**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

**Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования   
«Санкт-Петербургский государственный университет телекоммуникаций   
им. проф. М. А. Бонч-Бруевича»**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Факультет ИКСС

Кафедра ПИиВТ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

по дисциплине:

Разработка Java-приложений управления телекоммуникациями

Тема: «Разработка приложения для салона красоты»

Вариант: 97

Группа: ИКПИ-84

Студент: Василец Павел Васильевич

Руководитель: Помогалова Альбина Владимировна

Оценка: \_\_\_\_\_\_\_\_

Санкт-Петербург

2020

**§1. Реферат**

## **1.1 Цель работы**

Цель настоящей работы – проектирование, разработка и реализация CRM(Customer Relationship Management)-приложения для салона красоты «Beauty Saloon», ставящего себе задачу автоматизации взаимодействия с заказчиками, повышения уровня продаж и улучшения обслуживания клиентов путем сохранения информации о клиентах и дальнейшей ее использовании.

## **1.2 Методология проведения работы**

Основа данного курсового проекта базируется на основных положениях анализа предмета с детальным учетом его индивидуальных признаков, синтеза и моделирования.

## **1.3 Результат работы**

По итогам детального анализа предметной области и проделанной работы удалось добиться реализации, полностью функционирующей и отвечающей всем требованиями, современной CRM-системы.

Для хранения и обработки внутренних данных приложения была разработана база данных на основе MySQL – свободной реляционной системе управления базами данных.

С целью уменьшения времени у трудозатрат для реализации поставленной задачи был выбран Vaadin Java Web Framework - свободно распространяемый фреймворк для создания высоконагруженных веб-приложений. С помощью данного фреймворка удалось реализовать весь поставленный в задачах функционал, в частности:

* создание, удаление и редактирование клиентов компании;
* создание, удаление и редактирование работников компании;
* удобные формы регистрации и авторизации пользователей;
* защита от межсайтовых запросов и перехвата данных сессии;
* добавление и изменение перечня услуг, предоставляемых компанией;
* просмотр своего расписания работником;
* главная страница сайта с возможностью записи на процедуры;

Для управления взаимодействием базы данных, приложения и сервера был использован Spring Framework. С помощью него было реализовано:

* защита доступа к внутренним объектам приложения и страницам;
* авторизация и регистрация новых пользователей в системе;
* внедрение зависимостей, таких как сущности, репозитории, сервисы;

Сам проект собирался при помощи Apache Maven – фреймворка для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM (Project Object Model). С помощью данного фреймворка удалось добиться удобной и читаемой структуры проекта, быстрого поиска и интегрирования всех необходимых для реализации приложения зависимостей.

## **1.4 Область применения результатов**

Использование в качестве основ новых проектов для более детального изучения и освоения технологий, используемых в данной области.

## **1.5 Выводы**

Для достижения поставленной цели были рассмотрены технологии клиент-серверного программирования, теория разработки и создания баз данных, в частности с помощью реализующей спецификации JPA (Java Persistence API) библиотеки для языка программирования Java - Hibernate, решающую задачу объектно-ориентированного отображения сущностей (Entity) базы данных.

Содержание

[§1. Реферат 1](#_Toc60033977)

[**1.1 Цель работы** 1](#_Toc60033978)

[**1.2 Методология проведения работы** 1](#_Toc60033979)

[**1.3 Результат работы** 1](#_Toc60033980)

[**1.4 Область применения результатов** 2](#_Toc60033981)

[**1.5 Выводы** 2](#_Toc60033982)

[§2. Термины и определения 5](#_Toc60033983)

[§3. Введение 6](#_Toc60033984)

[§4. Основная часть 7](#_Toc60033985)

[**4.1 Направление исследования** 7](#_Toc60033986)

[**4.2 Обоснование выбранных направлений** 7](#_Toc60033987)

[**4.3 Описание процесса исследования** 7](#_Toc60033988)

[**4.3.1 Теоретическое исследование** 7](#_Toc60033989)

[**4.3.2 Экспериментальное исследование** 10](#_Toc60033990)

[**4.3.3 Оценка результатов исследования** 26](#_Toc60033991)

[§5. Заключение 27](#_Toc60033992)

[§6. Список использованных источников 28](#_Toc60033993)

[§7. Приложения 29](#_Toc60033994)

[**7.1 Приложение A** 29](#_Toc60033995)

[**7.1.1 Листинг AbstractEntity.java** 30](#_Toc60033996)

[**7.1.2 Листинг Customer.java** 30](#_Toc60033997)

[**7.1.3 Листинг Employee.java** 33](#_Toc60033998)

[**7.1.4 Листинг Role.java** 37](#_Toc60033999)

[**7.1.5 Листинг Service.java** 37](#_Toc60034000)

[**7.1.6 Листинг User.java** 39](#_Toc60034001)

[**7.1.7 Листинг CustomerRepository.java** 41](#_Toc60034002)

[**7.1.8 Листинг CustomerRepository.java** 41](#_Toc60034003)

[**7.1.9 Листинг ServiceRepository.java** 42](#_Toc60034004)

[**7.1.10 Листинг UserRepository.java** 42](#_Toc60034005)

[**7.1.11 Листинг CustomerService.java** 43](#_Toc60034006)

[**7.1.12 Листинг EmployeeService.java** 44](#_Toc60034007)

[**7.1.13 Листинг ServiceService.java** 45](#_Toc60034008)

[**7.1.14 Листинг UserService.java** 46](#_Toc60034009)

[**7.1.15 Листинг AuthException.java** 49](#_Toc60034010)

[**7.1.16 Листинг AuthorizedRoute.java** 49](#_Toc60034011)

[**7.1.17 Листинг AuthService.java** 50](#_Toc60034012)

[**7.1.18 Листинг CustomRequestCache.java** 52](#_Toc60034013)

[**7.1.19 Листинг SecurityConfiguration.java** 52](#_Toc60034014)

[**7.1.20 Листинг SecurityUtils.java** 54](#_Toc60034015)

[**7.1.21 Листинг AccountsView.java** 55](#_Toc60034016)

[**7.1.22 Листинг ChangeView.java** 56](#_Toc60034017)

[**7.1.23 Листинг ChangeUserForm.java** 60](#_Toc60034018)

[**7.1.24 Листинг CabinetApp.java** 64](#_Toc60034019)

[**7.1.25 Листинг WelcomeLayout.java** 66](#_Toc60034020)

[**7.1.26 Листинг DashBoardView.java** 67](#_Toc60034021)

[**7.1.27 Листинг CustomerForm.java** 69](#_Toc60034022)

[**7.1.28 Листинг CustomerListView.java** 74](#_Toc60034023)

[**7.1.29 Листинг EmployeeForm.java** 77](#_Toc60034024)

[**7.1.30 Листинг EmployeeListView.java** 82](#_Toc60034025)

[**7.1.31 Листинг LoginView.java** 85](#_Toc60034026)

[**7.1.32 Листинг Logout.java** 87](#_Toc60034027)

[**7.1.33 Листинг ServiceForm.java** 87](#_Toc60034028)

[**7.1.34 Листинг ServiceView.java** 90](#_Toc60034029)

[**7.1.35 Листинг LandingView.java** 92](#_Toc60034030)

[**7.1.36 Листинг RegisterDialog.java** 94](#_Toc60034031)

[**7.1.37 Листинг Constants.java** 98](#_Toc60034032)

[**7.1.38 Листинг Application.java** 98](#_Toc60034033)

[**7.1.39 Листинг application.properties** 99](#_Toc60034034)

[**7.1.40 Листинг shared-styles.css** 99](#_Toc60034035)

[**7.1.41 Листинг service-view.css** 100](#_Toc60034036)

[**7.1.42 Листинг employee-view.css** 101](#_Toc60034037)

[**7.1.43 Листинг customer-view.css** 101](#_Toc60034038)

[**7.1.44 Листинг change-view.css** 101](#_Toc60034039)

[**7.1.45 Листинг cabinet-view.css** 102](#_Toc60034040)

[**7.1.45 Листинг pom.xml** 103](#_Toc60034041)

# **§2. Термины и определения**

В настоящей пояснительной записке о курсовом проекте применяют следующие термины с соответствующими определения:

Фреймворк – программная платформа, определяющая структуру программной системы

CRM – система управления взаимоотношениями с клиентами

IT – information technology (информационные технологии)

База данных – представленная в объективной форме совокупность самостоятельных материалов, систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью ЭВМ

ОС – операционная система

Сущность – является ключевым элементов объектно-реляционного отображения

ЭВМ – электронно-вычислительная машина

Front-end - клиентская сторона пользовательского интерфейса к программно-аппаратной части

Back-end – программно-аппаратная часть сервиса.

# **§3. Введение**

В настоящее время, особенно в период пандемии, множество компаний уходят на удаленный режим работы. Массово закрывается офлайн-бизнес и открывается онлайн-бизнес. К онлайн-бизнесу можно отнести различные сервисы изготовления и доставки предметов быта и пищи, разного рода цифровых устройств и т.п. В связи с чем отпадает необходимость в строительстве или аренде помещений по приему и обработке приходящих клиентов. Как следствие сильно возросла востребованность онлайн-приложений управления взаимоотношениями с клиентами.

Необходимость исследования данной темы обуславливается быстрорастущей конкуренцией на рынке IT-компаний, занимающимися разработкой таких приложений, а также увеличивающимися требованиями компаний-заказчиков к таким приложения.

При проектировании и разработке данных приложений следует учитывать вышестоящие факторы, которые также обуславливают несколько других:

* высокая нагрузка на приложение;
* спрос на разработку и выпуск подобных приложений;

# **§4. Основная часть**

## **4.1 Направление исследования**

1. Прикладное исследование:
   * направленно на применение новых знаний для достижения практических целей и решения конкретных задач;
2. Разработка:
   * направлено на внедрение в практику результатов конкретных прикладных исследований;

## **4.2 Обоснование выбранных направлений**

Данные направления были выбраны с финансовой и опытной точки зрения, так как иные направления исследования малоприменимы к целям исследования.

## **4.3 Описание процесса исследования**

### **4.3.1 Теоретическое исследование**

Для реализации поставленной цели было необходимо внимательно изучить принципы построения приложения управления взаимоотношениями с клиентами и все технологии, которые необходимо использовать при реализации такого приложения.

Было выяснено, что CRM – это модель взаимодействия, основанная на теории, что центром философии бизнеса, в которой эта модель используется, является клиент. А главным направлением деятельности компании являются меры по обеспечению эффективного маркетинга, продаж и обслуживания клиентов. Данная модель взаимодействия должна осуществлять поддержку бизнес-целей, основными из которых являются:

* сбор, хранение и анализ информации о потребителях и партнерах;
* сбор и хранение информации о внутренних процессах компании;
* продажи;
* поддержка потребителей;

Сама CRM-система может состоять из:

* фронтальной части, обеспечивающей обслуживание клиентов онлайн через web-сервис;
* операционной части, обеспечивающей авторизацию и оперативную отчетность;
* хранилища данных (база данных);
* аналитической подсистемы;
* распределенной системы поддержки продаж;

CRM-системы имеют следующую классификацию:

* автоматизированная система управления продажами;
* управление маркетингом;
* управление клиентским обслуживанием;

Для реализации поставленной задачи больше всего подошла смешанная по классификации CRM-система, состоящая из фронтальной и операционной частей, а также хранилища данных в виде базы данных. По классификации предполагаемую CRM-систему можно отнести к автоматизированной системе управления продажами и клиентским обслуживанием.

В качестве основного инструмента работы с базой данных была выбрана свободная реляционная система управления базами данных MySQL. Выбрана она была по следующим причинам:

* MySQL может работать на любой ОС;
* полный функционал доступен бесплатно;
* легковесная, как следствие не требует мощного железа;

Инструментом разработки back-end и front-end логики CRM-системы был выбран Vaadin Java Web Framework (далее Vaadin), так как данный фреймворк разрабатывался в основном именно для разработки с помощью него CRM-систем. Vaadin содержит в себе весь необходимый для этого инструментарий.

Связующим звеньями всех технологий проекта были выбраны Spring Framework (далее Spring) и Apache Maven (далее Maven).

Spring Framework в свою очередь должен реализовать доступ к данным, хранящимся в базе данных, через JDBC (Java Database Connectivity) средства, а также их управлением: созданием, сохранением, изменением, удалением;

### **4.3.2 Экспериментальное исследование**

CRM-модель предполагает в системе наличие следующих сущностей:

* сотрудник компании (employee);
* клиент компании (customer);
* пользователь (user);

Так как пользователя необходимо разграничить по доступному ему функционалу, было решено ввести систему ролей, состоящую всего их двух ролей: USER – пользователь, ADMIN – администратор;

Пользователю с ролью ADMIN будет доступен полный функционал CRM-системы, в то время как пользователю с ролью USER лишь часть.

CRM-система с определенной ранее классификацией помимо сотрудников, клиентов и пользователей должна содержать в себе внутренние данные, необходимые не только для управления взаимодействием, но и для продаж. Исходя из того, что исследуемая модель CRM проектируется для салона красоты, можно сделать вывод, что продуктом этой компании будут услуги. В связи с чем было решено ввести еще одну сущность – «услуга» (service), которая будет хранить в себе весь спектр услуг, оказываемых салоном красоты.

После завершения проектирования сущностей базы данных было необходимо установить корректные связи между этими сущности. Итоговые связи выглядят следующим образом:

* employee-user: One-To-One;
* employee-service: Many-To-Many;
* customer-employee: Many-To-One;
* customer-service: Many-To-Many;

Владеющей сущностью является первая в пункте сущность, вторая соответственно является владеемой.

Более детально эту схему «сущность-связь» можно рассмотреть на EER-диаграмме базы данных:

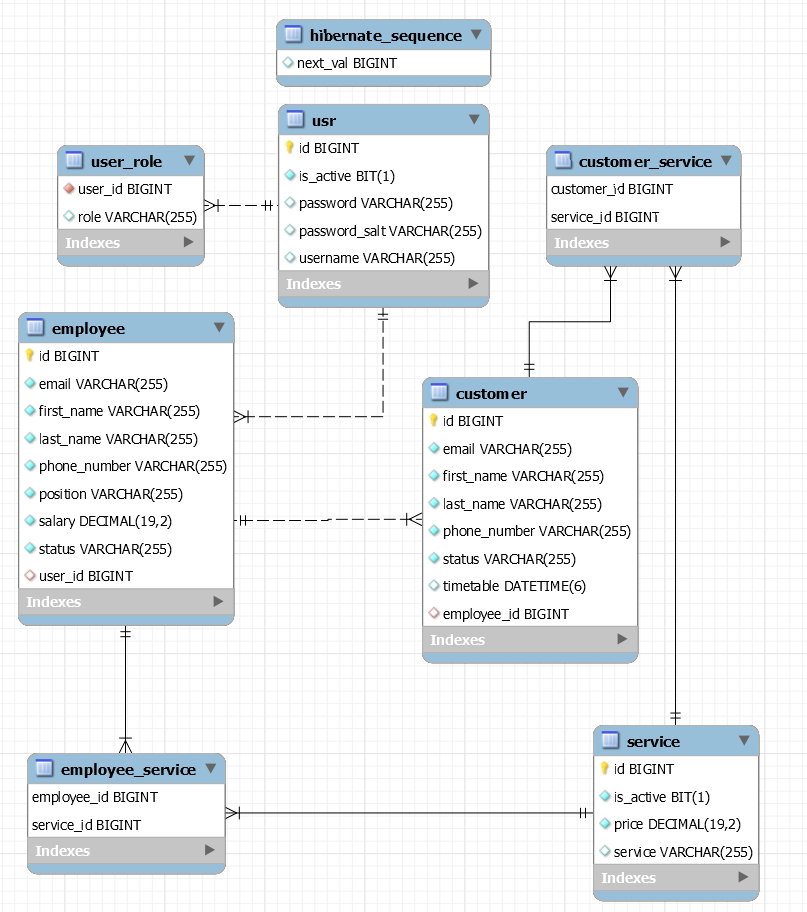


Рис.1 – EER-диаграмма базы данных проекта

На данном этапе было решено приступить к проектированию и реализации пользовательского интерфейса и внутренней логики CRM-системы взаимодействия пользователя с ней.

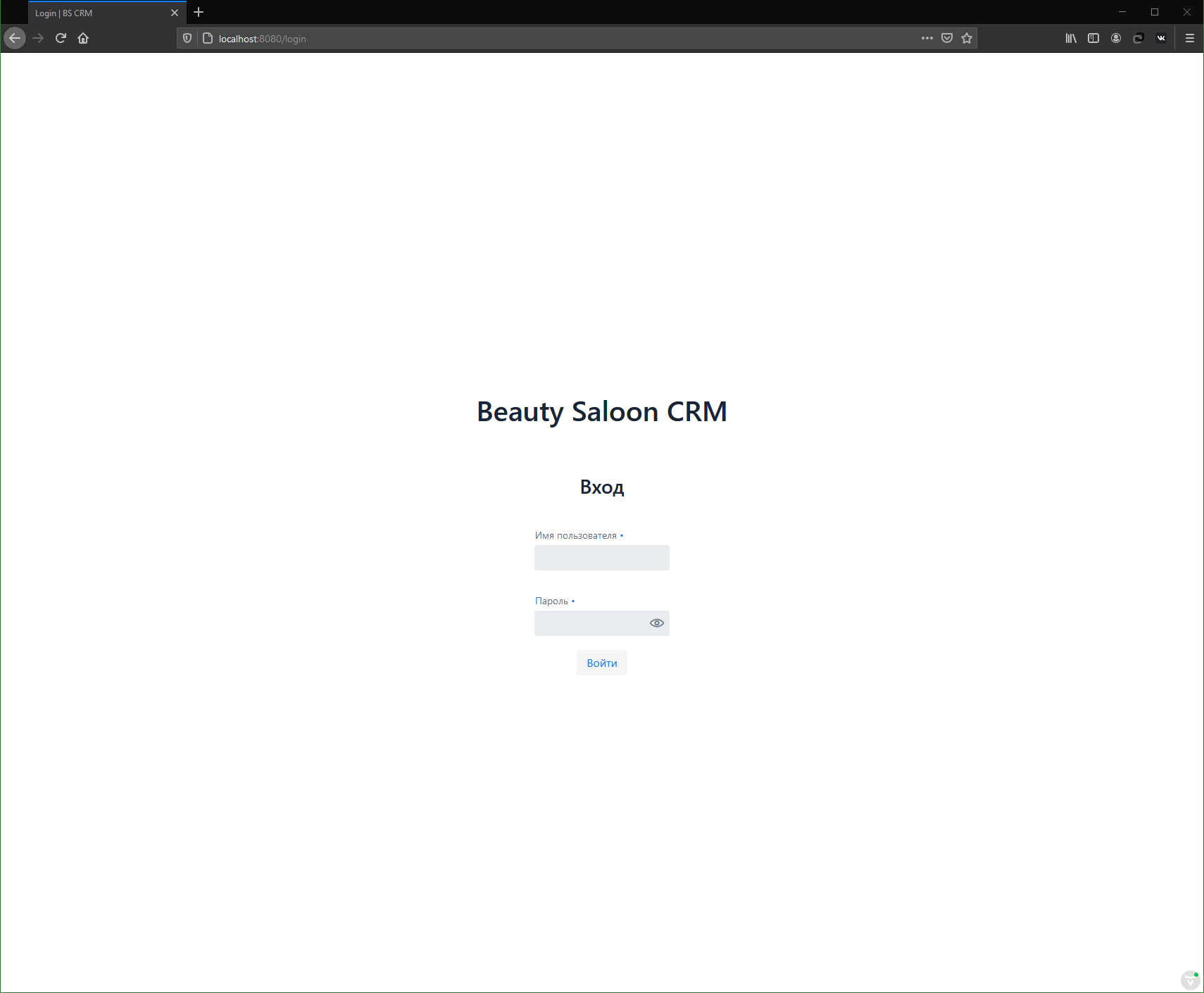
С помощью Vaadin была реализована форма авторизации в CRM-приложении: 

Рис.2 – Форма авторизации в CRM-системе

На ней пользователю требуется ввести корректный логин и пароль. Проверка логина пользователя не чувствительна к регистру. Регистрация пользователей с одинаковым логином заблокирована. Если пользователь ошибется в вводе логина или пароля, система ему об этом сообщит во всплывающем уведомлении:

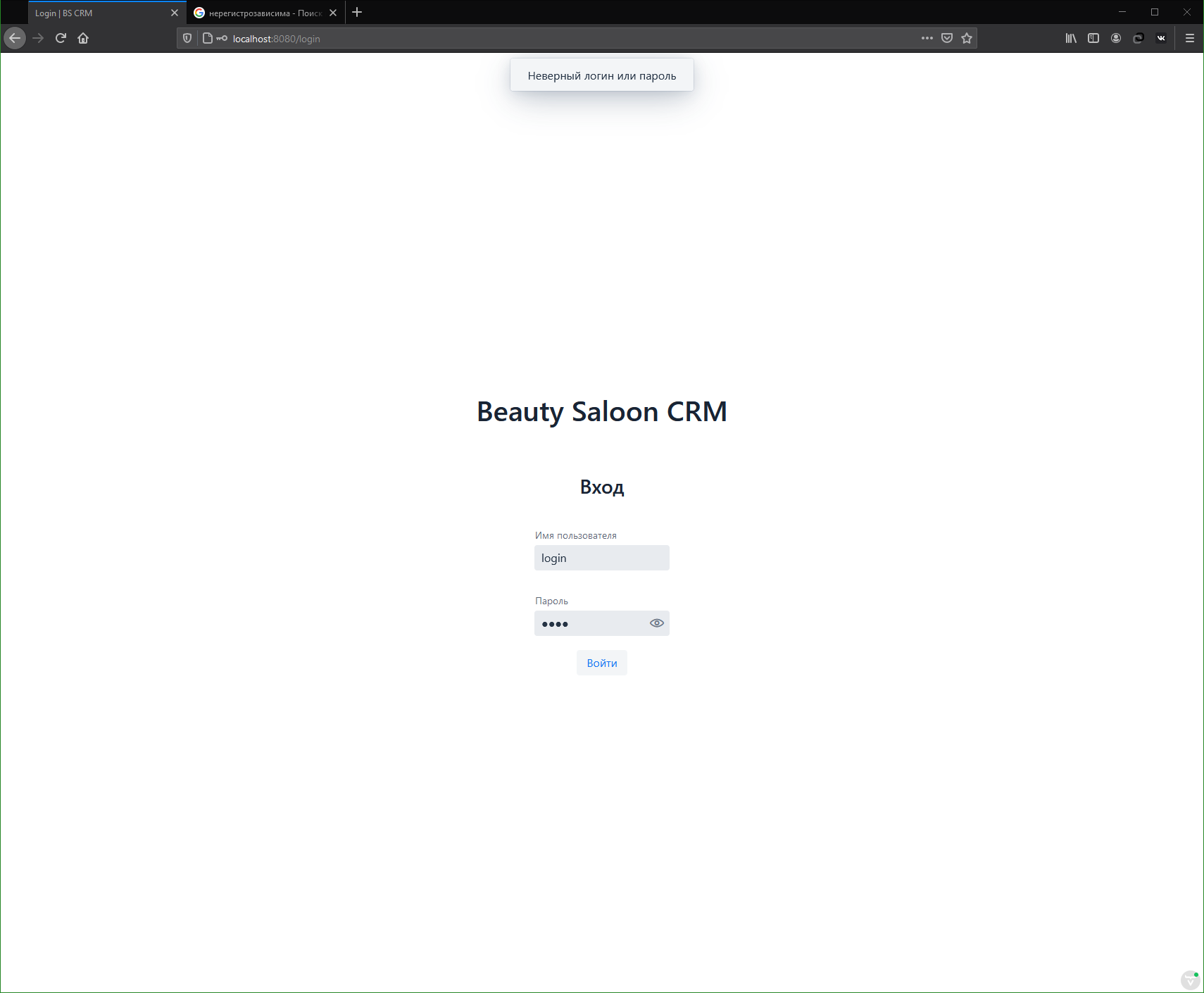


Рис.3 – Обработка неверного ввода данных учетной записи

После корректного ввода логина и пароля система предоставит пользователю доступ в CRM-систему с конечным функционалом в соответствии с его ролью. Страницей после авторизации является главная страница личного кабинета с приветствием сотрудника и блоком новостей:

