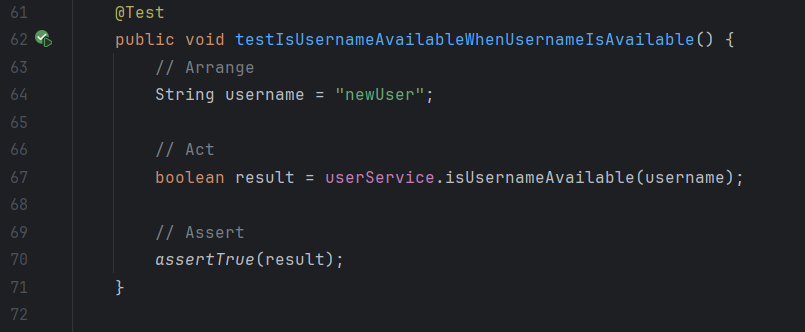
# LogIn тестТестване

Тези са два примера на тестови случаи за потвърждение на функционалността на влизането в проекта.

Тест за успешно влизане с валидни данни: Създава потребителски обекти с валидни потребителско име и парола. Предварително настройва мокнато репозитори да връща създадения потребителски обект при извикване на метода findByUsername(). Създава инстанция на LogInService с мокнатото репозитори. Проверява, че методът logIn на LogInService връща true, което означава успешно влизане с валидни данни.

Тест за неуспешно влизане с невалидни данни: Установяват се невалидни данни за влизане (например, потребител със същото потребителско име не съществува). Настройва мокнато репозитори да върне null при извикване на метода findByUsername(). Създава се инстанция на LogInService с мокнатото репозитори. Проверява, че методът logIn() на LogInService връща false, което означава неуспешно влизане с невалидни данни.

Тези тестови случаи покриват основните сценарии за влизане в системата с валидни и невалидни данни и се грижат да се провери коректното функциониране на логин услугата.



Тестът testIsUsernameAvailableWhenUsernameIsAvailable проверява функционалността на метода isUsernameAvailable() в сценарий, когато потребителското име е свободно.

Подготовка (Arrange): Задава се потребителско име newUser, което се предполага, че не е налично в системата.

Действие (Act): Извиква се методът isUsernameAvailable() с предварително подготвеното потребителско име newUser.

Уверяване (Assert): Проверява се дали резултатът от метода е true, което означава, че потребителското име newUser е наистина свободно.

Така този тест уверява, че когато се извика методът isUsernameAvailable() с потребителско име, което се предполага, че е свободно в системата, резултатът е true, което е очакваното поведение при наличие на такова потребителско име.



В тези два теста се проверява добавянето и редактирането на потребители.

testCreateUser(): Създава се мок обект на класа User с име "existingUser" и парола "password". С поведението на метода userRepository.saveAndFlush() се прави мок, използвайки метода when() на Mockito, като се уточнява, че трябва да върне existingUser при извикване с който и да е аргумент. Създава се инстанция на UserService с мокнатия UserRepository. Извиква се методът createUser() на userService с existingUser. Проверява се, че методът userRepository.saveAndFlush() е извикан точно веднъж със заснетия аргумент. Уверява се, че името на заснетия аргумент съвпада с името на existingUser.

testЕдитUser(): Създава се мок обект на класа User с име "existingUser" и парола "password". С поведението на метода userRepository.save () се прави мок, използвайки метода when() на Mockito, като се уточнява, че трябва да върне existingUser при извикване с който и да е аргумент. Създава се инстанция на UserService с мокнатия UserRepository. Извиква се методът editUser() на userService с existingUser. Проверява се, че методът userRepository.save() е извикан точно веднъж със заснетия аргумент. Уверява се, че името на заснетия аргумент съвпада с името на existingUser.

В заключение, тези тестове проверяват дали UserService правилно взаимодейства с UserRepository като записва потребителски обекти. Първият тест проверява поведението при създаване на потребител, докато вторият тест проверява поведението при редактиране на потребител.

# Заключение и възможно бъдещо развитие

Проектът за козметичен мениджър представлява важен инструмент за управление на козметичен бизнес. Чрез внимателно проектираната архитектура и използването на съвременни технологии, успяхме да разработим приложение, което предоставя удобство и функционалност за козметичните салони и техните клиенти.

Проектът се отличава с интуитивен потребителски интерфейс, който позволява лесно управление на клиентите, записване на срещи, управление на услугите и продуктите, както и генериране на отчети за финансовите данни. Благодарение на използването на модела MVC и различните слоеве на архитектурата, постигнахме чист и организиран код, който е лесен за разбиране, разширяване и поддръжка.

Възможно бъдещо развитие:

Добавяне на функционалности за онлайн записване на срещи и плащания.

Разширяване на аналитичните възможности за предоставяне на персонализирани препоръки за клиентите.

Разработка на мобилно приложение за по-голямо удобство и достъпност на потребителите.

Подобряване на сигурността на данните и въвеждане на мерки за защита на личната информация на клиентите.

Със стратегическо развитие и постоянно подобрение, проектът за козметичен мениджър има потенциал да стане важен инструмент за козметичната индустрия, като предостави на бизнесите в тази област иновативни решения за управление и нарастване на клиентелата.

# Използвани литературни източници и Уеб сайтове

[1] Ph. Webb, D. Syer, J. Long, S. Nicoll et. al. (2024), Spring Boot Reference Documentation, <https://docs.spring.io/spring-boot/docs/current/reference/html/>.

[2] R. Johnson, J. Hoeller, K. Donald, C. Sampaleanu et al. (2024) Spring Framework Documentation, <https://docs.spring.io/spring-framework/reference/>.

[3] O. Gierke, Th. Darimont, Ch. Strobl, M. Paluch, J. Bryant, G. Turnquist. (2024) Spring Data JPA, <https://docs.spring.io/spring-data/jpa/reference/>.

[4] Spring Security, (2024) <https://docs.spring.io/spring-security/reference/>.

[5] Thymeleaf(2024), <https://www.thymeleaf.org/documentation.html>.

[6] SQL Server technical documentation (2024), <https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16>.

[7] HTML: HyperText Markup Language (2024), <https://developer.mozilla.org/en-US/docs/Web/HTML>.

[8] CSS: Cascading Style Sheets (2024), <https://developer.mozilla.org/en-US/docs/Web/CSS>.

[9] JavaScript (2024), <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.

[10] IntelliJ IDEA overview (2024), <https://www.jetbrains.com/help/idea/discover-intellij-idea.html#next-steps>.

[11] Git--distributed-is-the-new-centralized, <https://git-scm.com/doc>.

[12] Apache Maven Project, <https://maven.apache.org/guides/>.

[13] Microsoft 365 documentation, <https://learn.microsoft.com/en-us/microsoft-365/?view=o365-worldwide>

[14] Word help & learning, <https://support.microsoft.com/en-us/word>

[15] Rezaei, S. (2019). Learning Spring Application Development: Develop dynamic, feature-rich, and robust Spring-based applications using the Spring Framework. Packt Publishing.

# Приложения

Използваните приложения са:

Microsoft Word – за структуриране и написване на документацията на проекта.

Microsoft PowerPoint – за изграждане на презентацията на проекта.

IntelliJ IDEA 2023.3 – за изписване на кода и функционалността на проекта.

LucidChart – за създаването на диаграмите.

Github – за прилежност и подреденост на целия проект, следене на срокове и създавани проблеми.