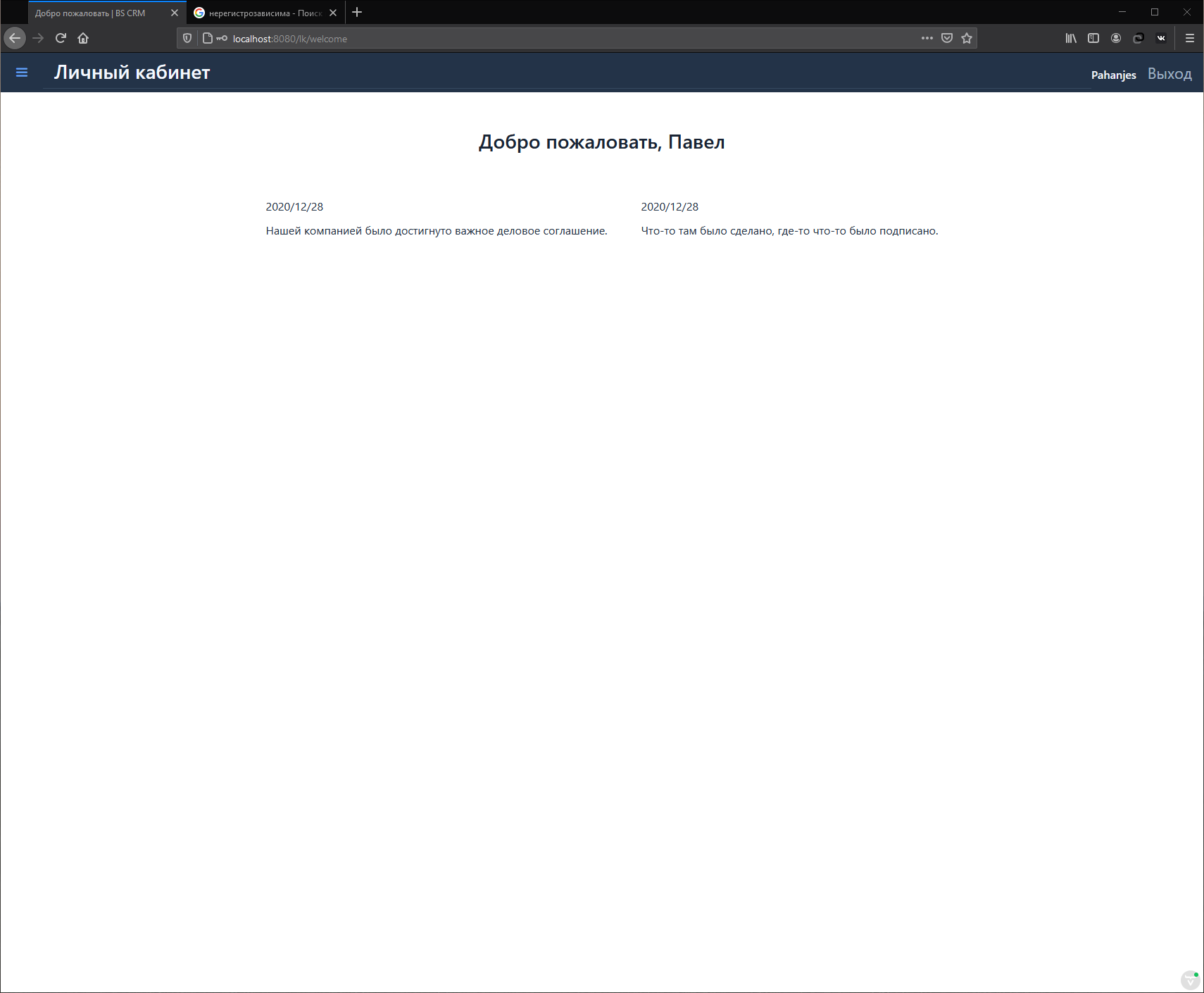


Рис.4 – Успешная авторизация пользователя

Для удобной навигации по личному кабинету реализовано выплывающее меню по левому краю личного кабинета:

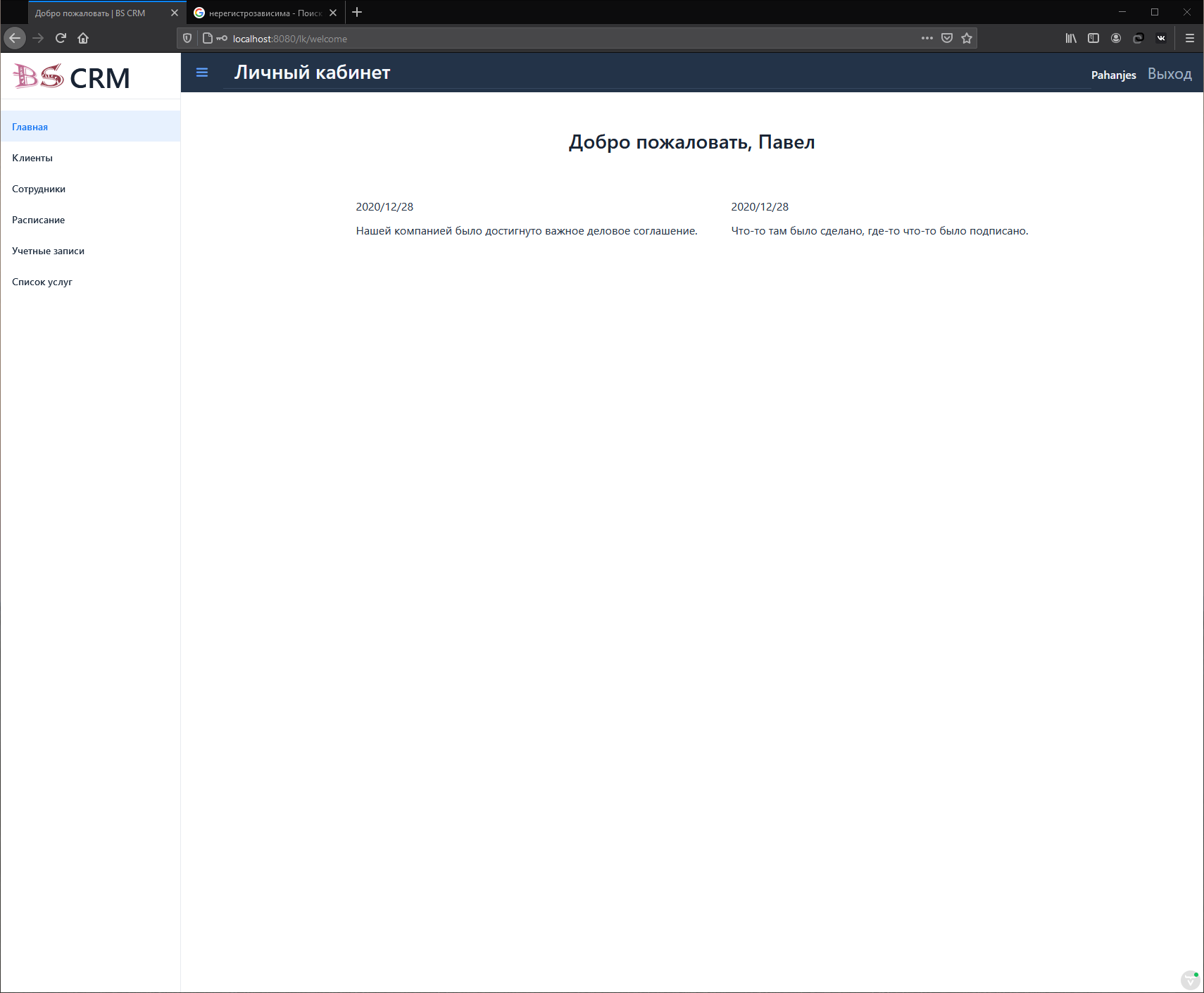


Рис.5 – Навигационное меню личного кабинета пользователя с ролью ADMIN

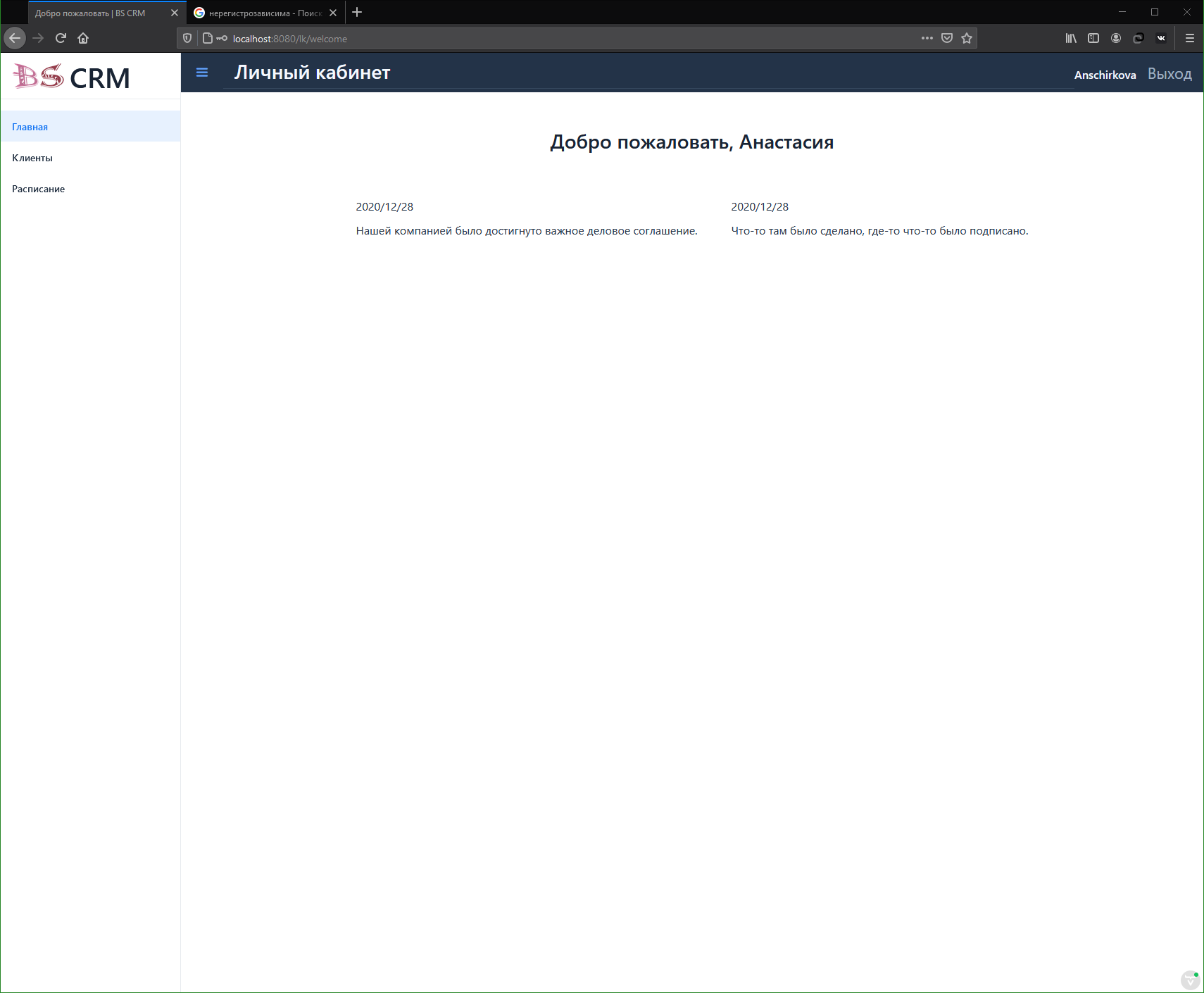


Рис.6 – Навигационное меню личного кабинета пользователя с ролью USER

Так как CRM-модель должна гарантировать хранение и обработку информации о клиентах компании, был реализован соответствующий функционал, позволяющий:

* просматривать и редактировать информацию о всех клиентах компании;
* удалять и добавлять клиентов;
* искать клиентов по фильтру имя/фамилия;
* сортировать клиентов в порядке возрастания/убывания по алфавиту/числам по фамилии, имени, номеру телефона, адресу электронной почты;

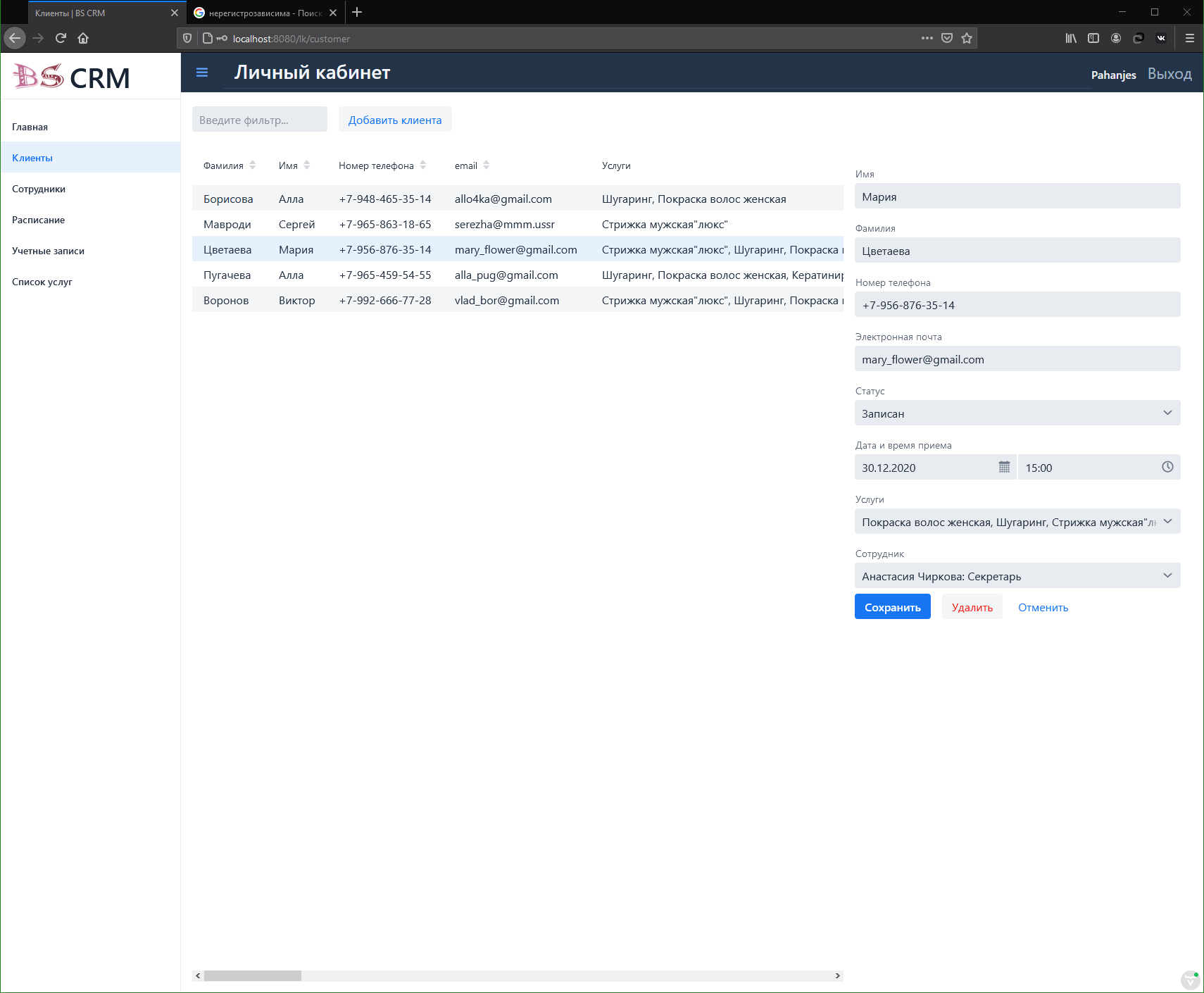


Рис.7 – Просмотр и редактирование информации клиентов компании

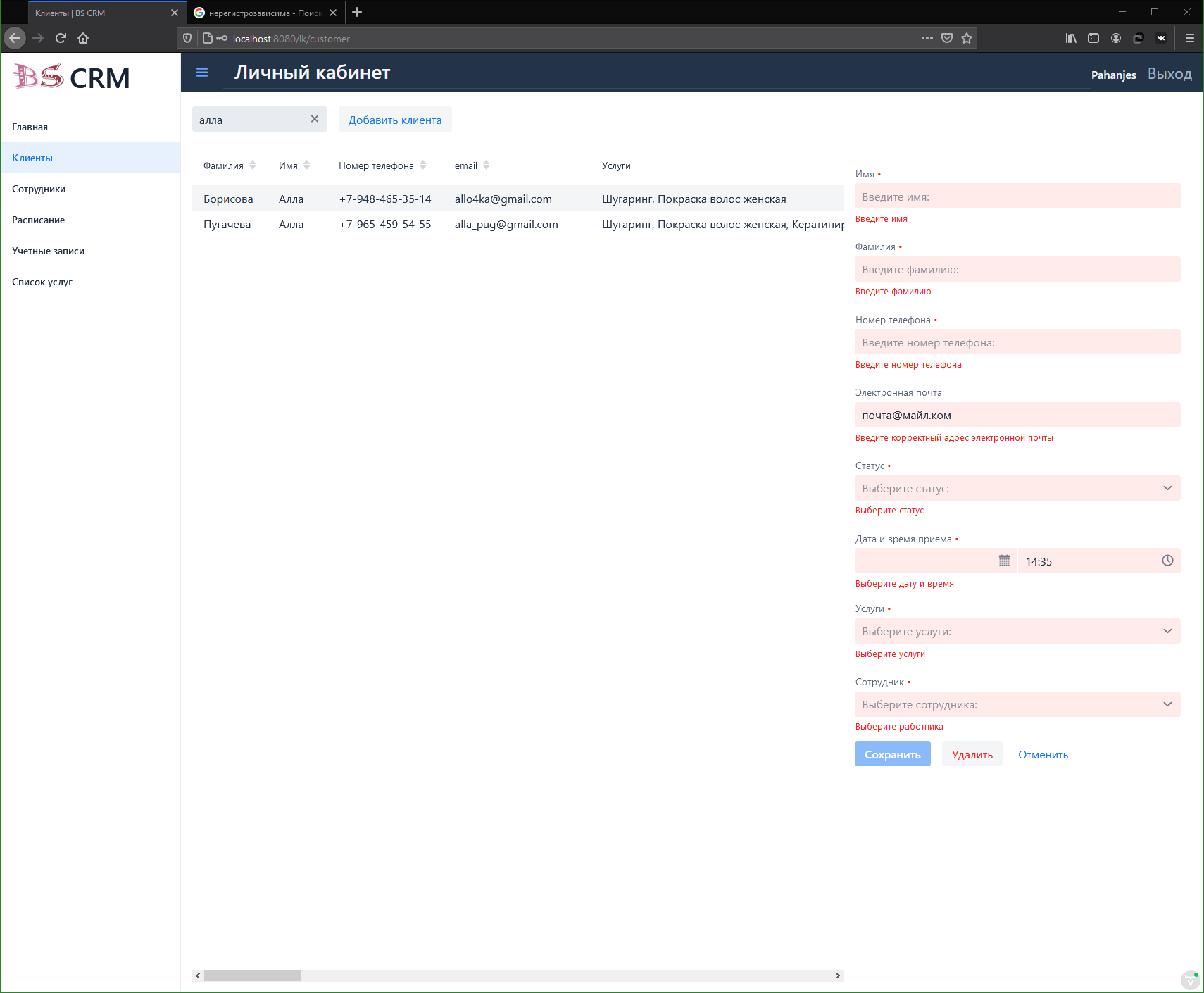


Рис.8 – Добавление клиента в компанию и поиск по имени

Подобный функционал был реализован и для работы с сотрудниками компании. При выборе сотрудника сразу можно увидеть его логин и роль пользователя в системе, которые в режиме только чтение. При добавлении нового сотрудника режим чтения выключается и появляется возможность их ввести в соответствующие поля:

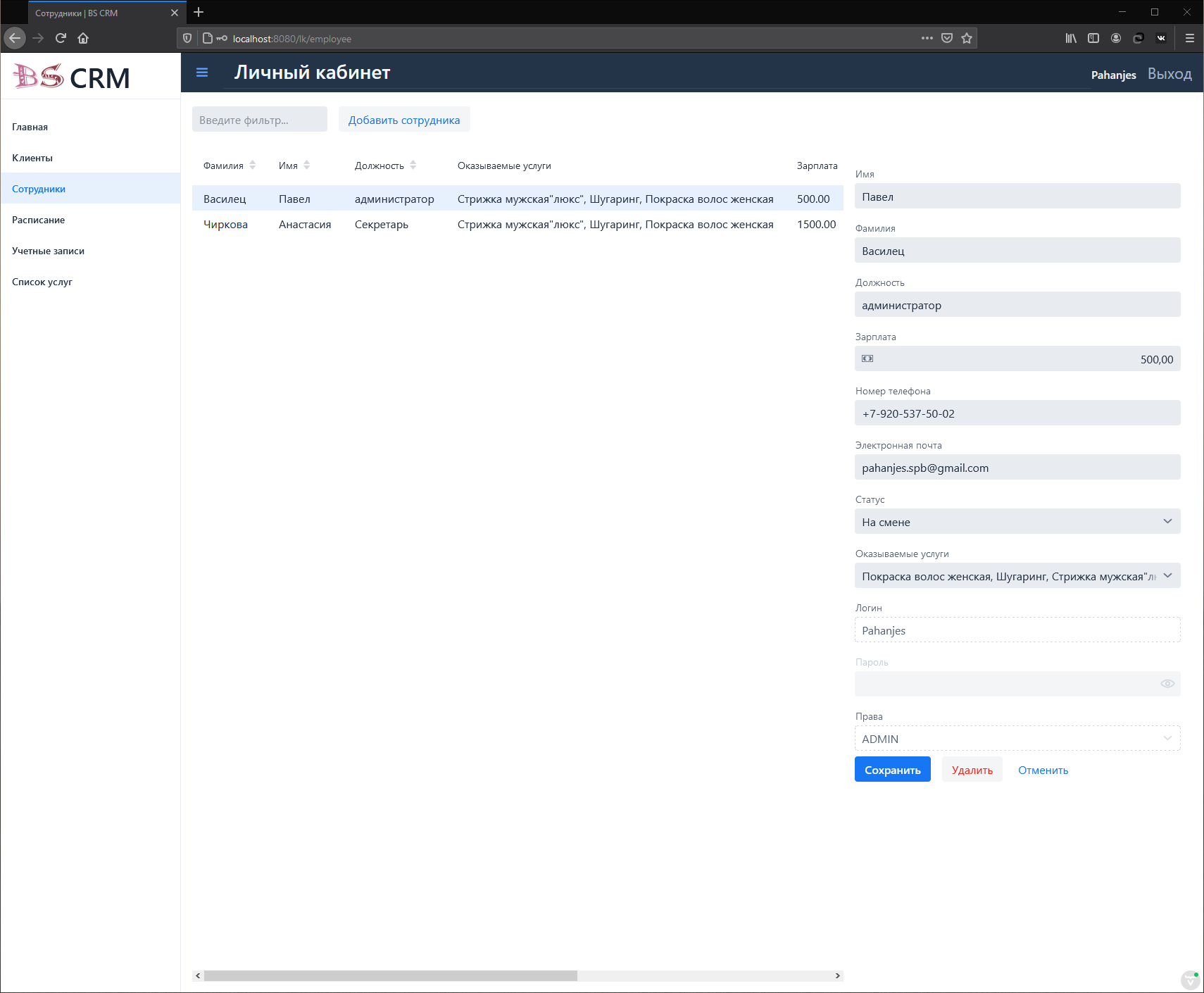


Рис.9 – Редактирование сотрудника компании

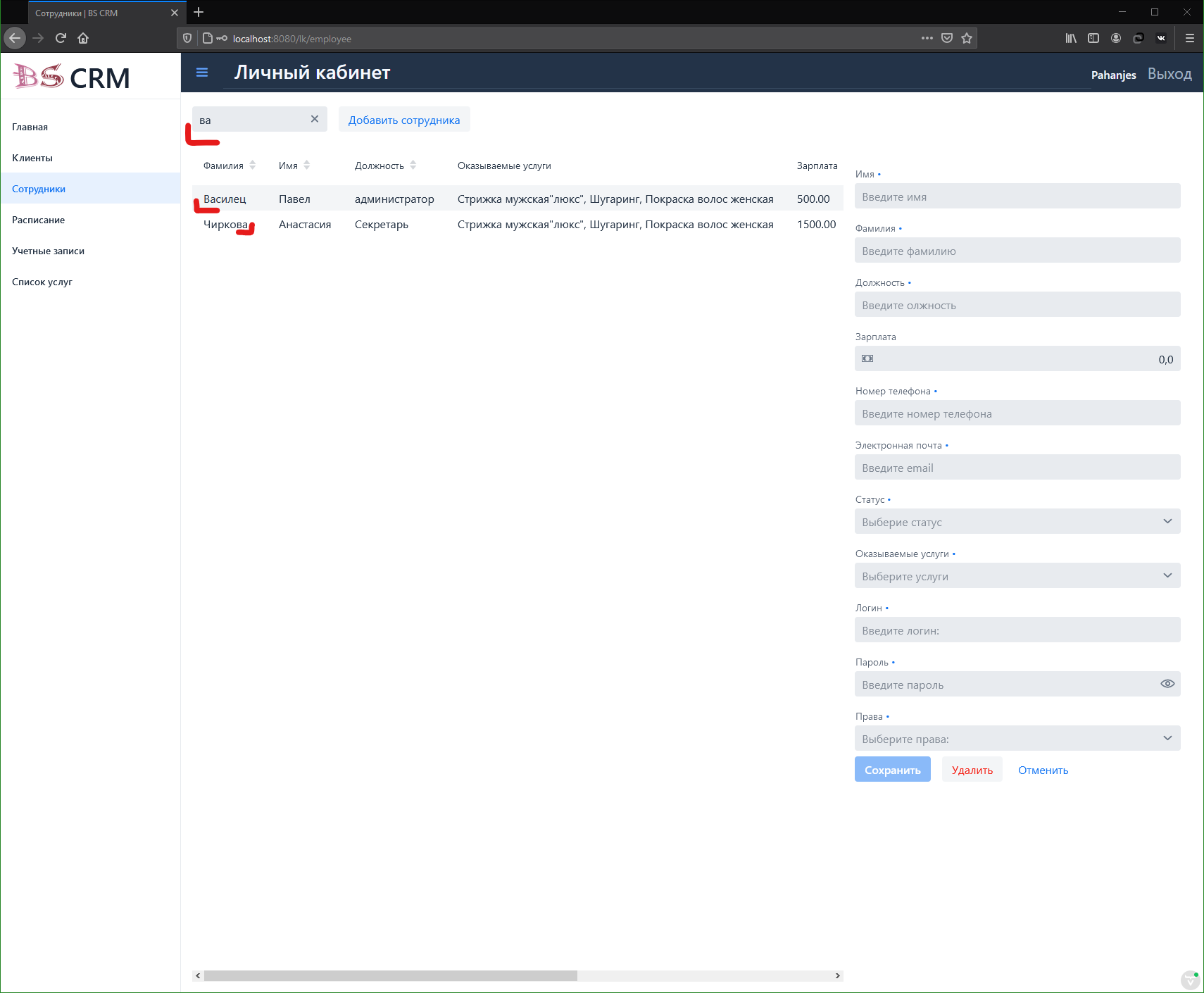


Рис.10 – Добавление нового пользователя и поиск о фамилии

Для удобства сотрудников была добавлена возможность просмотра своего расписания. В нем отображаются все клиенты, которые когда-либо были записаны на данного сотрудника. Так же присутствует сортировка по дате, времени, фамилии, имени, статусу и номеру телефона:

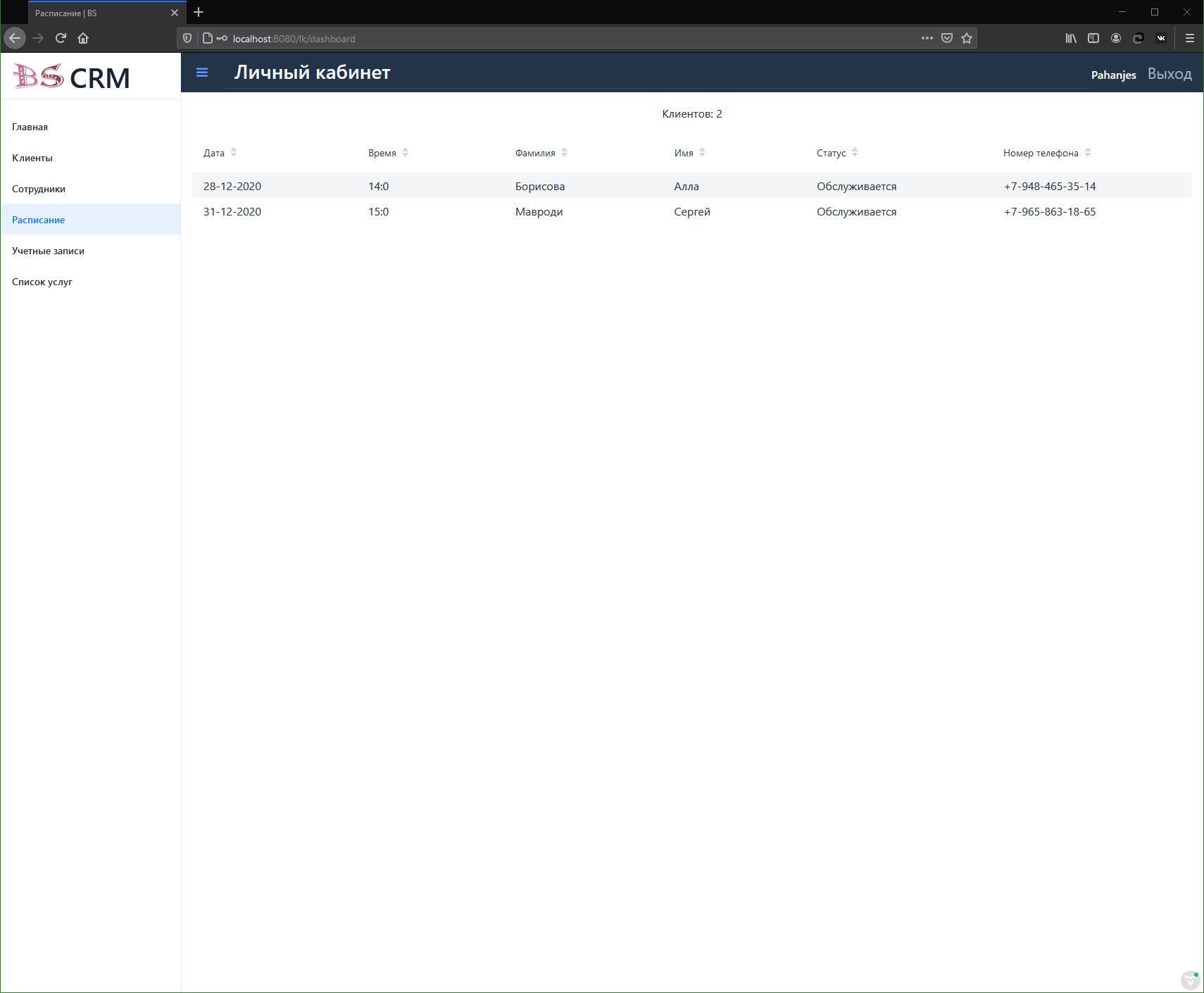


Рис.11 – Просмотр расписания сотрудника

Редактирование учетных записей было вынесено в отдельное меню, в котором представлен список всех учетных записей в системе, а также полная информация об этих учетных записях с возможностью ее редактирования.

Пользователь с ролью ADMIN имеет возможность удалять других пользователей, у которых любая роль. Но пользователь не может удалить сам себя. Это ограничение было введено с целью обеспечения целостности и работоспособности системы.

Опционально присутствует возможность изменить пароль учетной записи. Просмотреть его невозможно по той причине, что в базе данных он хранится в зашифрованном виде для защиты безопасности учетных записей в случае несанкционированного доступа к базе данных системы:

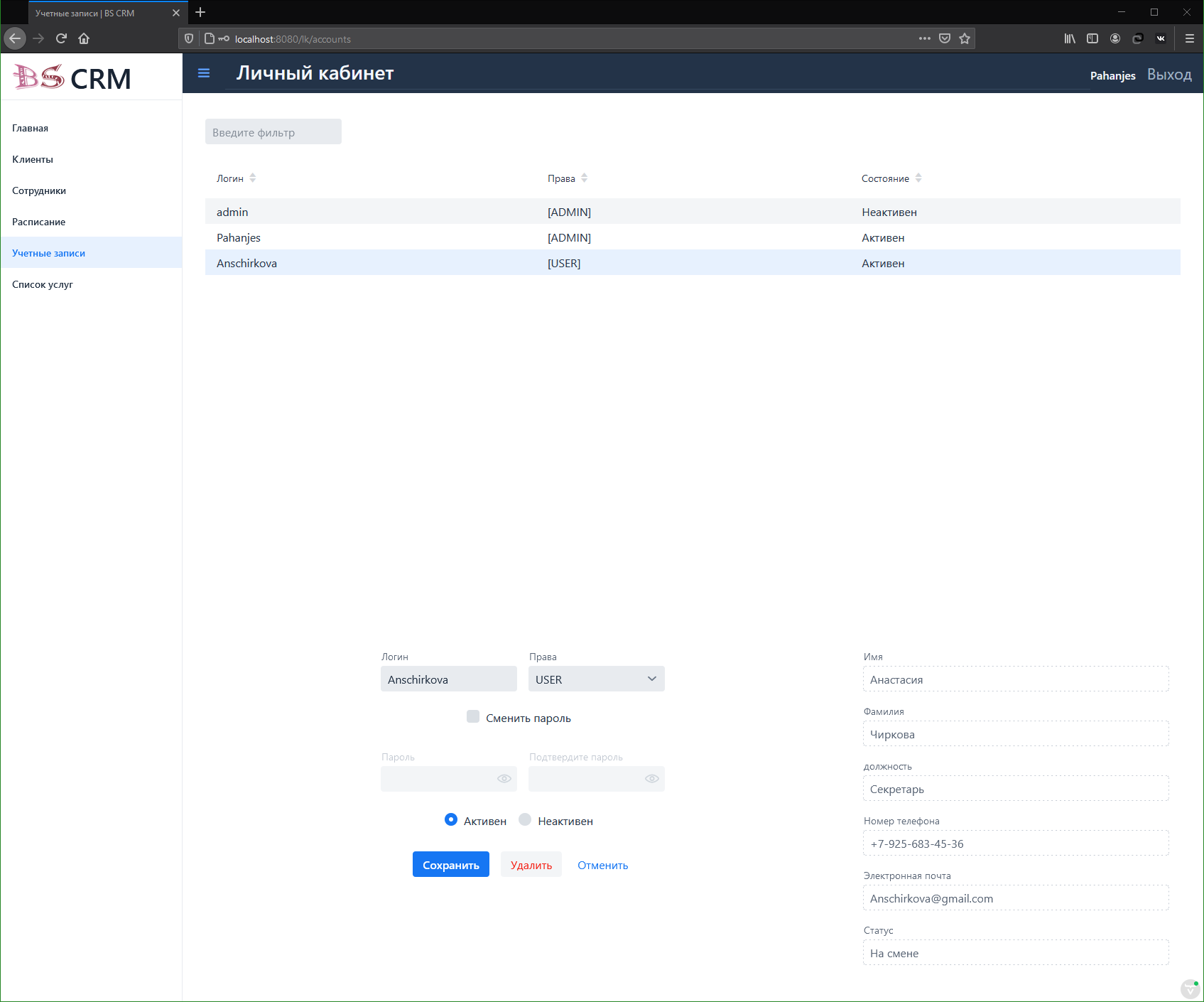


Рис.12 – Просмотр и редактирование учетных записей системы

Отдельным пунктом меню является «Список услуг» в котором в удобном виде хранится список всех услуг, предоставляемых компанией.

Услуги можно добавлять, редактировать и деактивировать на случай, если по каким-то причинам компания временно не может их предоставлять. В таком случае, если клиент попытается записаться, выбрав хотя бы одну из деактивированных услуг, система не даст ему это сделать, заботливо указав, какая именно услуга в данный момент недоступна:

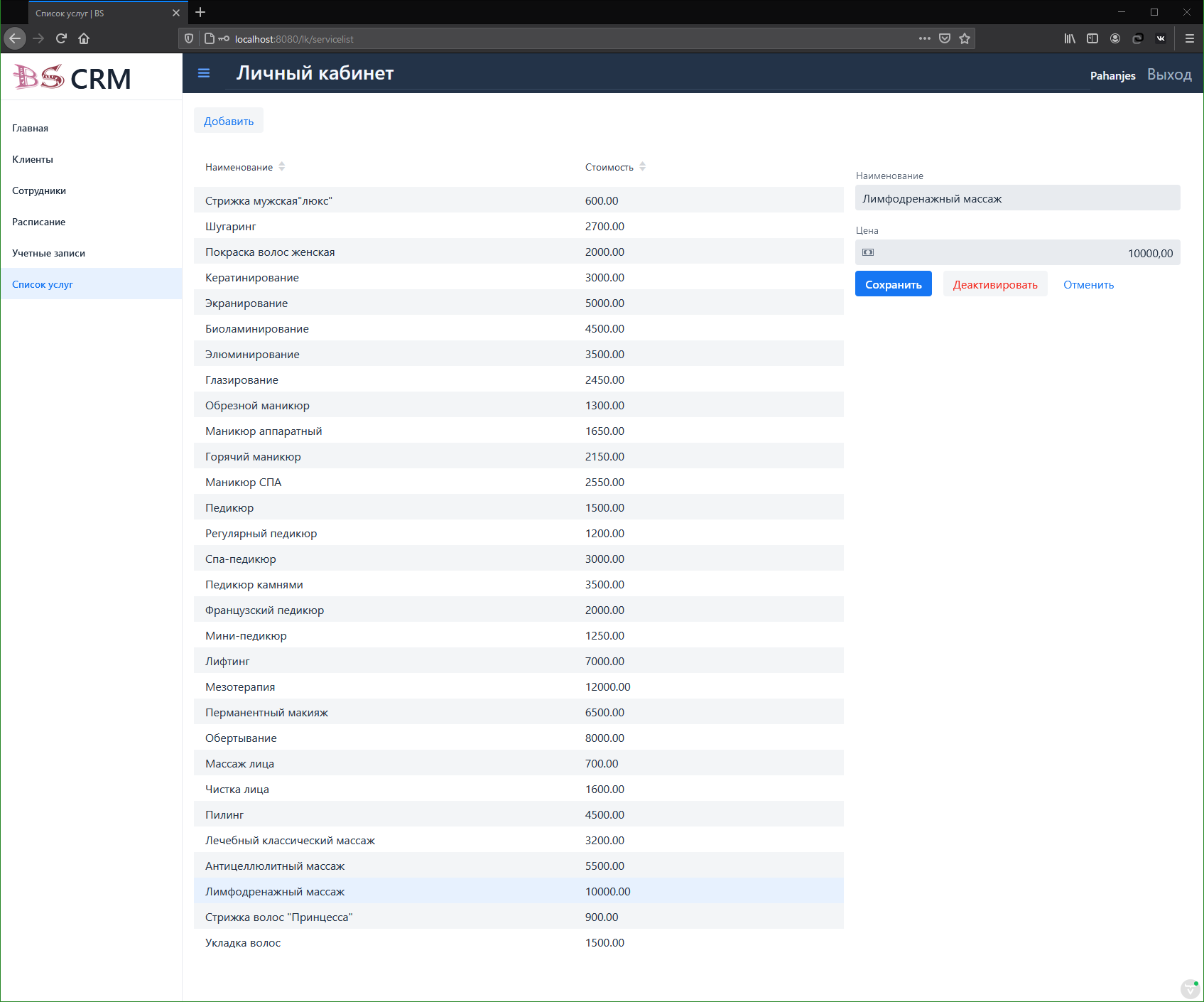


Рис.13 – Просмотр и редактирование услуг компании

Так как CRM-модель требует, чтобы клиент сам имел возможность взаимодействовать с компанией без участника посредника в лице обслуживающего персонала, был реализован главный «сайт-затычка» компании, на котором пользователю предоставляется возможность записаться на прием. Помимо этого, на главном сайте реализовано меню, позволяющее войти в CRM-систему компании или пройти обратно на главный сайт:

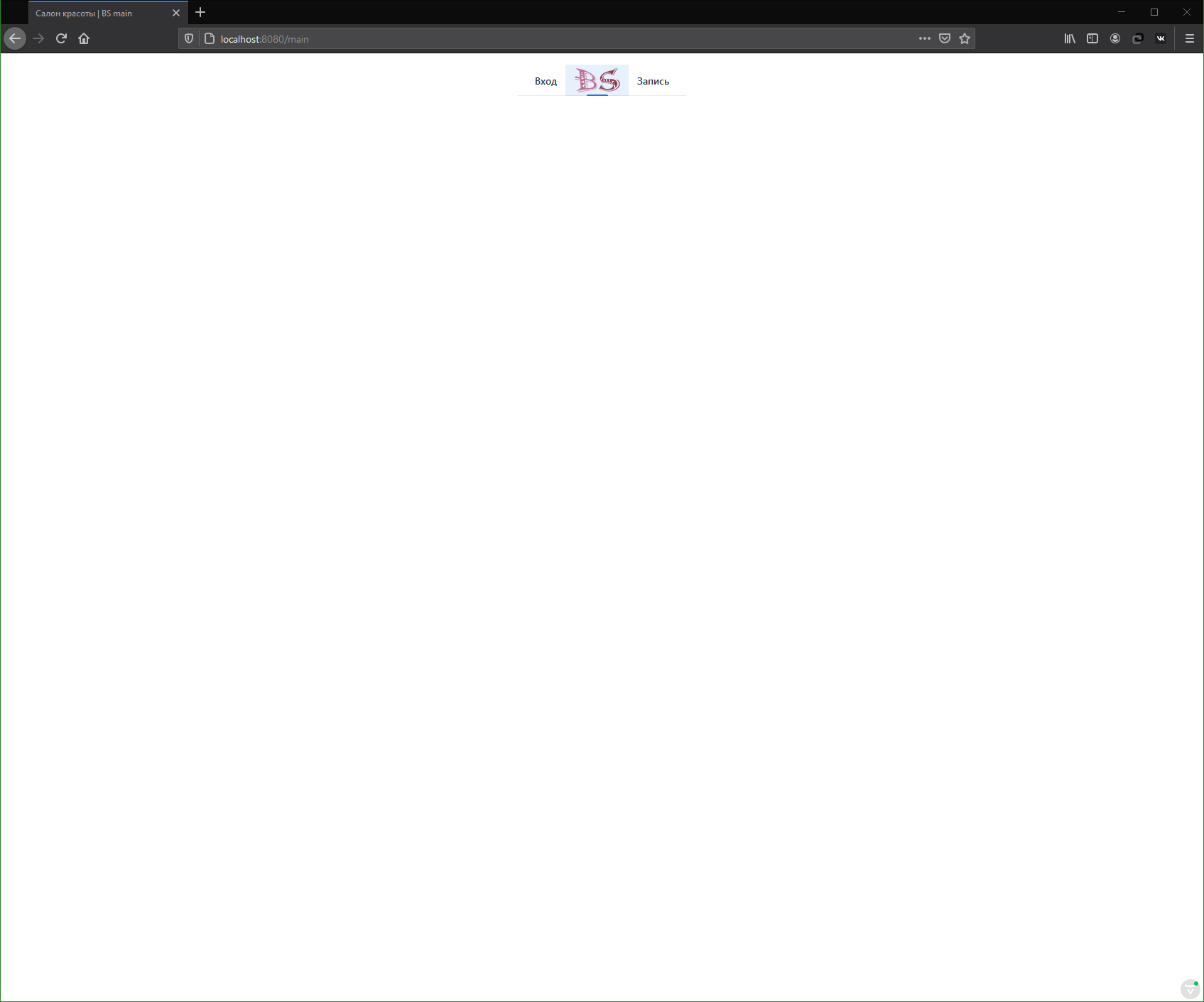


Рис.14 – Главный сайт компании с возможностью записаться на прием или авторизоваться

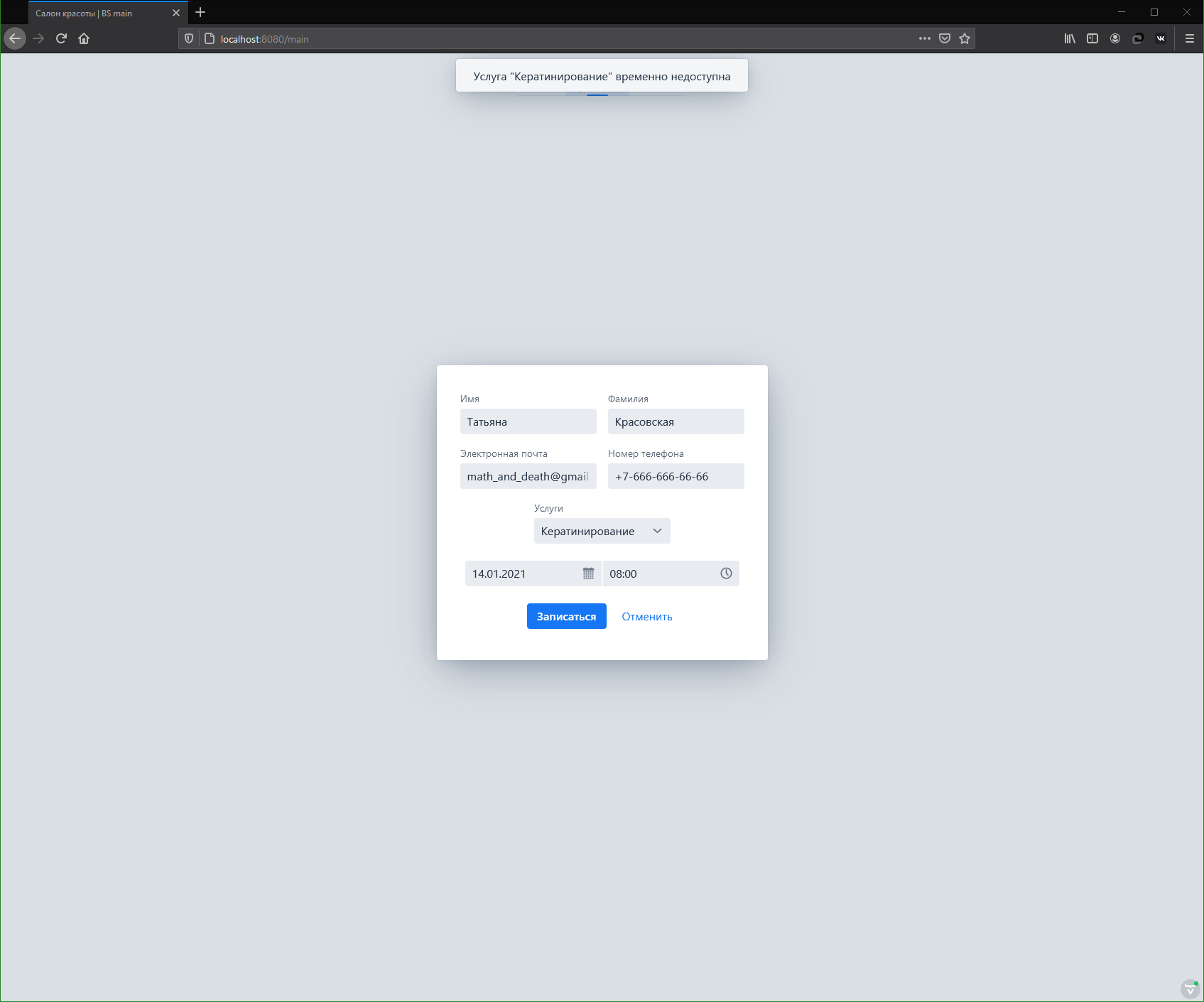


Рис.16 – Запись клиента на главном сайте компании

### **4.3.3 Оценка результатов исследования**

Результаты исследования показывают, что несмотря на увеличивающиеся конкуренцию и требования компаний-заказчиков не теряется возможность разработать приложение, реализующую CRM-модель и отвечающую как бизнес-требованиям, так и функциональным.

Для этого существуют удобные и гибкие технологии проектирования и разработки подобных высоконагруженных функциональных систем.

Данное приложение в полном объеме решает поставленную в данном исследовании задачу.

С точки зрения технико-экономической эффективности результаты исследования говорят о том, что внедрение CRM-модели в современном бизнесе в качестве главной модели более чем оправдано.

# **§5. Заключение**

В результате проведенных исследований стал очевидным тот факт, что современные проблемы в социо-экономической сфере также влияют и на многие другие сферы, в частности IT. Современный бизнес, во многом мигрирующий в онлайн, повышает конкуренцию и спрос на приложения, реализующие CRM-модель. Это говорит о том, что может быть еще некоторое множество смежных с IT областей, которые нуждаются в дополнительном исследовании, так как результаты таких исследований показывают реальную востребованность тех или иных существующих в IT технологий.

# **§6. Список использованных источников**

1. Vaadin: docs. – URL: <https://vaadin.com/docs/index.html> (дата обращения: 28.12.2020)
2. Java: JDK 15 Documentation. – URL: <https://docs.oracle.com/en/java/javase/15/> (дата обращения: 28.12.2020)
3. Spring: Guides. – URL: <https://spring.io/guides> (дата обращения: 28.12.2020)
4. Stackoverflow: questions. - URL: <https://stackoverflow.com/questions> (дата обращения: 28.12.2020)

# **§7. Приложения**

## **7.1 Приложение A**

**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

**Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования   
«Санкт-Петербургский государственный университет телекоммуникаций   
им. проф. М. А. Бонч-Бруевича»**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Факультет ИКСС

Кафедра ПИиВТ

Приложение А

(обаятельное)

к курсовому проекту

по дисциплине:

Разработка Java-приложение управления телекоммуникациями

Тема: «Разработка приложения для салона красоты»

Вариант: 97

Группа: ИКПИ-84

Студент: Василец Павел Васильевич

Руководитель: Помогалова Альбина Владимировна

Оценка: \_\_\_\_\_\_\_\_

Санкт-Петербург

2020

## **7.1.1 Листинг AbstractEntity.java**

package ru.pahanjes.beautysaloon.crm.backend.entity;  
  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.MappedSuperclass;  
  
@MappedSuperclass  
public abstract class AbstractEntity {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private Long id;  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public boolean isPersisted() {  
 return id != null;  
 }  
  
 @Override  
 public int hashCode() {  
 if (getId() != null) {  
 return getId().hashCode();  
 }  
 return super.hashCode();  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) {  
 return true;  
 }  
 if (obj == null) {  
 return false;  
 }  
 if (getClass() != obj.getClass()) {  
 return false;  
 }  
 AbstractEntity other = (AbstractEntity) obj;  
 if (getId() == null || other.getId() == null) {  
 return false;  
 }  
 return getId().equals(other.getId());  
 }  
}

## **7.1.2 Листинг Customer.java**

package ru.pahanjes.beautysaloon.crm.backend.entity;  
  
import javax.persistence.\*;  
import javax.validation.constraints.Email;  
import javax.validation.constraints.NotEmpty;  
import javax.validation.constraints.NotNull;  
import java.time.LocalDateTime;  
import java.util.Set;  
  
@Entity  
public class Customer extends AbstractEntity implements Cloneable {  
  
 public enum Status {  
 *SignedUp* ("Записан"),  
 *CanseldTheEntry* ("Запись была отменена"),  
 *Served* ("Обслуживается"),  
 *HasBeenServed* ("Обслуживание завершено");  
  
 private final String value;  
  
 Status(String value){  
 this.value = value;  
 }  
  
 public String getValue() {  
 return value;  
 }  
 }  
  
 @NotNull  
 @NotEmpty  
 private String firstName = "";  
  
 @NotNull  
 @NotEmpty  
 private String lastName = "";  
   
 @ManyToOne(fetch = FetchType.*EAGER*)  
 @JoinColumn(name = "employee\_id")  
 private Employee employee;  
  
 @ManyToMany(fetch = FetchType.*EAGER*, cascade = {CascadeType.*REFRESH*})  
 @JoinTable(  
 name = "customer\_service",  
 joinColumns = @JoinColumn(name = "customer\_id"),  
 inverseJoinColumns = @JoinColumn(name = "service\_id")  
 )  
 private Set<Service> services;  
  
 @Enumerated(EnumType.*STRING*)  
 @NotNull  
 private Customer.Status status;  
  
 @Email  
 @NotNull  
 @NotEmpty  
 private String email = "";  
  
 @NotNull  
 @NotEmpty  
 private String phoneNumber = "";  
  
 @Column(name = "timetable")  
 private java.time.LocalDateTime timetable;  
  
 public Customer(@NotNull @NotEmpty String firstName, @NotNull @NotEmpty String lastName, Employee employee, Set<Service> services,  
 @NotNull Customer.Status status, @Email @NotNull @NotEmpty String email, @NotNull @NotEmpty String phoneNumber,  
 LocalDateTime timetable) {  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.employee = employee;  
 this.services = services;  
 this.status = status;  
 this.email = email;  
 this.phoneNumber = phoneNumber;  
 this.timetable = timetable;  
 }  
  
 public Customer() {  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public Status getStatus() {  
 return status;  
 }  
  
 public void setStatus(Status status) {  
 this.status = status;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getPhoneNumber() {  
 return phoneNumber;  
 }  
  
 public void setPhoneNumber(String phoneNumber) {  
 this.phoneNumber = phoneNumber;  
 }  
  
 public void setEmployee(Employee employee) {  
 this.employee = employee;  
 }  
  
 public Employee getEmployee() {  
 return employee;  
 }  
  
 public LocalDateTime getTimetable() {  
 return timetable;  
 }  
  
 public void setTimetable(LocalDateTime timetable) {  
 this.timetable = timetable;  
 }  
  
 public Set<Service> getServices() {  
 return services;  
 }  
  
 public void setServices(Set<Service> services) {  
 this.services = services;  
 }  
  
 public void addService(Service service) {  
 this.services.add(service);  
 }  
  
 public void removeService(Service service) {  
 this.services.remove(service);  
 }  
  
 @Override  
 public String toString() {  
 return firstName + " " + lastName;  
 }  
}

## **7.1.3 Листинг Employee.java**

package ru.pahanjes.beautysaloon.crm.backend.entity;  
  
import javax.persistence.\*;  
import javax.validation.constraints.Email;  
import javax.validation.constraints.NotEmpty;  
import javax.validation.constraints.NotNull;  
import java.math.BigDecimal;  
import java.util.Set;  
  
@Entity  
public class Employee extends AbstractEntity{  
  
 public enum Status {  
 *CurrentlyWorking* ("На смене"),  
 *LunchBreak* ("На обеде"),  
 *WaitingForTheCustomer* ("Ожидает клиента"),  
 *OnVacation* ("В отпуске"),  
 *fired* ("Уволен");  
  
 private final String value;  
  
 static public int count() {  
 return 5;  
 }  
  
 Status(String value){  
 this.value = value;  
 }  
  
 public String getValue(){  
 return value;  
 }  
 }  
  
 @OneToMany(mappedBy = "employee", fetch = FetchType.*EAGER*)//LAZY  
 private Set<Customer> customers;  
  
 @OneToOne(cascade = {CascadeType.*REMOVE*}, fetch = FetchType.*EAGER*)//LAZY  
 @JoinColumn(name = "user\_id")  
 private User user;  
  
 @ManyToMany(fetch = FetchType.*EAGER*, cascade = {CascadeType.*MERGE*})  
 @JoinTable(  
 name = "employee\_service",  
 joinColumns = @JoinColumn(name = "employee\_id"),  
 inverseJoinColumns = @JoinColumn(name = "service\_id")  
 )  
 private Set<Service> services;  
  
 public Employee() {  
  
 }  
  
 public Employee(@NotNull @NotEmpty String firstName, @NotNull @NotEmpty String lastName, @NotNull Employee.Status status,  
 @Email @NotNull @NotEmpty String email, @NotNull @NotEmpty String phoneNumber,  
 @NotNull @NotEmpty String position, @NotNull BigDecimal salary  
 ) {  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.status = status;  
 this.email = email;  
 this.phoneNumber = phoneNumber;  
 this.position = position;  
 this.salary = salary;  
 }  
  
 public Employee(Set<Customer> customers, User user, Set<Service> services, @NotNull @NotEmpty String firstName,  
 @NotNull @NotEmpty String lastName, @NotNull Employee.Status status, @Email @NotNull @NotEmpty String email,  
 @NotNull @NotEmpty String phoneNumber, @NotNull @NotEmpty String position, @NotNull BigDecimal salary  
 ) {  
 this.customers = customers;  
 this.user = user;  
 this.services = services;  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.status = status;  
 this.email = email;  
 this.phoneNumber = phoneNumber;  
 this.position = position;  
 this.salary = salary;  
 }  
  
 @NotNull  
 @NotEmpty  
 private String firstName = "";  
  
 @NotNull  
 @NotEmpty  
 private String lastName = "";  
  
 @Enumerated(EnumType.*STRING*)  
 @NotNull  
 private Employee.Status status;  
  
 @Email  
 @NotNull  
 @NotEmpty  
 private String email = "";  
  
 @NotNull  
 @NotEmpty  
 private String phoneNumber = "";  
  
 @NotNull  
 @NotEmpty  
 private String position = "";  
  
 @NotNull  
 private BigDecimal salary = new BigDecimal("0.0");  
  
 public Set<Customer> getCustomers() {  
 return customers;  
 }  
  
 public void setCustomers(Set<Customer> customers) {  
 this.customers = customers;  
 }  
  
 public void addCustomer(Customer customer) {  
 this.customers.add(customer);  
 }  
  
 public void removeCustomer(Customer customer) {  
 this.customers.remove(customer);  
 }  
  
 public void clearCustomers() {  
 this.customers.clear();  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
 public Status getStatus() {  
 return status;  
 }  
  
 public void setStatus(Status status) {  
 this.status = status;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public String getPhoneNumber() {  
 return phoneNumber;  
 }  
  
 public void setPhoneNumber(String phoneNumber) {  
 this.phoneNumber = phoneNumber;  
 }  
  
 public String getPosition() {  
 return position;  
 }  
  
 public void setPosition(String position) {  
 this.position = position;  
 }  
  
 public String getFullNameWithPosition() {  
 return firstName + ' ' + lastName + ": " + position;  
 }  
  
 public User getUser() {  
 return user;  
 }  
  
 public void setUser(User user) {  
 this.user = user;  
 }  
  
 public BigDecimal getSalary() {  
 return salary;  
 }  
  
 public void setSalary(BigDecimal sal) {  
 this.salary = sal;  
 }  
  
 public Set<Service> getServices() {  
 return services;  
 }  
  
 public void setServices(Set<Service> services) {  
 this.services = services;  
 }  
  
 public void addService(Service service) {  
 this.services.add(service);  
 }  
  
 public void removeService(Service service) {  
 this.services.remove(service);  
 }  
  
}

## **7.1.4 Листинг Role.java**

package ru.pahanjes.beautysaloon.crm.backend.entity;  
  
import org.springframework.security.core.GrantedAuthority;  
  
public enum Role implements GrantedAuthority {  
  
 *USER*, *ADMIN*;  
  
 @Override  
 public String getAuthority() {  
 return name();  
 }  
}

## **7.1.5 Листинг Service.java**

package ru.pahanjes.beautysaloon.crm.backend.entity;  
  
import javax.persistence.\*;  
import javax.validation.constraints.NotNull;  
import java.math.BigDecimal;  
  
@Entity  
public class Service extends AbstractEntity {  
  
 @Column(name = "service")  
 private String service;  
  
 @NotNull  
 private BigDecimal price;  
  
 @NotNull  
 boolean isActive = true;  
  
 public Service(String service, @NotNull BigDecimal price) {  
 this.service = service;  
 this.price = price;  
 }  
  
 public Service(String service, @NotNull BigDecimal price, @NotNull boolean isActive) {  
 this.service = service;  
 this.price = price;  
 this.isActive = isActive;  
 }  
  
 public Service() {  
  
 }  
  
 public String getService() {  
 return service;  
 }  
  
 public void setService(String service) {  
 this.service = service;  
 }  
  
 public BigDecimal getPrice() {  
 return price;  
 }  
  
 public void setPrice(BigDecimal price) {  
 this.price = price;  
 }  
  
 public boolean isActive() {  
 return isActive;  
 }  
  
 public void setActive(boolean active) {  
 isActive = active;  
 }  
}

## **7.1.6 Листинг User.java**

package ru.pahanjes.beautysaloon.crm.backend.entity;  
  
import org.apache.commons.codec.digest.DigestUtils;  
import org.apache.commons.lang3.RandomStringUtils;  
import org.springframework.security.core.GrantedAuthority;  
import org.springframework.security.core.userdetails.UserDetails;  
  
import javax.persistence.\*;  
import java.util.Collection;  
import java.util.Set;  
  
@Entity  
@Table(name = "usr")  
public class User extends AbstractEntity implements UserDetails {  
  
 private String username;  
 private final String passwordSalt = RandomStringUtils.*random*(32);  
 private String password;  
  
 private boolean isActive;  
   
 @OneToOne(mappedBy = "user")  
 private Employee employee;  
  
 @ElementCollection(targetClass = Role.class, fetch = FetchType.*EAGER*)  
 @CollectionTable(name = "user\_role", joinColumns = @JoinColumn(name = "user\_id"))  
 @Enumerated(EnumType.*STRING*)  
 private Set<Role> role;  
  
 public User() {  
 }  
  
 public User(String username, String password, boolean isActive, Employee employee, Set<Role> role) {  
 this.username = username;  
 setPassword(password);  
 this.isActive = isActive;  
 this.employee = employee;  
 this.role = role;  
 }  
  
 @Override  
 public Collection<? extends GrantedAuthority> getAuthorities() {  
 return getRole();  
 }  
  
 @Override  
 public String getPassword() {  
 return password;  
 }  
  
 @Override  
 public String getUsername() {  
 return username;  
 }  
  
 @Override  
 public boolean isAccountNonExpired() {  
 return true;  
 }  
  
 @Override  
 public boolean isAccountNonLocked() {  
 return true;  
 }  
  
 @Override  
 public boolean isCredentialsNonExpired() {  
 return true;  
 }  
  
 @Override  
 public boolean isEnabled() {  
 return isActive();  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public String getPasswordSalt() {  
 return passwordSalt;  
 }  
  
 public void setPassword(String password) {  
 this.password = DigestUtils.*sha1Hex*(password + passwordSalt);  
 }  
  
 public boolean isActive() {  
 return isActive;  
 }  
  
 public void setActive(boolean active) {  
 isActive = active;  
 }  
  
 public Set<Role> getRole() {  
 return role;  
 }  
  
 public void setRole(Set<Role> role) {  
 this.role = role;  
 }  
  
 public Employee getEmployee() {  
 return employee;  
 }  
  
 public void setEmployee(Employee employee) {  
 this.employee = employee;  
 }  
  
 public boolean checkPassword(String password) {  
 return DigestUtils.*sha1Hex*(password + passwordSalt).equals(getPassword());  
 }  
  
}

## **7.1.7 Листинг CustomerRepository.java**

package ru.pahanjes.beautysaloon.crm.backend.repository;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.query.Param;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Customer;  
  
import java.util.List;  
  
public interface CustomerRepository extends JpaRepository<Customer, Long> {  
 @Query("Select c from Customer c " +  
 "where lower(c.firstName) like lower(concat('%', :searchTerm, '%')) " +  
 "or lower(c.lastName) like lower(concat('%', :searchTerm, '%'))")  
 List<Customer> search(@Param("searchTerm") String searchTerm);  
  
 @Query("select c from Customer c where c.id= :searchTerm")  
 Customer search(@Param("searchTerm") Long id);  
}

## **7.1.8 Листинг CustomerRepository.java**

package ru.pahanjes.beautysaloon.crm.backend.repository;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.query.Param;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Employee;  
  
import java.util.List;  
  
public interface EmployeeRepository extends JpaRepository<Employee, Long> {  
 @Query("Select e from Employee e " +  
 "where lower(e.firstName) like lower(concat('%', :searchTerm, '%')) " +  
 "or lower(e.lastName) like lower(concat('%', :searchTerm, '%'))")  
 List<Employee> search(@Param("searchTerm") String searchTerm);  
  
 @Query("Select e from Employee e where e.id= :searchTerm")  
 Employee search(@Param("searchTerm") Long id);  
}

## **7.1.9 Листинг ServiceRepository.java**

package ru.pahanjes.beautysaloon.crm.backend.repository;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Service;  
  
public interface ServiceRepository extends JpaRepository<Service, Long> {  
}

## **7.1.10 Листинг UserRepository.java**

package ru.pahanjes.beautysaloon.crm.backend.repository;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.query.Param;  
import ru.pahanjes.beautysaloon.crm.backend.entity.User;  
  
import java.util.List;  
  
public interface UserRepository extends JpaRepository<User, Long> {  
  
 User findByUsername(String username);  
  
 @Query("Select u from User u where lower(u.username) like lower(concat('%', :searchTerm, '%'))")  
 List<User> search(@Param("searchTerm") String searchTerm);  
  
 @Query("Select u from User u where u.id= :searchTerm")  
 User search(@Param("searchTerm") Long id);  
}

## **7.1.11 Листинг CustomerService.java**

package ru.pahanjes.beautysaloon.crm.backend.service;  
  
import org.springframework.stereotype.Service;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Customer;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Employee;  
import ru.pahanjes.beautysaloon.crm.backend.repository.CustomerRepository;  
import ru.pahanjes.beautysaloon.crm.backend.repository.EmployeeRepository;  
  
import java.util.List;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
@Service  
public class CustomerService {  
 private static final Logger *LOGGER* = Logger.*getLogger*(CustomerService.class.getName());  
 private CustomerRepository customerRepository;  
 private EmployeeRepository employeeRepository;  
  
 public CustomerService(CustomerRepository customerRepository,  
 EmployeeRepository employeeRepository) {  
 this.customerRepository = customerRepository;  
 this.employeeRepository = employeeRepository;  
 }  
  
 public List<Customer> findAll() {  
 return customerRepository.findAll();  
 }  
  
 public List<Customer> findAll(String filter) {  
 if(filter == null || filter.isEmpty()){  
 return customerRepository.findAll();  
 } else {  
 return customerRepository.search(filter);  
 }  
 }  
  
 public Customer findById(Long id) {  
 return customerRepository.search(id);  
 }  
  
 public Long count() {  
 return customerRepository.count();  
 }  
  
 public void delete(Customer customer) {  
 if (customer.getEmployee() != null) {  
 Employee employee = customer.getEmployee();  
 employee.removeCustomer(customer);  
 employeeRepository.save(employee);  
 customer.setEmployee(null);  
 }  
 customerRepository.delete(customer);  
 }  
  
 public void save(Customer customer) {  
 if(customer == null) {  
 *LOGGER*.log(Level.*SEVERE*,  
 "customer\_error");  
 return;  
 }  
 customerRepository.save(customer);  
 }  
}

## **7.1.12 Листинг EmployeeService.java**

package ru.pahanjes.beautysaloon.crm.backend.service;  
  
import org.springframework.stereotype.Service;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Employee;  
import ru.pahanjes.beautysaloon.crm.backend.repository.EmployeeRepository;  
  
import java.util.List;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
@Service  
public class EmployeeService {  
  
 private static final Logger *LOGGER* = Logger.*getLogger*(EmployeeService.class.getName());  
 private EmployeeRepository employeeRepository;  
  
 public EmployeeService(EmployeeRepository employeeRepository) {  
 this.employeeRepository = employeeRepository;  
 }  
  
 public List<Employee> findAll() {  
 return employeeRepository.findAll();  
 }  
  
 public List<Employee> findAllWithoutAccount() {  
 List<Employee> employeesWithoutAccount = employeeRepository.findAll();  
 employeesWithoutAccount.removeIf(employee -> employee.getUser() != null);  
 return employeesWithoutAccount;  
 }  
  
 public List<Employee> findAll(String filter) {  
 if(filter == null || filter.isEmpty()){  
 return employeeRepository.findAll();  
 } else {  
 return employeeRepository.search(filter);  
 }  
 }  
  
 public Employee findById(Long id) {  
 return employeeRepository.search(id);  
 }  
  
 public void delete(Employee employee) {  
 if (employee.getCustomers().size() != 0) employee.clearCustomers();  
 employeeRepository.save(employee);  
 employeeRepository.delete(employee);  
 }  
  
 public void save(Employee employee) {  
 if(employee == null) {  
 *LOGGER*.log(Level.*SEVERE*,  
 "Клиент пуст. Вы уверены, что хотите подключить форму к приложению?");  
 return;  
 }  
 employeeRepository.save(employee);  
 }  
}

## **7.1.13 Листинг ServiceService.java**

package ru.pahanjes.beautysaloon.crm.backend.service;  
  
  
import ru.pahanjes.beautysaloon.crm.backend.entity.Service;  
import ru.pahanjes.beautysaloon.crm.backend.repository.EmployeeRepository;  
import ru.pahanjes.beautysaloon.crm.backend.repository.ServiceRepository;  
  
import java.util.List;  
  
@org.springframework.stereotype.Service  
public class ServiceService {  
 private ServiceRepository serviceRepository;  
 private EmployeeRepository employeeRepository;  
  
 public ServiceService(ServiceRepository serviceRepository, EmployeeRepository employeeRepository) {  
 this.serviceRepository = serviceRepository;  
 this.employeeRepository = employeeRepository;  
 }  
  
 public List<Service> findAll() {  
 return serviceRepository.findAll();  
 }  
  
 public Long count() {  
 return serviceRepository.count();  
 }  
  
 public void delete(Service service) {  
 serviceRepository.delete(service);  
 }  
  
 public void save(Service service) {  
 serviceRepository.save(service);  
 }  
}

## **7.1.14 Листинг UserService.java**

package ru.pahanjes.beautysaloon.crm.backend.service;  
  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.core.userdetails.UsernameNotFoundException;  
import org.springframework.stereotype.Service;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Customer;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Employee;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Role;  
import ru.pahanjes.beautysaloon.crm.backend.entity.User;  
import ru.pahanjes.beautysaloon.crm.backend.repository.UserRepository;  
  
import javax.annotation.PostConstruct;  
import java.math.BigDecimal;  
import java.util.Collections;  
import java.util.List;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
@Service  
public class UserService implements UserDetailsService {  
  
 private final UserRepository userRepository;  
 private final EmployeeService employeeService;  
 private final ServiceService serviceService;  
 private final CustomerService customerService;  
 private static final Logger *LOGGER* = Logger.*getLogger*(CustomerService.class.getName());  
  
 public UserService(UserRepository userRepository, EmployeeService employeeService, ServiceService serviceService, CustomerService customerService) {  
 this.userRepository = userRepository;  
 this.employeeService = employeeService;  
 this.serviceService = serviceService;  
 this.customerService = customerService;  
 }  
  
 public List<User> findAll() {  
 return userRepository.findAll();  
 }  
  
 public List<User> findAll(String filter) {  
 if (filter == null || filter.isEmpty()) {  
 return userRepository.findAll();  
 } else {  
 return userRepository.search(filter);  
 }  
 }  
  
 public User findById(Long id) {  
 return userRepository.search(id);  
 }  
  
 @Override  
 public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {  
 return userRepository.findByUsername(username);  
 }  
  
 @PostConstruct  
 public void createDefaultUser() {  
 if(userRepository.count() == 0) {  
 User user = new User("user", "pass", true, null, Collections.*singleton*(Role.*USER*));  
 User admin = new User("admin", "pass", true, null, Collections.*singleton*(Role.*ADMIN*));  
 userRepository.save(user);  
 userRepository.save(admin);  
 }  
 if(customerService.findAll().size() == 0) {  
 Customer customerWomen = new Customer(  
 "Алла",  
 "Борисова",  
 null,  
 null,  
 Customer.Status.*Served*,  
 "allo4ka@gmail.com",  
 "+7-948-465-35-14",  
 null  
 );  
 Customer customerMen = new Customer(  
 "Сергей",  
 "Мавроди",  
 null,  
 null,  
 Customer.Status.*Served*,  
 "serezha@mmm.ussr",  
 "+7-965-863-18-65",  
 null  
 );  
 customerService.save(customerWomen);  
 customerService.save(customerMen);  
 }  
 if(serviceService.findAll().size() == 0) {  
 ru.pahanjes.beautysaloon.crm.backend.entity.Service haircut = new ru.pahanjes.beautysaloon.crm.backend.entity.Service(  
 "Стрижка мужская\"люкс\"",  
 new BigDecimal("600.0"),  
 true  
 );  
 ru.pahanjes.beautysaloon.crm.backend.entity.Service shugaring = new ru.pahanjes.beautysaloon.crm.backend.entity.Service(  
 "Шугаринг",  
 new BigDecimal("2700.0"),  
 true  
 );  
 ru.pahanjes.beautysaloon.crm.backend.entity.Service hairPaint = new ru.pahanjes.beautysaloon.crm.backend.entity.Service(  
 "Покраска волос женская",  
 new BigDecimal("2000.0"),  
 true  
 );  
 serviceService.save(haircut);  
 serviceService.save(shugaring);  
 serviceService.save(hairPaint);  
 }  
 if(employeeService.findAll().size() == 0) {  
 Employee administrator = new Employee(  
 "Павел",  
 "Василец",  
 Employee.Status.*CurrentlyWorking*,  
 "pahanjes.spb@gmail.com",  
 "+7-920-537-50-02",  
 "администратор",  
 new BigDecimal("500.0")  
 );  
 Employee secretary = new Employee(  
 "Анастасия",  
 "Чиркова",  
 Employee.Status.*CurrentlyWorking*,  
 "Anschirkova@gmail.com",  
 "+7-925-683-45-36",  
 "Секретарь",  
 new BigDecimal("1500.0")  
 );  
 employeeService.save(administrator);  
 employeeService.save(secretary);  
 }  
 }  
  
 public void delete(User user) {  
 if(user.getEmployee() != null) {  
 if(user.getEmployee().getCustomers().size() != 0) {  
 user.getEmployee().getCustomers().forEach(customer -> {  
 customer.setEmployee(null);  
 customerService.save(customer);  
 });  
 }  
 employeeService.delete(user.getEmployee());  
 }  
 userRepository.delete(user);  
 }  
  
 public void save(User user) {  
 if (user == null){  
 *LOGGER*.log(Level.*SEVERE*,  
 "error");  
 return;  
 }  
 userRepository.save(user);  
 }  
}

## **7.1.15 Листинг AuthException.java**

package ru.pahanjes.beautysaloon.crm.exception;  
  
public class AuthException extends Exception{  
 public AuthException(String errorMessage) {  
 super(errorMessage);  
 }  
}

## **7.1.16 Листинг AuthorizedRoute.java**

package ru.pahanjes.beautysaloon.crm.security;  
  
import com.vaadin.flow.component.Component;  
  
public class AuthorizedRoute {  
  
 private String Route;  
 private String name;  
 private Class<? extends Component> view;  
  
 public AuthorizedRoute(String route, String name, Class<? extends Component> view) {  
 Route = route;  
 this.name = name;  
 this.view = view;  
 }  
  
 public String getRoute() {  
 return Route;  
 }  
  
 public void setRoute(String route) {  
 Route = route;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public Class<? extends Component> getView() {  
 return view;  
 }  
  
 public void setView(Class<? extends Component> view) {  
 this.view = view;  
 }  
}

## **7.1.17 Листинг AuthService.java**

package ru.pahanjes.beautysaloon.crm.security;  
  
import com.vaadin.flow.router.RouteConfiguration;  
import com.vaadin.flow.server.VaadinSession;  
import org.springframework.stereotype.Service;  
import ru.pahanjes.beautysaloon.crm.UI.views.accounts.AccountsView;  
import ru.pahanjes.beautysaloon.crm.UI.views.cabinet.CabinetApp;  
import ru.pahanjes.beautysaloon.crm.UI.views.cabinet.WelcomeLayout;  
import ru.pahanjes.beautysaloon.crm.UI.views.dashboard.DashBoardView;  
import ru.pahanjes.beautysaloon.crm.UI.views.list.CustomerListView;  
import ru.pahanjes.beautysaloon.crm.UI.views.list.EmployeeListView;  
import ru.pahanjes.beautysaloon.crm.UI.views.service.ServiceView;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Employee;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Role;  
import ru.pahanjes.beautysaloon.crm.backend.entity.User;  
import ru.pahanjes.beautysaloon.crm.backend.repository.EmployeeRepository;  
import ru.pahanjes.beautysaloon.crm.backend.repository.UserRepository;  
import ru.pahanjes.beautysaloon.crm.exception.AuthException;  
  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.List;  
import java.util.Set;  
  
@Service  
public class AuthService extends Exception {  
  
 private AuthorizedRoute route;  
 private final UserRepository userRepository;  
 private EmployeeRepository employeeRepository;  
  
 public AuthService(UserRepository userRepository, EmployeeRepository employeeRepository){  
 this.userRepository = userRepository;  
 this.employeeRepository = employeeRepository;  
 }  
  
 public void authenticate(String username, String password) throws AuthException {  
 User user = userRepository.findByUsername(username);  
 if (user != null && user.checkPassword(password)) {  
 if(!user.isEnabled()) {  
 throw new AuthException("Пользователь временно заблокирован");  
 }  
 VaadinSession.*getCurrent*().setAttribute(User.class, user);  
 createRoutes(user.getRole());  
 } else {  
 throw new AuthException("Неверный логин или пароль");  
 }  
 }  
  
 public int register(String username, String password, Employee employee, Role role) {  
 if(userRepository.findByUsername(username) != null) {  
 return -1;  
 }  
 User user = new User();  
 user.setUsername(username);  
 user.setPassword(password);  
 user.setActive(true);  
 user.setRole(Collections.*singleton*(role));  
 employee.setUser(user);  
 userRepository.save(user);  
 employeeRepository.save(employee);  
 return 0;  
 }  
  
 private void createRoutes(Set<Role> role) {  
 getAuthorizedRoutes(role).stream()  
 .forEach(route ->  
 RouteConfiguration.*forSessionScope*().setRoute(  
 route.getRoute(), route.getView(), CabinetApp.class));  
 }  
  
 public List<AuthorizedRoute> getAuthorizedRoutes(Set<Role> role) {  
 ArrayList<AuthorizedRoute> routes = new ArrayList<>();  
  
 if (role.toArray()[0].equals(Role.*USER*)){  
 routes.add(new AuthorizedRoute("lk/welcome", "Главная", WelcomeLayout.class));  
 routes.add(new AuthorizedRoute("lk/customer", "Клиенты", CustomerListView.class));  
 routes.add(new AuthorizedRoute("lk/dashboard", "Расписание", DashBoardView.class));  
 } else if (role.toArray()[0].equals(Role.*ADMIN*)) {  
 routes.add(new AuthorizedRoute("lk/welcome", "Главная", WelcomeLayout.class));  
 routes.add(new AuthorizedRoute("lk/customer", "Клиенты", CustomerListView.class));  
 routes.add(new AuthorizedRoute("lk/employee", "Сотрудники", EmployeeListView.class));  
 routes.add(new AuthorizedRoute("lk/dashboard", "Расписание", DashBoardView.class));  
 routes.add(new AuthorizedRoute("lk/accounts", "Учетные записи", AccountsView.class));  
 routes.add(new AuthorizedRoute("lk/servicelist", "Список услуг", ServiceView.class));  
 }  
 return routes;  
 }  
}

## **7.1.18 Листинг CustomRequestCache.java**

package ru.pahanjes.beautysaloon.crm.security;  
  
import org.springframework.security.web.savedrequest.HttpSessionRequestCache;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
  
class CustomRequestCache extends HttpSessionRequestCache {  
  
 // Сохраняет запросы, не прошедшие проверку подлинности, чтобы перенаправить пользователя  
 // На страницу, к которой он пытался получить доступ, после входа в систему.  
 @Override  
 public void saveRequest(HttpServletRequest request, HttpServletResponse response) {  
 if (!SecurityUtils.*isFrameworkInternalRequest*(request)) {  
 super.saveRequest(request, response);  
 }  
 }  
  
}

## **7.1.19 Листинг SecurityConfiguration.java**

package ru.pahanjes.beautysaloon.crm.security;  
  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.builders.WebSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
import org.springframework.security.crypto.password.NoOpPasswordEncoder;  
import ru.pahanjes.beautysaloon.crm.backend.service.UserService;  
  
@EnableWebSecurity  
@Configuration  
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {  
  
 /\*  
 \* URL to enter the account in process  
 \*/  
 private static final String *LOGIN\_PROCESSING\_URL* = "/login";  
 /\*  
 \* URL to enter the account  
 \*/  
 private static final String *LOGIN\_URL* = "/logout";  
 /\*  
 \* URL to first page after logout  
 \*/  
 private static final String *LOGOUT\_SUCCESS\_URL* = "/login";  
  
 private final UserService userService;  
  
 public SecurityConfiguration(UserService userService) {  
 this.userService = userService;  
 }  
  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 // Отключает защиту от подделки межсайтовых запросов (CSRF), так как Vaadin уже имеет защиту CSRF.  
 http.csrf().disable()  
 // Использует CustomRequestCache для отслеживания неавторизованных запросов,  
 // Чтобы пользователи перенаправлялись соответствующим образом после входа в систему.  
 .requestCache().requestCache(new CustomRequestCache())  
 // Включает авторизацию.  
 .and()  
 .authorizeRequests()  
 .antMatchers("/main").permitAll()  
 // Разрешает весь внутренний трафик из фреймворка Vaadin.  
 .requestMatchers(SecurityUtils::*isFrameworkInternalRequest*).permitAll()  
 // Разрешает весь аутентифицированный трафик.  
 .anyRequest().authenticated()  
 // Включает вход на основе формы и разрешает неаутентифицированный доступ к нему.  
 .and().formLogin()  
 .loginPage(*LOGIN\_URL*).permitAll()  
 // Настраивает URL-адреса страницы входа.  
 .loginProcessingUrl(*LOGIN\_PROCESSING\_URL*)  
 .defaultSuccessUrl("/lk/welcome")  
 // Настраивает URL-адрес выхода.  
 .and()  
 .logout().  
 logoutSuccessUrl(*LOGOUT\_SUCCESS\_URL*)  
 ;  
 }  
  
 @Override  
 protected void configure(AuthenticationManagerBuilder auth) throws Exception {  
 auth.userDetailsService(userService)  
 .passwordEncoder(NoOpPasswordEncoder.*getInstance*())  
 ;  
 }  
  
 // Исключение связь Vaadin-framework и статических активов из Spring Security.  
 @Override  
 public void configure(WebSecurity web) {  
 web.ignoring().antMatchers(  
 "/VAADIN/\*\*",  
 "/favicon.ico",  
 "/robots.txt",  
 "/manifest.webmanifest",  
 "/sw.js",  
 "/offline.html",  
 "/icons/\*\*",  
 "/images/\*\*",  
 "/styles/\*\*",  
 "/frontend/\*\*"  
 );  
 }  
  
}

## **7.1.20 Листинг SecurityUtils.java**

package ru.pahanjes.beautysaloon.crm.security;  
  
import com.vaadin.flow.server.ServletHelper;  
import com.vaadin.flow.shared.ApplicationConstants;  
import org.springframework.security.authentication.AnonymousAuthenticationToken;  
import org.springframework.security.core.Authentication;  
import org.springframework.security.core.context.SecurityContextHolder;  
  
import javax.servlet.http.HttpServletRequest;  
import java.util.stream.Stream;  
  
public final class SecurityUtils {  
  
 private SecurityUtils() {  
 // Util methods only  
 }  
  
 // Определяет является ли запрос внутрненним  
 static boolean isFrameworkInternalRequest(HttpServletRequest request) {  
 final String parameterValue = request.getParameter(ApplicationConstants.*REQUEST\_TYPE\_PARAMETER*);  
 return parameterValue != null  
 && Stream.*of*(ServletHelper.RequestType.*values*())  
 .anyMatch(r -> r.getIdentifier().equals(parameterValue));  
 }  
  
 // Проверяет вошел ли текущий пользователь в систему  
 static boolean isUserLoggedIn() {  
 Authentication authentication = SecurityContextHolder.*getContext*().getAuthentication();  
 return authentication != null  
 && !(authentication instanceof AnonymousAuthenticationToken)  
 && authentication.isAuthenticated();  
 }  
}

## **7.1.21 Листинг AccountsView.java**

package ru.pahanjes.beautysaloon.crm.UI.views.accounts;  
  
import com.vaadin.flow.component.Component;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.combobox.ComboBox;  
import com.vaadin.flow.component.html.Div;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.tabs.Tab;  
import com.vaadin.flow.component.tabs.Tabs;  
import com.vaadin.flow.component.textfield.PasswordField;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import ru.pahanjes.beautysaloon.crm.UI.views.accounts.change.ChangeView;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Employee;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Role;  
import ru.pahanjes.beautysaloon.crm.backend.service.CustomerService;  
import ru.pahanjes.beautysaloon.crm.backend.service.EmployeeService;  
import ru.pahanjes.beautysaloon.crm.backend.service.UserService;  
import ru.pahanjes.beautysaloon.crm.security.AuthService;  
  
import java.util.HashMap;  
import java.util.Map;  
  
@Route(value = "lk/accounts")  
@PageTitle("Учетные записи | BS CRM")  
public class AccountsView extends VerticalLayout {  
  
 private Tabs subMenu = new Tabs();  
 private TextField username = new TextField("Имя пользователя");  
 private PasswordField password = new PasswordField("Пароль");  
 private PasswordField confirmPassword = new PasswordField("Подтвердите пароль");  
 private ComboBox<Employee> employees = new ComboBox<>("Сотрудник");  
 private ComboBox<Role> role = new ComboBox<>("Права");  
 private Button register = new Button("Зарегистрировать");  
 private HorizontalLayout firstLayout = new HorizontalLayout();  
 private HorizontalLayout secondLayout = new HorizontalLayout();  
 Div content = new Div();  
 private ChangeView changeView;  
 private AuthService authService;  
 private EmployeeService employeeService;  
 private UserService userService;  
 private CustomerService customerService;  
  
 public AccountsView(AuthService authService, EmployeeService employeeService, UserService userService, CustomerService customerService){  
 this.authService = authService;  
 this.employeeService = employeeService;  
 this.userService = userService;  
 this.customerService = customerService;  
 addClassName("accounts-view");  
 setSizeFull();  
  
 setAlignItems(Alignment.*CENTER*);  
 changeView = new ChangeView(userService, employeeService, customerService);  
  
 content.setId("content");  
 content.setSizeFull();  
  
 createMenuTabs(subMenu, content);  
 subMenu.setId("subMenu");  
  
 add(  
 changeView  
 );  
 }  
  
 public void createMenuTabs(Tabs tabs, Div div) {  
 Tab tab1 = new Tab("Регистрация");  
 Tab tab2 = new Tab("Редактирование");  
 Div div2 = new Div(changeView);  
 div2.setVisible(false);  
  
 Map<Tab, Component> tabsToPages = new HashMap<>();  
 tabsToPages.put(tab2, div2);  
 tabs.add(tab2);  
 div.add(div2);  
  
 tabs.setSelectedIndex(0);  
 tabs.addSelectedChangeListener(selected -> {  
 tabsToPages.values().forEach(page -> page.setVisible(false));  
 tabsToPages.get(tabs.getSelectedTab()).setVisible(true);  
 });  
 }  
  
}

## **7.1.22 Листинг ChangeView.java**

package ru.pahanjes.beautysaloon.crm.UI.views.accounts.change;  
  
import com.vaadin.flow.component.dependency.CssImport;  
import com.vaadin.flow.component.formlayout.FormLayout;  
import com.vaadin.flow.component.grid.Grid;  
import com.vaadin.flow.component.grid.GridVariant;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.EmailField;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.data.value.ValueChangeMode;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Employee;  
import ru.pahanjes.beautysaloon.crm.backend.entity.User;  
import ru.pahanjes.beautysaloon.crm.backend.service.CustomerService;  
import ru.pahanjes.beautysaloon.crm.backend.service.EmployeeService;  
import ru.pahanjes.beautysaloon.crm.backend.service.UserService;  
  
@CssImport("./styles/views/change-view.css")  
public class ChangeView extends VerticalLayout {  
  
 private class ShowEmployeeForm extends FormLayout {  
  
 private TextField firstName = new TextField("Имя");  
 private TextField lastName = new TextField("Фамилия");  
 private TextField position = new TextField("должность");  
 private EmailField email = new EmailField("Электронная почта");  
 private TextField phoneNumber = new TextField("Номер телефона");  
 private TextField status = new TextField("Статус");  
 private Employee employee;  
  
 public ShowEmployeeForm(){  
 addClassName("employee-form");  
 add(  
 firstName,  
 lastName,  
 position,  
 phoneNumber,  
 email,  
 status  
 );  
 }  
  
 public ShowEmployeeForm(Employee employee){  
 addClassName("employee-form");  
 configureForm();  
 add(  
 firstName,  
 lastName,  
 position,  
 phoneNumber,  
 email,  
 status  
 );  
 }  
  
 private void configureForm() {  
  
 firstName.setValue(employee.getFirstName());  
 firstName.setReadOnly(true);  
 lastName.setValue(employee.getLastName());  
 lastName.setReadOnly(true);  
 position.setValue(employee.getPosition());  
 position.setReadOnly(true);  
 email.setValue(employee.getEmail());  
 email.setReadOnly(true);  
 phoneNumber.setValue(employee.getPhoneNumber());  
 phoneNumber.setReadOnly(true);  
 status.setValue(employee.getStatus().getValue());  
 status.setReadOnly(true);  
 }  
  
 private void setFieldsReadOnly() {  
 firstName.setReadOnly(true);  
 lastName.setReadOnly(true);  
 position.setReadOnly(true);  
 email.setReadOnly(true);  
 phoneNumber.setReadOnly(true);  
 status.setReadOnly(true);  
 }  
  
 public Employee getEmployee() {  
 return employee;  
 }  
  
 public void setEmployee(Employee employee) {  
 this.employee = employee;  
 setFieldsReadOnly();  
 if(employee != null) {  
 configureForm();  
 }  
 }  
 }  
  
 private Grid<User> userGrid = new Grid<>(User.class);  
 private TextField filter = new TextField();  
 private UserService userService;  
 private EmployeeService employeeService;  
 private CustomerService customerService;  
 private ChangeUserForm userForm;  
 private ShowEmployeeForm employeeForm;  
 private HorizontalLayout userInfoAndAboutLayout = new HorizontalLayout();  
  
 public ChangeView (UserService userService, EmployeeService employeeService, CustomerService customerService){  
 this.userService = userService;  
 this.employeeService = employeeService;  
 this.customerService = customerService;  
 setJustifyContentMode(JustifyContentMode.*CENTER*);  
 addClassName("change-view");  
 userForm = new ChangeUserForm();  
 userForm.addListener(ChangeUserForm.SaveUserEvent.class, this::saveUser);  
 userForm.addListener(ChangeUserForm.DeleteUserEvent.class, this::deleteUser);  
 userForm.addListener(ChangeUserForm.CloseUserEvent.class, closeUserEvent -> closeEditor());  
 userForm.addClassName("user-form");  
 employeeForm = new ShowEmployeeForm();  
 employeeForm.addClassName("employee-form");  
 setSizeFull();  
 configureGrid();  
 configureLayout();  
 add(getToolBar(), userGrid, userInfoAndAboutLayout);  
 updateUserGrid();  
 }  
  
 private void configureGrid() {  
 userGrid.addClassName("user-grid");  
 userGrid.removeAllColumns();  
 userGrid.addColumn(user -> user.getUsername() == null ? "-" : user.getUsername()).setHeader("Логин").setSortable(true);  
 userGrid.addColumn(user -> user.getRole() == null ? "-" : user.getRole()).setHeader("Права").setSortable(true);  
 userGrid.addColumn(user -> user.isActive() ? "Активен" : "Неактивен").setHeader("Состояние").setSortable(true);  
 userGrid.getColumns().forEach(userColumn -> userColumn.setAutoWidth(true));  
 userGrid.asSingleSelect().addValueChangeListener(select -> {  
 editUser(select.getValue());  
 });  
 userGrid.addThemeVariants(GridVariant.*LUMO\_NO\_BORDER*, GridVariant.*LUMO\_NO\_ROW\_BORDERS*, GridVariant.*LUMO\_ROW\_STRIPES*);  
 }  
  
 private void configureLayout() {  
 userInfoAndAboutLayout.add(userForm, employeeForm);  
 userInfoAndAboutLayout.setVisible(false);  
 }  
  
 private void editUser(User user) {  
 if(user == null) {  
 closeEditor();  
 } else {  
 userForm.setUser(user);  
 employeeForm.setEmployee(user.getEmployee());  
 userInfoAndAboutLayout.setVisible(true);  
 addClassName("user-editor");  
 }  
 }  
  
 private void closeEditor() {  
 userForm.setUser(null);  
 employeeForm.setEmployee(null);  
 userInfoAndAboutLayout.setVisible(false);  
 removeClassName("user-editor");  
 }  
  
 private HorizontalLayout getToolBar() {  
 filter.setPlaceholder("Введите фильтр");  
 filter.setClearButtonVisible(true);  
 filter.setValueChangeMode(ValueChangeMode.*LAZY*);  
 filter.addValueChangeListener(change -> updateUserGrid());  
  
 HorizontalLayout toolBar = new HorizontalLayout(filter);  
 toolBar.addClassName("tool-bar");  
 return toolBar;  
 }  
  
 private void updateUserGrid() {  
 userGrid.setItems(userService.findAll(filter.getValue()));  
 }  
  
 private void saveUser(ChangeUserForm.SaveUserEvent saveUserEvent) {  
 userService.save(saveUserEvent.getUser());  
 updateUserGrid();  
 closeEditor();  
 }  
  
 private void deleteUser(ChangeUserForm.DeleteUserEvent deleteUserEvent) {  
 userService.delete(deleteUserEvent.getUser());  
 updateUserGrid();  
 closeEditor();  
 }  
  
 private void deleteUserFromEmployee(User user) {  
 if(user.getEmployee() != null) {  
 user.getEmployee().setUser(null);  
 employeeService.save(user.getEmployee());  
 employeeService.delete(user.getEmployee());  
 user.setEmployee(null);  
 }  
 }  
  
}

## **7.1.23 Листинг ChangeUserForm.java**

package ru.pahanjes.beautysaloon.crm.UI.views.accounts.change;  
  
import com.vaadin.flow.component.Component;  
import com.vaadin.flow.component.ComponentEvent;  
import com.vaadin.flow.component.ComponentEventListener;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.button.ButtonVariant;  
import com.vaadin.flow.component.checkbox.Checkbox;  
import com.vaadin.flow.component.combobox.ComboBox;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.radiobutton.RadioButtonGroup;  
import com.vaadin.flow.component.textfield.PasswordField;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.server.VaadinSession;  
import com.vaadin.flow.shared.Registration;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Role;  
import ru.pahanjes.beautysaloon.crm.backend.entity.User;  
  
import java.util.Collections;  
import java.util.Set;  
  
  
public class ChangeUserForm extends VerticalLayout {  
  
 private TextField username = new TextField("Логин");  
 private PasswordField password = new PasswordField("Пароль");  
 private PasswordField confirmPassword = new PasswordField("Подтвердите пароль");  
 private ComboBox<Role> role = new ComboBox<>("Права");  
 private RadioButtonGroup<String> isActive = new RadioButtonGroup<>();  
 private Checkbox changePassword = new Checkbox();  
 private Button save = new Button("Сохранить");  
 private Button delete = new Button("Удалить");  
 private Button close = new Button("Отменить");  
 private User user;  
 private final Long currentUserId = VaadinSession.*getCurrent*().getAttribute(User.class).getId();  
  
 public ChangeUserForm(User user) {  
 this.user = user;  
 addClassName("user-form");  
 setDefaultHorizontalComponentAlignment(Alignment.*CENTER*);  
 configureRadioButton();  
 configureForm();  
 add(  
 createTopLayout(),  
 changePassword,  
 createCenterLayout(),  
 isActive,  
 createButtonsLayout()  
 );  
 }  
  
 public ChangeUserForm() {  
 addClassName("user-form");  
 setDefaultHorizontalComponentAlignment(Alignment.*CENTER*);  
 configureRadioButton();  
 add(  
 createTopLayout(),  
 changePassword,  
 createCenterLayout(),  
 isActive,  
 createButtonsLayout()  
 );  
 }  
  
 private void configureForm() {  
 username.setValue(user.getUsername());  
 password.setEnabled(false);  
 confirmPassword.setEnabled(false);  
 changePassword.setLabel("Сменить пароль");  
 changePassword.setValue(false);  
 changePassword.addValueChangeListener(change -> {  
 if (change.getValue()) {  
 password.setEnabled(true);  
 confirmPassword.setEnabled(true);  
 } else if (!change.getValue()) {  
 password.setEnabled(false);  
 confirmPassword.setEnabled(false);  
 }  
 });  
 role.setItems(Role.*values*());  
 role.setValue(user.getRole().iterator().next());  
 isActive.setValue(user.isActive() ? "Активен" : "Неактивен");  
 }  
  
 private void configureRadioButton() {  
 isActive.setItems("Активен", "Неактивен");  
 }  
  
 private Component createButtonsLayout() {  
 save.addClickListener(save -> {  
 updateUser();  
 fireEvent(new SaveUserEvent(this, user));  
 });  
 save.addThemeVariants(ButtonVariant.*LUMO\_PRIMARY*);  
 delete.addThemeVariants(ButtonVariant.*LUMO\_ERROR*);  
 delete.addClickListener(delete -> fireEvent(new DeleteUserEvent(this, user)));  
 close.addClickListener(close -> fireEvent(new CloseUserEvent(this)));  
 close.addThemeVariants(ButtonVariant.*LUMO\_TERTIARY*);  
  
 return new HorizontalLayout(save, delete, close);  
 }  
  
 private Component createTopLayout() {  
 return new HorizontalLayout(  
 username,  
 role  
 );  
 }  
  
 private Component createCenterLayout() {  
 HorizontalLayout horizontalLayout = new HorizontalLayout(password, confirmPassword);  
 horizontalLayout.setAlignItems(Alignment.*CENTER*);  
 return horizontalLayout;  
 }  
  
 private void updateUser() {  
 if(changePassword.getValue()) {  
 if(password.getValue().equals(confirmPassword.getValue())) {  
 user.setPassword(password.getValue());  
 } else {  
 Notification.*show*("Пароли не совпадают");  
 return;  
 }  
 }  
 user.setUsername(username.getValue());  
 user.setRole(Collections.*singleton*(role.getValue()));  
 user.setActive(isActive.getValue().equals("Активен"));  
 }  
  
 public User getUser() {  
 return user;  
 }  
  
 public void setUser(User user) {  
 this.user = user;  
 if(user != null) {  
 configureForm();  
 delete.setEnabled(!user.getId().equals(currentUserId));  
 }  
 }  
  
 public String getUsername() {  
 return username.getValue();  
 }  
  
 public Set<Role> getRole() {  
 return Collections.*singleton*(role.getValue());  
 }  
  
 public boolean isChangePassword() {  
 return changePassword.getValue();  
 }  
  
 public String getPassword() {  
 return password.getValue();  
 }  
  
 public String getConfirmPassword() {  
 return confirmPassword.getValue();  
 }  
  
 public boolean isActive() {  
 return isActive.getValue().equals("Активен");  
 }  
  
 public static abstract class ChangeUserFormEvent extends ComponentEvent<ChangeUserForm> {  
 private User user;  
  
 protected ChangeUserFormEvent(ChangeUserForm source, User user) {  
 super(source, false);  
 this.user = user;  
 }  
  
 public User getUser() {  
 return user;  
 }  
 }  
  
 public static class SaveUserEvent extends ChangeUserFormEvent {  
 public SaveUserEvent(ChangeUserForm source, User user) {  
 super(source, user);  
 }  
 }  
  
 public static class DeleteUserEvent extends ChangeUserFormEvent {  
 public DeleteUserEvent(ChangeUserForm source, User user) {  
 super(source, user);  
 }  
 }  
  
 public static class CloseUserEvent extends ChangeUserFormEvent {  
 public CloseUserEvent(ChangeUserForm source) {  
 super(source, null);  
 }  
 }  
  
 public <T extends ComponentEvent<?>> Registration addListener(Class<T> eventType,  
 ComponentEventListener<T> listener) {  
 return getEventBus().addListener(eventType, listener);  
 }  
}

## **7.1.24 Листинг CabinetApp.java**

package ru.pahanjes.beautysaloon.crm.UI.views.cabinet;  
  
import com.vaadin.flow.component.Component;  
import com.vaadin.flow.component.applayout.AppLayout;  
import com.vaadin.flow.component.applayout.DrawerToggle;  
import com.vaadin.flow.component.dependency.CssImport;  
import com.vaadin.flow.component.dependency.JsModule;  
import com.vaadin.flow.component.html.Anchor;  
import com.vaadin.flow.component.html.H1;  
import com.vaadin.flow.component.html.H5;  
import com.vaadin.flow.component.html.Image;  
import com.vaadin.flow.component.orderedlayout.FlexComponent;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.tabs.Tab;  
import com.vaadin.flow.component.tabs.Tabs;  
import com.vaadin.flow.component.tabs.TabsVariant;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.router.RouterLink;  
import com.vaadin.flow.server.VaadinSession;  
import ru.pahanjes.beautysaloon.crm.backend.entity.User;  
import ru.pahanjes.beautysaloon.crm.security.AuthService;  
  
@Route(value = "lk")  
@JsModule("./styles/shared-styles.js")  
@CssImport("./styles/views/cabinet-view.css")  
public class CabinetApp extends AppLayout {  
  
 private final Tabs menu;  
 private final AuthService authService;  
  
 public CabinetApp(AuthService authService) {  
 this.authService = authService;  
 setPrimarySection(Section.*DRAWER*);  
 menu = createMenu();  
 createHeader();  
 createDrawBarContent();  
 }  
  
 private Tabs createMenu() {  
 final Tabs tabs = new Tabs();  
 tabs.setOrientation(Tabs.Orientation.*VERTICAL*);  
 tabs.addThemeVariants(TabsVariant.*LUMO\_MINIMAL*);  
 tabs.setId("tabs");  
 tabs.add(createMenuItems());  
 return tabs;  
 }  
  
 private Component[] createMenuItems() {  
 User user = VaadinSession.*getCurrent*().getAttribute(User.class);  
 return authService.getAuthorizedRoutes(user.getRole()).stream()  
 .map(routes -> createTab(routes.getName(), routes.getView())).toArray(Component[]::new);  
 }  
  
 private Tab createTab(String text, Class<? extends Component> navigationTarget) {  
 final Tab tab = new Tab();  
 tab.add(new RouterLink(text, navigationTarget));  
 return tab;  
 }  
  
 private void createDrawBarContent() {  
 VerticalLayout menuLayout = new VerticalLayout();  
 menuLayout.setSizeFull();  
 menuLayout.setPadding(false);  
 menuLayout.setSpacing(false);  
 menuLayout.getThemeList().set("spacing-s", true);  
 menuLayout.setAlignItems(FlexComponent.Alignment.*STRETCH*);  
 HorizontalLayout logoLayout = new HorizontalLayout();  
 logoLayout.setId("logo");  
 logoLayout.setAlignItems(FlexComponent.Alignment.*CENTER*);  
 logoLayout.add(new Image("images/logo.png", "Beauty Saloon CRM"));  
 logoLayout.add(new H1("CRM"));  
 menuLayout.add(logoLayout, menu);  
 addToDrawer(menuLayout);  
 setDrawerOpened(false);  
 }  
  
 private void createHeader() {  
 H1 logo = new H1("Личный кабинет");  
 logo.setId("logo");  
 HorizontalLayout header = new HorizontalLayout();  
 header.setId("header");  
 header.getThemeList().set("dark", true);  
 header.setWidthFull();  
 header.setSpacing(false);  
 header.setAlignItems(FlexComponent.Alignment.*CENTER*);  
 header.add(new DrawerToggle());  
 Anchor logout = new Anchor("logout", "Выход");  
 logout.setId("anchor");  
 header.add(  
 logo,  
 new H5(VaadinSession.*getCurrent*().getAttribute(User.class).getUsername()),  
 logout);  
 header.expand(logo);  
 addToNavbar(header);  
 }  
  
}

## **7.1.25 Листинг WelcomeLayout.java**

package ru.pahanjes.beautysaloon.crm.UI.views.cabinet;  
  
import com.vaadin.flow.component.Component;  
import com.vaadin.flow.component.html.Div;  
import com.vaadin.flow.component.html.H2;  
import com.vaadin.flow.component.html.Paragraph;  
import com.vaadin.flow.component.html.Span;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.server.VaadinSession;  
import ru.pahanjes.beautysaloon.crm.backend.entity.User;  
  
import java.time.LocalDateTime;  
  
@Route(value = "lk/welcome")  
@PageTitle("Добро пожаловать | BS CRM")  
public class WelcomeLayout extends VerticalLayout {  
  
 private static class NewsBlock extends Div {  
 private final String datetime;  
 private final String news;  
  
 public NewsBlock(String datetime, String news) {  
 this.datetime = datetime;  
 this.news = news;  
 getStyle().set("border", " black");  
 add(getNewsBlock());  
 }  
  
 private Component getNewsBlock() {  
 VerticalLayout layout = new VerticalLayout();  
 layout.add(new Span(datetime));  
 layout.add(new Paragraph(news));  
  
 return layout;  
 }  
 }  
  
 public WelcomeLayout() {  
 add(  
 createHeaderLayout(),  
 createNewsLayout()  
 );  
 }  
  
 private Component createHeaderLayout(){  
 H2 header;  
 if(VaadinSession.*getCurrent*().getAttribute(User.class).getEmployee() == null) {  
 header = new H2("Добро пожаловать");  
 } else {  
 header = new H2("Добро пожаловать, " + VaadinSession.*getCurrent*().getAttribute(User.class).getEmployee().getFirstName());  
 }  
 HorizontalLayout headerLayout = new HorizontalLayout(header);  
 headerLayout.setWidthFull();  
 headerLayout.setJustifyContentMode(JustifyContentMode.*CENTER*);  
 return headerLayout;  
 }  
  
 private Component createNewsLayout() {  
 VerticalLayout newsLayout = new VerticalLayout();  
 newsLayout.setSizeFull();  
 newsLayout.setJustifyContentMode(JustifyContentMode.*CENTER*);  
 HorizontalLayout newsBlock = new HorizontalLayout();  
 newsBlock.setWidthFull();  
 newsBlock.setJustifyContentMode(JustifyContentMode.*CENTER*);  
 newsBlock.add(  
 new NewsBlock(LocalDateTime.*now*().getYear() + "/" + LocalDateTime.*now*().getMonthValue() + "/" + LocalDateTime.*now*().getDayOfMonth(), "Нашей компанией было достигнуто важное деловое соглашение."),  
 new NewsBlock(LocalDateTime.*now*().getYear() + "/" + LocalDateTime.*now*().getMonthValue() + "/" + LocalDateTime.*now*().getDayOfMonth(), "Что-то там было сделано, где-то что-то было подписано.")  
 );  
 newsLayout.add(  
 newsBlock  
 );  
  
 return newsLayout;  
 }  
}

## **7.1.26 Листинг DashBoardView.java**

package ru.pahanjes.beautysaloon.crm.UI.views.dashboard;  
  
import com.vaadin.flow.component.grid.Grid;  
import com.vaadin.flow.component.grid.GridVariant;  
import com.vaadin.flow.component.html.Span;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.server.VaadinSession;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Customer;  
import ru.pahanjes.beautysaloon.crm.backend.entity.User;  
import ru.pahanjes.beautysaloon.crm.backend.service.CustomerService;  
import ru.pahanjes.beautysaloon.crm.backend.service.EmployeeService;  
  
import java.util.ArrayList;  
import java.util.List;  
  
@Route(value = "lk/dashboard")  
@PageTitle("Расписание | BS")  
public class DashBoardView extends VerticalLayout {  
  
 private final CustomerService customerService;  
 private final EmployeeService employeeService;  
 private Long employeeId = VaadinSession.*getCurrent*().getAttribute(User.class).getEmployee().getId();  
 private List<Customer> customerList;  
 private Grid<Customer> timetable;  
  
 public DashBoardView(CustomerService customerService, EmployeeService employeeService){  
 this.customerService = customerService;  
 this.employeeService = employeeService;  
 setSizeFull();  
 addClassName("dashboard-view");  
 timetable = new Grid<>(Customer.class);  
 setDefaultHorizontalComponentAlignment(Alignment.*CENTER*);  
 configureList();  
 configureGrid();  
 updateTimetable();  
 add(  
 getCustomerStats(),  
 timetable  
 );  
 }  
  
 private void configureList() {  
 customerList = new ArrayList<>(employeeService.findById(employeeId).getCustomers());  
 }  
  
 private void configureGrid() {  
 timetable.addClassName("timetable");  
 timetable.setSizeFull();  
 timetable.removeAllColumns();  
 timetable.addColumn(customer -> customer.getTimetable() == null ? "Не установлена" : customer.getTimetable().getDayOfMonth() + "-" + customer.getTimetable().getMonthValue() + "-" + customer.getTimetable().getYear()).setHeader("Дата").setSortable(true);  
 timetable.addColumn(customer -> customer.getTimetable() == null ? "Не установлено" : customer.getTimetable().getHour() + ":" + customer.getTimetable().getMinute()).setHeader("Время").setSortable(true);  
 timetable.addColumn(customer -> customer.getLastName() == null ? "-" : customer.getLastName()).setHeader("Фамилия").setSortable(true);  
 timetable.addColumn(customer -> customer.getFirstName() == null ? "-" : customer.getFirstName()).setHeader("Имя").setSortable(true);  
 timetable.addColumn(customer -> customer.getStatus() == null ? "-" : customer.getStatus().getValue()).setHeader("Статус").setSortable(true);  
 timetable.addColumn(customer -> customer.getPhoneNumber() == null ? "-" : customer.getPhoneNumber()).setHeader("Номер телефона").setSortable(true);  
 timetable.getColumns().forEach(customerColumn -> customerColumn.setAutoWidth(true));  
 timetable.addThemeVariants(GridVariant.*LUMO\_NO\_BORDER*, GridVariant.*LUMO\_NO\_ROW\_BORDERS*, GridVariant.*LUMO\_ROW\_STRIPES*);  
 }  
  
 private Span getCustomerStats() {  
 Span stats;  
 if(employeeId != null) {  
 stats = new Span("Клиентов: " + employeeService.findById(employeeId).getCustomers().size());  
 } else {  
 stats = new Span("Клиентов: 0");  
 }  
 stats.addClassName("customer-stats");  
 return stats;  
 }  
  
 private void updateTimetable(){  
 if(customerList != null) {  
 timetable.setItems(customerList);  
 }  
 }  
}

## **7.1.27 Листинг CustomerForm.java**

package ru.pahanjes.beautysaloon.crm.UI.views.list;  
  
import com.vaadin.componentfactory.multiselect.MultiComboBox;  
import com.vaadin.flow.component.Component;  
import com.vaadin.flow.component.ComponentEvent;  
import com.vaadin.flow.component.ComponentEventListener;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.button.ButtonVariant;  
import com.vaadin.flow.component.combobox.ComboBox;  
import com.vaadin.flow.component.datepicker.DatePicker;  
import com.vaadin.flow.component.datetimepicker.DateTimePicker;  
import com.vaadin.flow.component.formlayout.FormLayout;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.textfield.EmailField;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.data.binder.BeanValidationBinder;  
import com.vaadin.flow.data.binder.Binder;  
import com.vaadin.flow.data.validator.EmailValidator;  
import com.vaadin.flow.server.VaadinSession;  
import com.vaadin.flow.shared.Registration;  
import ru.pahanjes.beautysaloon.crm.backend.entity.\*;  
import ru.pahanjes.beautysaloon.crm.backend.repository.ServiceRepository;  
import ru.pahanjes.beautysaloon.crm.backend.service.EmployeeService;  
import ru.pahanjes.beautysaloon.crm.util.Constants;  
  
import java.time.DayOfWeek;  
import java.time.LocalDateTime;  
import java.util.Arrays;  
import java.util.List;  
import java.util.Objects;  
import java.util.Optional;  
  
public class CustomerForm extends FormLayout {  
 private final List<Employee> employees;  
 private final ServiceRepository serviceRepository;  
 private final EmployeeService employeeService;  
 private final TextField firstName = new TextField("Имя");  
 private final TextField lastName = new TextField("Фамилия");  
 private final EmailField email = new EmailField("Электронная почта");  
 private final TextField phoneNumber = new TextField("Номер телефона");  
 private final ComboBox<Customer.Status> status = new ComboBox<>("Статус");  
 private DateTimePicker timetable = new DateTimePicker();  
 private final MultiComboBox<Service> services = new MultiComboBox<>("Услуги");  
 private final ComboBox<Employee> employee = new ComboBox<>("Сотрудник");  
  
 private final Button save = new Button("Сохранить");  
 private final Button delete = new Button("Удалить");  
 private final Button close = new Button("Отменить");  
  
 Binder<Customer> binder = new BeanValidationBinder<>(Customer.class);  
  
 public CustomerForm(List<Employee> employees, ServiceRepository serviceRepository, EmployeeService employeeService){  
 this.employees = employees;  
 this.serviceRepository = serviceRepository;  
 this.employeeService = employeeService;  
 addClassName("customer-form");  
  
 configureBinder();  
 status.setItems(Customer.Status.*values*());  
 status.setItemLabelGenerator(Customer.Status::getValue);  
 timetable.setLabel("Дата и время приема");  
 timetable.setMin(Constants.*COMPANY\_FOUNDATION\_DATE*);  
 timetable.setMax(LocalDateTime.*now*().plusMonths(2));  
 timetable.setRequiredIndicatorVisible(true);  
 services.setItems(serviceRepository.findAll());  
 services.setItemLabelGenerator(Service::getService);  
 employee.setItems(employees);  
 employee.setItemLabelGenerator(Employee::getFullNameWithPosition);  
 add(  
 firstName,  
 lastName,  
 phoneNumber,  
 email,  
 status,  
 timetable,  
 services,  
 employee,  
 createButtonsLayout()  
 );  
 }  
  
 private void configureBinder() {  
 binder.forField(firstName)  
 .withValidator(  
 fname -> !fname.isEmpty(), "Введите имя")  
 .bind(Customer::getFirstName, Customer::setFirstName);  
 binder.forField(lastName)  
 .withValidator(  
 lname -> !lname.isEmpty(), "Введите фамилию")  
 .bind(Customer::getLastName, Customer::setLastName);  
 binder.forField(email)  
 .withValidator(  
 new EmailValidator("Введите корректный адрес электронной почты"))  
 .bind(Customer::getEmail, Customer::setEmail);  
 binder.forField(phoneNumber)  
 .withValidator(  
 phone -> !phone.isEmpty(), "Введите номер телефона")  
 .bind(Customer::getPhoneNumber, Customer::setPhoneNumber);  
 binder.forField(status)  
 .withValidator(  
 Objects::*nonNull*, "Выберите статус")  
 .bind(Customer::getStatus, Customer::setStatus);  
 binder.forField(timetable)  
 .withValidator(  
 Objects::*nonNull*, "Выберите дату и время"  
 )  
 .withValidator(time -> !DayOfWeek.*SUNDAY*.equals(time.getDayOfWeek())  
 && time.getHour() >= 8 && time.getHour() <= 21,  
 "Запись может быть произведена только с понедельника по субботу с 8:00 до 21:00")  
 .bind(Customer::getTimetable, Customer::setTimetable);  
 binder.forField(services)  
 .withValidator(  
 services1 -> services1.size() > 0, "Выберите услуги")  
 .bind(Customer::getServices, Customer::setServices);  
 binder.forField(employee)  
 .withValidator(  
 Objects::*nonNull*, "Выберите работника")  
 .bind(Customer::getEmployee, Customer::setEmployee);  
 firstName.setRequired(true);  
 firstName.setRequiredIndicatorVisible(true);  
 firstName.setPlaceholder("Введите имя: ");  
  
 lastName.setRequired(true);  
 lastName.setRequiredIndicatorVisible(true);  
 lastName.setPlaceholder("Введите фамилию: ");  
  
 phoneNumber.setRequired(true);  
 phoneNumber.setRequiredIndicatorVisible(true);  
 phoneNumber.setPlaceholder("Введите номер телефона: ");  
  
 email.setRequiredIndicatorVisible(true);  
 email.setPlaceholder("Введите адрес электронной почты: ");  
  
 status.setRequired(true);  
 status.setRequiredIndicatorVisible(true);  
 status.setPlaceholder("Выберите статус: ");  
  
 timetable.setRequiredIndicatorVisible(true);  
 timetable.setDatePickerI18n(new DatePicker.DatePickerI18n()  
 .setWeek("Неделя").setCalendar("Календарь").setClear("Очистить")  
 .setToday("Сегодня").setCancel("Отмена").setFirstDayOfWeek(1)  
 .setMonthNames(Arrays.*asList*(  
 "Январь", "Февраль", "Март", "Апрель",  
 "Май", "Июнь", "Июль", "Август", "Сентябрь",  
 "Октябрь", "Ноябрь", "Декабрь"))  
 .setWeekdays(Arrays.*asList*(  
 "Воскресенье", "Понедельник", "Вторник", "Среда",  
 "Четверг", "Пятница", "Суббота"))  
 .setWeekdaysShort(Arrays.*asList*(  
 "Вс", "Пн", "Вт", "Ср", "Чт", "Пт", "Сб"))  
 );  
  
 services.setRequired(true);  
 services.setRequiredIndicatorVisible(true);  
 services.setPlaceholder("Выберите услуги: ");  
 services.setI18n(new MultiComboBox.MultiComboBoxI18n()  
 .setClear("Очистить")  
 .setSelect("Выбрать все"));  
  
 employee.setRequired(true);  
 employee.setRequiredIndicatorVisible(true);  
 employee.setPlaceholder("Выберите сотрудника: ");  
 }  
  
 public void setCustomer(Customer customer){  
 binder.setBean(customer);  
 if(VaadinSession.*getCurrent*().getAttribute(User.class).getRole().iterator().next() == Role.*USER*) {  
 employee.setReadOnly(true);  
 delete.setEnabled(false);  
 timetable.setReadOnly(false);  
 if(customer != null) {  
 if(customer.getEmployee() == null) {  
 employee.setReadOnly(false);  
 }  
 }  
 if(customer != null && customer.getEmployee() != null) {  
 if(!VaadinSession.*getCurrent*().getAttribute(User.class).getEmployee().getFullNameWithPosition().equals(customer.getEmployee().getFullNameWithPosition())) {  
 timetable.setReadOnly(true);  
 }  
 }  
 }  
 if(customer != null) {  
 if (customer.getTimetable() == null) {  
 timetable.clear();  
 }  
 }  
 }  
  
 private Component createButtonsLayout() {  
 save.addThemeVariants(ButtonVariant.*LUMO\_PRIMARY*);  
 delete.addThemeVariants(ButtonVariant.*LUMO\_ERROR*);  
 close.addThemeVariants(ButtonVariant.*LUMO\_TERTIARY*);  
  
 save.addClickListener(click -> validateAndSave());  
 delete.addClickListener(delete -> fireEvent(new DeleteEvent(this, binder.getBean())));  
 close.addClickListener(close -> fireEvent(new CloseEvent(this)));  
  
 binder.addStatusChangeListener(event -> save.setEnabled(binder.isValid()));  
  
 return new HorizontalLayout(save, delete, close);  
 }  
  
 private void validateAndSave() {  
 if(timetable.isEmpty()) {  
 Notification.*show*("Установите дату приема");  
 return;  
 }  
 Optional<Service> notActiveService = binder.getBean().getServices().stream().filter(service -> !service.isActive()).findFirst();  
 if (notActiveService.isPresent()) {  
 Notification.*show*("Услуга \"" + notActiveService.get().getService() + "\" в данный момент недоступна");  
 return;  
 }  
 if(binder.isValid()){  
 Employee employee = employeeService.findById(this.employee.getValue().getId());  
 if(employee.getCustomers().stream().anyMatch(  
 customer -> customer  
 .getFirstName().equals(binder.getBean().getFirstName())  
 && customer.getLastName().equals(binder.getBean().getLastName())  
 && customer.getPhoneNumber().equals(binder.getBean().getPhoneNumber())  
 )  
 ) {  
 fireEvent(new SaveEvent(this, binder.getBean(), false));  
 } else {  
 fireEvent(new SaveEvent(this, binder.getBean(), true));  
 }  
 }  
 }  
  
 public static abstract class CustomerFormEvent extends ComponentEvent<CustomerForm> {  
 private Customer customer;  
 private boolean isNewCustomer;  
  
 protected CustomerFormEvent(CustomerForm source, Customer customer) {  
 super(source, false);  
 this.customer = customer;  
 }  
  
 protected CustomerFormEvent(CustomerForm source, Customer customer, boolean isNewCustomer) {  
 super(source, false);  
 this.customer = customer;  
 this.isNewCustomer = isNewCustomer;  
 }  
  
 public Customer getCustomer(){  
 return customer;  
 }  
  
 public boolean isNewCustomer() {  
 return this.isNewCustomer;  
 }  
 }  
  
 public static class SaveEvent extends CustomerFormEvent {  
 public SaveEvent(CustomerForm source, Customer customer, boolean isNewCustomer) {  
 super(source, customer, isNewCustomer);  
 }  
 }  
  
 public static class DeleteEvent extends CustomerFormEvent {  
 public DeleteEvent(CustomerForm source, Customer customer) {  
 super(source, customer);  
 }  
 }  
  
 public static class CloseEvent extends CustomerFormEvent {  
 public CloseEvent(CustomerForm source) {  
 super(source, null);  
 }  
 }  
  
 public <T extends ComponentEvent<?>> Registration addListener (Class<T> eventType,  
 ComponentEventListener<T> listener) {  
 return getEventBus().addListener(eventType, listener);  
 }  
}

## **7.1.28 Листинг CustomerListView.java**

package ru.pahanjes.beautysaloon.crm.UI.views.list;  
  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.dependency.CssImport;  
import com.vaadin.flow.component.grid.Grid;  
import com.vaadin.flow.component.grid.GridVariant;  
import com.vaadin.flow.component.html.Div;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.data.value.ValueChangeMode;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Customer;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Employee;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Service;  
import ru.pahanjes.beautysaloon.crm.backend.repository.ServiceRepository;  
import ru.pahanjes.beautysaloon.crm.backend.service.CustomerService;  
import ru.pahanjes.beautysaloon.crm.backend.service.EmployeeService;  
import ru.pahanjes.beautysaloon.crm.backend.service.UserService;  
  
import java.util.Set;  
  
@Route(value = "lk/customer")  
@PageTitle("Клиенты | BS CRM")  
@CssImport("./styles/views/customer-view.css")  
public class CustomerListView extends VerticalLayout {  
  
 private Grid<Customer> customerGrid = new Grid<>(Customer.class);  
 private Grid<Employee> employeeGrid = new Grid<>(Employee.class);  
 private TextField filter = new TextField();  
 private CustomerService customerService;  
 private EmployeeService employeeService;  
 private ServiceRepository serviceRepository;  
 private UserService userService;  
 private CustomerForm customerForm;  
  
 public CustomerListView(CustomerService customerService, EmployeeService employeeService, ServiceRepository serviceRepository, UserService userService) {  
 this.customerService = customerService;  
 this.employeeService = employeeService;  
 this.serviceRepository = serviceRepository;  
 this.userService = userService;  
 addClassName("customer-list-view");  
 setSizeFull();  
 configureGrids();  
  
 customerForm = new CustomerForm(employeeService.findAll(), serviceRepository, employeeService);  
 customerForm.addListener(CustomerForm.SaveEvent.class, this::saveCustomer);  
 customerForm.addListener(CustomerForm.DeleteEvent.class, this::deleteCustomer);  
 customerForm.addListener(CustomerForm.CloseEvent.class, closeEvent -> closeCustomerEditor());  
  
 Div content = new Div(customerGrid, customerForm);  
 content.addClassName("customer-content");  
 content.setSizeFull();  
  
 add(getToolBar(), content);  
 updateCustomerList();  
 closeCustomerEditor();  
 }  
  
 private HorizontalLayout getToolBar() {  
 filter.setPlaceholder("Введите фильтр...");  
 filter.setClearButtonVisible(true);  
 filter.setValueChangeMode(ValueChangeMode.*LAZY*);  
 filter.addValueChangeListener(event -> updateCustomerList());  
  
 Button addCustomerButton = new Button("Добавить клиента", click -> addCustomer());  
 HorizontalLayout toolbar = new HorizontalLayout(filter, addCustomerButton);  
 toolbar.addClassName("toolbar");  
 return toolbar;  
 }  
  
 private void addCustomer() {  
 customerGrid.asSingleSelect().clear();  
 editCustomer(new Customer());  
 }  
  
 private void configureGrids() {  
 customerGrid.addClassName("customer-grid");  
 customerGrid.setSizeFull();  
 customerGrid.removeAllColumns();  
 customerGrid.addColumn(customer -> customer.getLastName() == null ? "-" : customer.getLastName()).setHeader("Фамилия").setSortable(true);  
 customerGrid.addColumn(customer -> customer.getFirstName() == null ? "-" : customer.getFirstName()).setHeader("Имя").setSortable(true);  
 customerGrid.addColumn(customer -> customer.getPhoneNumber() == null ? "-" : customer.getPhoneNumber()).setHeader("Номер телефона").setSortable(true);  
 customerGrid.addColumn(customer -> customer.getEmail() == null ? "-" : customer.getEmail()).setHeader("email").setSortable(true);  
 customerGrid.addColumn(customer -> customer.getServices().size() == 0 ? "-" : *setToString*(customer.getServices())).setHeader("Услуги").setSortable(false);  
 customerGrid.addColumn(customer -> customer.getStatus() == null ? "-" : customer.getStatus().getValue()).setHeader("Статус").setSortable(true);  
 customerGrid.getColumns().forEach(customerColumn -> customerColumn.setAutoWidth(true));  
 customerGrid.asSingleSelect().addValueChangeListener(event -> editCustomer(event.getValue()));  
 customerGrid.addThemeVariants(GridVariant.*LUMO\_NO\_BORDER*, GridVariant.*LUMO\_NO\_ROW\_BORDERS*, GridVariant.*LUMO\_ROW\_STRIPES*);  
 }  
  
 private static String setToString(Set<Service> serviceSet) {  
 StringBuilder result = new StringBuilder();  
 for(Service service : serviceSet) {  
 result.append(service.getService()).append(", ");  
 }  
 result.delete(result.length()-2, result.length()-1);  
 return result.toString();  
 }  
  
 private void saveCustomer(CustomerForm.SaveEvent saveEvent){  
 Employee employee = employeeService.findById(saveEvent.getCustomer().getEmployee().getId());  
 employee.addCustomer(saveEvent.getCustomer());  
 employeeService.save(employee);  
 saveEvent.getCustomer().setEmployee(employeeService.findById(employee.getId()));  
 customerService.save(saveEvent.getCustomer());  
  
 updateCustomerList();  
 closeCustomerEditor();  
 }  
  
 private void deleteCustomer(CustomerForm.DeleteEvent deleteEvent){  
 customerService.delete(deleteEvent.getCustomer());  
 updateCustomerList();  
 closeCustomerEditor();  
 }  
  
 private void editCustomer(Customer customer){  
 if(customer == null){  
 closeCustomerEditor();  
 } else {  
 customerForm.setCustomer(customer);  
 customerForm.setVisible(true);  
 addClassName("customer-editing");  
 }  
 }  
  
 private void closeCustomerEditor() {  
 customerForm.setCustomer(null);  
 customerForm.setVisible(false);  
 removeClassName("customer-editing");  
 }  
  
 private void updateCustomerList(){  
 customerGrid.setItems(customerService.findAll(filter.getValue()));  
 }  
  
}

## **7.1.29 Листинг EmployeeForm.java**

package ru.pahanjes.beautysaloon.crm.UI.views.list;  
  
import com.vaadin.componentfactory.multiselect.MultiComboBox;  
import com.vaadin.flow.component.Component;  
import com.vaadin.flow.component.ComponentEvent;  
import com.vaadin.flow.component.ComponentEventListener;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.button.ButtonVariant;  
import com.vaadin.flow.component.combobox.ComboBox;  
import com.vaadin.flow.component.formlayout.FormLayout;  
import com.vaadin.flow.component.icon.Icon;  
import com.vaadin.flow.component.icon.VaadinIcon;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.textfield.\*;  
import com.vaadin.flow.data.binder.BeanValidationBinder;  
import com.vaadin.flow.data.binder.Binder;  
import com.vaadin.flow.data.validator.EmailValidator;  
import com.vaadin.flow.server.VaadinSession;  
import com.vaadin.flow.shared.Registration;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Employee;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Role;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Service;  
import ru.pahanjes.beautysaloon.crm.backend.entity.User;  
import ru.pahanjes.beautysaloon.crm.backend.repository.EmployeeRepository;  
import ru.pahanjes.beautysaloon.crm.backend.repository.ServiceRepository;  
import ru.pahanjes.beautysaloon.crm.backend.repository.UserRepository;  
import ru.pahanjes.beautysaloon.crm.backend.service.UserService;  
import ru.pahanjes.beautysaloon.crm.security.AuthService;  
  
import java.math.BigDecimal;  
import java.util.List;  
import java.util.Objects;  
  
public class EmployeeForm extends FormLayout {  
  
 private final List<Employee> employees;  
 private final ServiceRepository serviceRepository;  
 private UserRepository userRepository;  
 private EmployeeRepository employeeRepository;  
 private AuthService authService;  
 private UserService userService;  
 TextField firstName = new TextField("Имя");  
 TextField lastName = new TextField("Фамилия");  
 TextField position = new TextField("Должность");  
 BigDecimalField salary = new BigDecimalField("Зарплата");  
 EmailField email = new EmailField("Электронная почта");  
 TextField phoneNumber = new TextField("Номер телефона");  
 ComboBox<Employee.Status> status = new ComboBox<>("Статус");  
 MultiComboBox<Service> services = new MultiComboBox<>("Оказываемые услуги");  
 TextField login = new TextField("Логин");  
 PasswordField password = new PasswordField("Пароль");  
 ComboBox<Role> role = new ComboBox<Role>("Права");  
  
 Button save = new Button("Сохранить");  
 Button delete = new Button("Удалить");  
 Button close = new Button("Отменить");  
  
 Binder<Employee> binder = new BeanValidationBinder<>(Employee.class);  
  
 public EmployeeForm(List<Employee> employees, ServiceRepository serviceRepository, UserRepository userRepository, EmployeeRepository employeeRepository, AuthService authService){  
 this.employees = employees;  
 this.serviceRepository = serviceRepository;  
 this.userRepository = userRepository;  
 this.employeeRepository = employeeRepository;  
 this.authService = authService;  
 this.userService = userService;  
 addClassName("employee-form");  
  
 salary.addThemeVariants(TextFieldVariant.*LUMO\_ALIGN\_RIGHT*);  
 salary.setPrefixComponent(new Icon(VaadinIcon.*MONEY*));  
 salary.setValue(new BigDecimal(0));  
 services.setItems(serviceRepository.findAll());  
 services.setItemLabelGenerator(Service::getService);  
 status.setItems(Employee.Status.*values*());  
 status.setItemLabelGenerator(Employee.Status::getValue);  
 add(  
 firstName,  
 lastName,  
 position,  
 salary,  
 phoneNumber,  
 email,  
 status,  
 services,  
 login,  
 password,  
 role,  
 createButtonsLayout()  
 );  
 }  
  
 private void configureBinder() {  
 binder.forField(firstName)  
 .withValidator(  
 fname -> !fname.isEmpty(), "Введите имя")  
 .bind(Employee::getFirstName, Employee::setFirstName);  
 binder.forField(lastName)  
 .withValidator(  
 lname -> !lname.isEmpty(), "Введите фамилию")  
 .bind(Employee::getLastName, Employee::setLastName);  
 binder.forField(position)  
 .withValidator(  
 pos -> !pos.isEmpty(), "Введите должность")  
 .bind(Employee::getPosition, Employee::setPosition);  
 binder.bind(salary, Employee::getSalary, Employee::setSalary);  
 binder.forField(email)  
 .withValidator(  
 new EmailValidator("Введите корректный адрес электронной почты"))  
 .bind(Employee::getEmail, Employee::setEmail);  
 binder.forField(phoneNumber)  
 .withValidator(  
 phone -> !phone.isEmpty(), "Введите номер телефона")  
 .bind(Employee::getPhoneNumber, Employee::setPhoneNumber);  
 binder.forField(status)  
 .withValidator(  
 Objects::*nonNull*, "Выберите статус")  
 .bind(Employee::getStatus, Employee::setStatus);  
 binder.forField(services)  
 .withValidator(  
 services1 -> services1.size() > 0, "Выберите услуги")  
 .bind(Employee::getServices, Employee::setServices);  
 if(binder.getBean() != null) {  
 if (binder.getBean().getUser() == null) {  
 firstName.setPlaceholder("Введите имя");  
 firstName.setRequired(true);  
 firstName.setRequiredIndicatorVisible(true);  
  
 lastName.setPlaceholder("Введите фамилию");  
 lastName.setRequired(true);  
 lastName.setRequiredIndicatorVisible(true);  
  
 position.setPlaceholder("Введите олжность");  
 position.setRequired(true);  
 position.setRequiredIndicatorVisible(true);  
  
 salary.setRequiredIndicatorVisible(true);  
  
 phoneNumber.setPlaceholder("Введите номер телефона");  
 phoneNumber.setRequired(true);  
 phoneNumber.setRequiredIndicatorVisible(true);  
  
 email.setPlaceholder("Введите email");  
 email.setRequiredIndicatorVisible(true);  
  
 status.setPlaceholder("Выберие статус");  
 status.setRequired(true);  
 status.setRequiredIndicatorVisible(true);  
  
 services.setPlaceholder("Выберите услуги");  
 services.setRequired(true);  
 services.setRequiredIndicatorVisible(true);  
  
 login.setValue("");  
 login.setPlaceholder("Введите логин: ");  
 login.setRequiredIndicatorVisible(true);  
 login.setRequired(true);  
 login.setReadOnly(false);  
 password.setValue("");  
 password.setPlaceholder("Введите пароль");  
 password.setRequiredIndicatorVisible(true);  
 password.setRequired(true);  
 password.setEnabled(true);  
 role.setPlaceholder("Выберите права: ");  
 role.setItems(Role.*values*());  
 role.setRequired(true);  
 role.setRequiredIndicatorVisible(true);  
 role.setReadOnly(false);  
 } else {  
 firstName.setRequired(false);  
 firstName.setRequiredIndicatorVisible(false);  
  
 lastName.setRequired(false);  
 lastName.setRequiredIndicatorVisible(false);  
  
 position.setRequired(false);  
 position.setRequiredIndicatorVisible(false);  
  
 salary.setRequiredIndicatorVisible(false);  
  
 phoneNumber.setRequired(false);  
 phoneNumber.setRequiredIndicatorVisible(false);  
  
 email.setRequiredIndicatorVisible(false);  
  
 status.setRequired(false);  
 status.setRequiredIndicatorVisible(false);  
  
 services.setRequired(false);  
 services.setRequiredIndicatorVisible(false);  
  
 login.setValue(binder.getBean().getUser().getUsername());  
 login.setRequired(false);  
 login.setRequiredIndicatorVisible(false);  
 login.setReadOnly(true);  
 password.setEnabled(false);  
 password.setRequired(false);  
 password.setRequiredIndicatorVisible(false);  
 role.setItems(Role.*values*());  
 role.setValue(binder.getBean().getUser().getRole().iterator().next());  
 role.setRequired(false);  
 role.setRequiredIndicatorVisible(false);  
 role.setReadOnly(true);  
 }  
 }  
 }  
  
 public void setEmployee(Employee employee){  
 binder.setBean(employee);  
 configureBinder();  
 }  
  
 private Component createButtonsLayout(){  
 save.addThemeVariants(ButtonVariant.*LUMO\_PRIMARY*);  
 delete.addThemeVariants(ButtonVariant.*LUMO\_ERROR*);  
 close.addThemeVariants(ButtonVariant.*LUMO\_TERTIARY*);  
  
 save.addClickListener(click -> {  
 if(login.getValue().isEmpty()) {  
 Notification.*show*("Введите логин");  
 } else if (password.getValue().isEmpty()) {  
 Notification.*show*("Введите пароль");  
 } else if (status.getValue() == null) {  
 Notification.*show*("Выберите статус");  
 } else {  
 validateAndSave();  
 }  
 });  
 delete.addClickListener(delete -> {  
 if(!binder.getBean().getId().equals(VaadinSession.*getCurrent*().getAttribute(User.class).getEmployee().getId())) {  
 fireEvent(new DeleteEvent(this, binder.getBean()));  
 } else {  
 Notification.*show*("Нельзя удалить себя");  
 }  
 });  
 close.addClickListener(close -> fireEvent(new CloseEvent(this)));  
  
 binder.addStatusChangeListener(event -> save.setEnabled(binder.isValid()));  
  
 return new HorizontalLayout(save, delete, close);  
 }  
  
 private void validateAndSave() {  
  
 if(binder.isValid()){  
 if(binder.getBean().getUser() == null) {  
 if(authService.register(login.getValue(), password.getValue(), binder.getBean(), role.getValue()) == -1) {  
 Notification.*show*("Пользователь с таким именем уже существует");  
 return;  
 }  
 }  
 fireEvent(new SaveEvent(this, binder.getBean()));  
 }  
 }  
  
 public static abstract class EmployeeFormEvent extends ComponentEvent<EmployeeForm> {  
 private Employee employee;  
  
 protected EmployeeFormEvent(EmployeeForm source, Employee employee){  
 super(source, false);  
 this.employee = employee;  
 }  
  
 public Employee getEmployee(){  
 return employee;  
 }  
 }  
  
 public static class SaveEvent extends EmployeeFormEvent {  
 public SaveEvent(EmployeeForm source, Employee employee) {  
 super(source, employee);  
 }  
 }  
  
 public static class DeleteEvent extends EmployeeFormEvent {  
 public DeleteEvent(EmployeeForm source, Employee employee) {  
 super(source, employee);  
 }  
 }  
  
 public static class CloseEvent extends EmployeeFormEvent {  
 public CloseEvent(EmployeeForm source) {  
 super(source, null);  
 }  
 }  
  
 public <T extends ComponentEvent<?>> Registration addListener (Class<T> eventType,  
 ComponentEventListener<T> listener) {  
 return getEventBus().addListener(eventType, listener);  
 }  
}

## **7.1.30 Листинг EmployeeListView.java**

package ru.pahanjes.beautysaloon.crm.UI.views.list;  
  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.dependency.CssImport;  
import com.vaadin.flow.component.grid.Grid;  
import com.vaadin.flow.component.grid.GridVariant;  
import com.vaadin.flow.component.html.Div;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.data.value.ValueChangeMode;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Employee;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Service;  
import ru.pahanjes.beautysaloon.crm.backend.repository.EmployeeRepository;  
import ru.pahanjes.beautysaloon.crm.backend.repository.ServiceRepository;  
import ru.pahanjes.beautysaloon.crm.backend.repository.UserRepository;  
import ru.pahanjes.beautysaloon.crm.backend.service.CustomerService;  
import ru.pahanjes.beautysaloon.crm.backend.service.EmployeeService;  
import ru.pahanjes.beautysaloon.crm.backend.service.UserService;  
import ru.pahanjes.beautysaloon.crm.security.AuthService;  
  
import java.math.BigDecimal;  
import java.util.Set;  
  
@Route(value = "lk/employee")  
@PageTitle("Сотрудники | BS CRM")  
@CssImport("./styles/views/employee-view.css")  
public class EmployeeListView extends VerticalLayout {  
  
 private Grid<Employee> employeeGrid = new Grid<>(Employee.class);  
 TextField filter = new TextField();  
 private CustomerService customerService;  
 private EmployeeService employeeService;  
 private ServiceRepository serviceRepository;  
 private UserRepository userRepository;  
 private EmployeeRepository employeeRepository;  
 private UserService userService;  
 private AuthService authService;  
 private EmployeeForm employeeForm;  
  
 public EmployeeListView(CustomerService customerService, EmployeeService employeeService, ServiceRepository serviceRepository, UserRepository userRepository, EmployeeRepository employeeRepository, UserService userService, AuthService authService){  
 this.customerService = customerService;  
 this.employeeService = employeeService;  
 this.serviceRepository = serviceRepository;  
 this.userRepository = userRepository;  
 this.employeeRepository = employeeRepository;  
 this.userService = userService;  
 this.authService = authService;  
 addClassName("employee-list-view");  
 setSizeFull();  
 configureGrid();  
 getToolBar();  
  
 employeeForm = new EmployeeForm(employeeService.findAll(), serviceRepository, userRepository, employeeRepository, authService);  
 employeeForm.addListener(EmployeeForm.SaveEvent.class, this::saveEmployee);  
 employeeForm.addListener(EmployeeForm.DeleteEvent.class, this::deleteEmployee);  
 employeeForm.addListener(EmployeeForm.CloseEvent.class, closeEvent -> closeEmployeeEditor());  
  
 Div content = new Div(employeeGrid, employeeForm);  
 content.addClassName("employee-content");  
 content.setSizeFull();  
  
 add(getToolBar(), content);  
 updateEmployeeList();  
 closeEmployeeEditor();  
 }  
  
 private HorizontalLayout getToolBar() {  
 filter.setPlaceholder("Введите фильтр...");  
 filter.setClearButtonVisible(true);  
 filter.setValueChangeMode(ValueChangeMode.*LAZY*);  
 filter.addValueChangeListener(event -> updateEmployeeList());  
  
 Button addEmployeeButton = new Button("Добавить сотрудника", click -> addEmployee());  
 HorizontalLayout toolbar = new HorizontalLayout(filter, addEmployeeButton);  
 toolbar.addClassName("toolbar");  
 return toolbar;  
 }  
  
 private void addEmployee() {  
 employeeGrid.asSingleSelect().clear();  
 editEmployee(new Employee());  
 }  
  
 private void configureGrid() {  
 employeeGrid.addClassName("employee-grid");  
 employeeGrid.setSizeFull();  
 employeeGrid.removeAllColumns();  
 employeeGrid.addColumn(employee -> employee.getLastName() == null ? "-" : employee.getLastName()).setHeader("Фамилия").setSortable(true);  
 employeeGrid.addColumn(employee -> employee.getFirstName() == null ? "-" : employee.getFirstName()).setHeader("Имя").setSortable(true);  
 employeeGrid.addColumn(employee -> employee.getPosition() == null ? "-" : employee.getPosition()).setHeader("Должность").setSortable(true);  
 employeeGrid.addColumn(employee -> employee.getServices().size() == 0 ? "-" : *setToString*(employee.getServices())).setHeader("Оказываемые услуги").setSortable(false);  
 employeeGrid.addColumn(employee -> employee.getSalary().equals(new BigDecimal(0)) ? "-" : employee.getSalary()).setHeader("Зарплата").setSortable(true);  
 employeeGrid.addColumn(employee -> employee.getPhoneNumber() == null ? "-" : employee.getPhoneNumber()).setHeader("Номер телефона").setSortable(true);  
 employeeGrid.addColumn(employee -> employee.getEmail() == null ? "-" : employee.getEmail()).setHeader("email").setSortable(true);  
 employeeGrid.addColumn(employee -> employee.getStatus() == null ? "-" : employee.getStatus().getValue()).setHeader("Статус").setSortable(true);  
 employeeGrid.getColumns().forEach(customerColumn -> customerColumn.setAutoWidth(true));  
 employeeGrid.asSingleSelect().addValueChangeListener(event -> editEmployee(event.getValue()));  
 employeeGrid.addThemeVariants(GridVariant.*LUMO\_NO\_BORDER*, GridVariant.*LUMO\_NO\_ROW\_BORDERS*, GridVariant.*LUMO\_ROW\_STRIPES*);  
 }  
  
 private static String setToString(Set<Service> serviceSet) {  
 StringBuilder result = new StringBuilder();  
 for(Service service : serviceSet) {  
 result.append(service.getService()).append(", ");  
 }  
 result.delete(result.length()-2, result.length()-1);  
 return result.toString();  
 }  
  
 private void saveEmployee(EmployeeForm.SaveEvent saveEvent){  
 employeeService.save(saveEvent.getEmployee());  
 updateEmployeeList();  
 closeEmployeeEditor();  
 }  
  
 private void deleteEmployee(EmployeeForm.DeleteEvent deleteEvent){  
 employeeService.delete(deleteEvent.getEmployee());  
 updateEmployeeList();  
 closeEmployeeEditor();  
 }  
  
 private void editEmployee(Employee employee){  
 if(employee == null){  
 closeEmployeeEditor();  
 } else {  
 employeeForm.setEmployee(employee);  
 employeeForm.setVisible(true);  
 addClassName("employee-editing");  
 }  
 }  
  
 private void closeEmployeeEditor() {  
 employeeForm.setEmployee(null);  
 employeeForm.setVisible(false);  
 removeClassName("employee-editing");  
 }  
  
 private void updateEmployeeList(){  
 employeeGrid.setItems(employeeService.findAll(filter.getValue()));  
 }  
}

## **7.1.31 Листинг LoginView.java**

package ru.pahanjes.beautysaloon.crm.UI.views.login;  
  
import com.vaadin.flow.component.Key;  
import com.vaadin.flow.component.UI;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.html.H1;  
import com.vaadin.flow.component.html.H2;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.PasswordField;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import ru.pahanjes.beautysaloon.crm.exception.AuthException;  
import ru.pahanjes.beautysaloon.crm.security.AuthService;  
  
  
@Route(value = "login")  
@PageTitle("Login | BS CRM")  
public class LoginView extends VerticalLayout {  
  
 public LoginView(AuthService authService) {  
 setId("login-view");  
 setSizeFull();  
  
 setJustifyContentMode(JustifyContentMode.*CENTER*);  
 setAlignItems(Alignment.*CENTER*);  
  
 H1 headerWelcome = new H1("Beauty Saloon CRM");  
 headerWelcome.setId("welcome-header");  
 H2 headerLogIn = new H2("Вход");  
 headerLogIn.setId("login-header");  
  
 TextField username = new TextField("Имя пользователя");  
 username.setRequiredIndicatorVisible(true);  
 username.setRequired(true);  
 username.setErrorMessage("Поле не может быть пустым!");  
 username.setId("textfield-username");  
 PasswordField password = new PasswordField("Пароль");  
 password.setRequiredIndicatorVisible(true);  
 password.setRequired(true);  
 password.setErrorMessage("Поле не может быть пустым!");  
 password.setId("passwordfield-password");  
  
 Button buttonLogIn = new Button("Войти", click -> {  
 try {  
 authService.authenticate(username.getValue(), password.getValue());  
 UI.*getCurrent*().navigate("lk/welcome");  
 } catch (AuthException exception) {  
 Notification notification = new Notification(exception.getMessage(), 3000, Notification.Position.*TOP\_CENTER*);  
 notification.open();  
 }  
 });  
  
 buttonLogIn.addClickShortcut(Key.*ENTER*);  
  
 add(  
 headerWelcome,  
 headerLogIn,  
 username,  
 password,  
 buttonLogIn  
 );  
 }  
  
}

## **7.1.32 Листинг Logout.java**

package ru.pahanjes.beautysaloon.crm.UI.views.logout;  
  
import com.vaadin.flow.component.UI;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.server.VaadinSession;  
  
@Route(value = "logout")  
public class Logout extends VerticalLayout {  
  
 public Logout() {  
 UI.*getCurrent*().getPage().setLocation("login");  
 VaadinSession.*getCurrent*().getSession().invalidate();  
 VaadinSession.*getCurrent*().close();  
 }  
}

## **7.1.33 Листинг ServiceForm.java**

package ru.pahanjes.beautysaloon.crm.UI.views.service;  
  
import com.vaadin.flow.component.Component;  
import com.vaadin.flow.component.ComponentEvent;  
import com.vaadin.flow.component.ComponentEventListener;  
import com.vaadin.flow.component.Key;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.button.ButtonVariant;  
import com.vaadin.flow.component.formlayout.FormLayout;  
import com.vaadin.flow.component.icon.Icon;  
import com.vaadin.flow.component.icon.VaadinIcon;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.textfield.BigDecimalField;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.component.textfield.TextFieldVariant;  
import com.vaadin.flow.data.binder.BeanValidationBinder;  
import com.vaadin.flow.data.binder.Binder;  
import com.vaadin.flow.shared.Registration;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Service;  
  
import java.math.BigDecimal;  
  
public class ServiceForm extends FormLayout {  
 private TextField service = new TextField("Наименование");  
 private BigDecimalField price = new BigDecimalField("Цена");  
 private Binder<Service> binder = new BeanValidationBinder<>(Service.class);  
 private Button save = new Button("Сохранить");  
 private Button delete = new Button();  
 private Button close = new Button("Отменить");  
  
 public ServiceForm() {  
 addClassName("service-form");  
 configureBinder();  
 price.addThemeVariants(TextFieldVariant.*LUMO\_ALIGN\_RIGHT*);  
 price.setPrefixComponent(new Icon(VaadinIcon.*MONEY*));  
 price.setValue(new BigDecimal(0));  
 add(  
 service,  
 price,  
 createButtonLayout()  
 );  
 }  
  
 private void configureBinder() {  
 binder.forField(service)  
 .withValidator(service -> !service.isEmpty(), "Введите название услуги")  
 .bind(Service::getService, Service::setService);  
 binder.forField(price)  
 .bind(Service::getPrice, Service::setPrice);  
  
 service.setRequired(true);  
 service.setRequiredIndicatorVisible(true);  
 service.setPlaceholder("Введите наименование: ");  
  
 price.setRequiredIndicatorVisible(true);  
 price.setPlaceholder("Введите стоимость услуги: ");  
 }  
  
 private Component createButtonLayout() {  
 save.addThemeVariants(ButtonVariant.*LUMO\_PRIMARY*);  
 delete.addThemeVariants(ButtonVariant.*LUMO\_ERROR*);  
 close.addThemeVariants(ButtonVariant.*LUMO\_TERTIARY*);  
  
 save.addClickShortcut(Key.*ENTER*);  
 close.addClickShortcut(Key.*ESCAPE*);  
  
 save.addClickListener(save -> validateAndSave());  
 delete.addClickListener(delete -> fireEvent(new DeleteEvent(this, binder.getBean())));  
 close.addClickListener(close -> fireEvent(new CloseEvent(this)));  
  
 binder.addStatusChangeListener(event -> save.setEnabled(binder.isValid()));  
  
 return new HorizontalLayout(save, delete, close);  
 }  
  
 public void setService(Service service) {  
 binder.setBean(service);  
 if(binder.getBean() != null) {  
 if(binder.getBean().isActive()) {  
 delete.setEnabled(true);  
 delete.setText("Деактивировать");  
 } else {  
 delete.setEnabled(true);  
 delete.setText("Активировать");  
 }  
 } else {  
 delete.setText("Активация");  
 delete.setEnabled(false);  
 }  
 }  
  
 private void validateAndSave() {  
 if(binder.isValid()){  
 fireEvent(new SaveEvent(this, binder.getBean()));  
 }  
 }  
  
 public static abstract class ServiceFormEvent extends ComponentEvent<ServiceForm> {  
 private Service service;  
  
 protected ServiceFormEvent(ServiceForm source, Service service){  
 super(source, false);  
 this.service = service;  
 }  
  
 public Service getService(){  
 return service;  
 }  
 }  
  
 public static class SaveEvent extends ServiceFormEvent {  
 public SaveEvent(ServiceForm source, Service service) {  
 super(source, service);  
 }  
 }  
  
 public static class DeleteEvent extends ServiceFormEvent {  
 public DeleteEvent(ServiceForm source, Service service) {  
 super(source, service);  
 }  
 }  
  
 public static class CloseEvent extends ServiceFormEvent {  
 public CloseEvent(ServiceForm source) {  
 super(source, null);  
 }  
 }  
  
 public <T extends ComponentEvent<?>> Registration addListener (Class<T> eventType,  
 ComponentEventListener<T> listener) {  
 return getEventBus().addListener(eventType, listener);  
 }  
}

## **7.1.34 Листинг ServiceView.java**

package ru.pahanjes.beautysaloon.crm.UI.views.service;  
  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.dependency.CssImport;  
import com.vaadin.flow.component.grid.Grid;  
import com.vaadin.flow.component.grid.GridVariant;  
import com.vaadin.flow.component.html.Div;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Service;  
import ru.pahanjes.beautysaloon.crm.backend.service.ServiceService;  
  
@Route(value = "lk/servicelist")  
@PageTitle("Список услуг | BS")  
@CssImport("./styles/views/service-view.css")  
public class ServiceView extends VerticalLayout {  
 private final ServiceService serviceService;  
 private ServiceForm serviceForm;  
 private Button addServiceButton;  
 private Grid<Service> serviceGrid = new Grid<>(Service.class);;  
  
 public ServiceView(ServiceService serviceService) {  
 this.serviceService = serviceService;  
 addClassName("service-view");  
 setSizeFull();  
 addServiceButton = new Button("Добавить");  
 addServiceButton.addClickListener(click -> addService());  
 serviceForm = new ServiceForm();  
 serviceForm.addListener(ServiceForm.SaveEvent.class, this::saveService);  
 serviceForm.addListener(ServiceForm.DeleteEvent.class, this::deleteService);  
 serviceForm.addListener(ServiceForm.CloseEvent.class, closeEvent -> closeServiceEditor());  
 configureGrid();  
 Div content = new Div(serviceGrid, serviceForm);  
 content.addClassName("service-content");  
 content.setSizeFull();  
 add(  
 new HorizontalLayout(addServiceButton),  
 content  
 );  
 updateGrid();  
 closeServiceEditor();  
 }  
  
 private void addService() {  
 serviceGrid.asSingleSelect().clear();  
 editService(new Service());  
  
 }  
  
 private void saveService(ServiceForm.SaveEvent saveEvent) {  
 serviceService.save(saveEvent.getService());  
 updateGrid();  
 closeServiceEditor();  
 }  
  
 private void deleteService(ServiceForm.DeleteEvent deleteEvent) {  
 deleteEvent.getService().setActive(!deleteEvent.getService().isActive());  
 serviceService.save(deleteEvent.getService());  
 updateGrid();  
 closeServiceEditor();  
 }  
  
 private void editService(Service service) {  
 if(service == null) {  
 closeServiceEditor();  
 } else {  
 serviceForm.setService(service);  
 serviceForm.setVisible(true);  
 addClassName("service-editing");  
 }  
 }  
  
 private void closeServiceEditor() {  
 serviceForm.setService(null);  
 serviceForm.setVisible(false);  
 removeClassName("service-editing");  
 }  
  
 private void configureGrid() {  
 serviceGrid.setSizeFull();  
 serviceGrid.addClassName("service-grid");  
 serviceGrid.removeAllColumns();  
 serviceGrid.addColumn(service -> service.getService() == null ? "-" : service.getService()).setHeader("Наименование").setSortable(true);  
 serviceGrid.addColumn(service -> service.getPrice() == null ? "-" : service.getPrice()).setHeader("Стоимость").setSortable(true);  
 serviceGrid.getColumns().forEach(serviceColumn -> serviceColumn.setAutoWidth(true));  
 serviceGrid.asSingleSelect().addValueChangeListener(event -> editService(event.getValue()));  
 serviceGrid.addThemeVariants(GridVariant.*LUMO\_NO\_BORDER*, GridVariant.*LUMO\_NO\_ROW\_BORDERS*, GridVariant.*LUMO\_ROW\_STRIPES*);  
 }  
  
 private void updateGrid() {  
 serviceGrid.setItems(serviceService.findAll());  
 }  
}

## **7.1.35 Листинг LandingView.java**

package ru.pahanjes.beautysaloon.crm.UI.views;  
  
import com.vaadin.flow.component.Component;  
import com.vaadin.flow.component.UI;  
import com.vaadin.flow.component.html.Image;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.tabs.Tab;  
import com.vaadin.flow.component.tabs.Tabs;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import ru.pahanjes.beautysaloon.crm.backend.service.CustomerService;  
import ru.pahanjes.beautysaloon.crm.backend.service.EmployeeService;  
import ru.pahanjes.beautysaloon.crm.backend.service.ServiceService;  
  
@Route(value = "main")  
@PageTitle("Салон красоты | BS main")  
public class LandingView extends VerticalLayout {  
  
 private final Tabs menu = new Tabs();  
 private final CustomerService customerService;  
 private final EmployeeService employeeService;  
 private ServiceService serviceService;  
 private final RegisterDialog registerDialog;  
  
 public LandingView(CustomerService customerService, EmployeeService employeeService, ServiceService serviceService) {  
 this.customerService = customerService;  
 this.employeeService = employeeService;  
 this.serviceService = serviceService;  
 registerDialog = new RegisterDialog(customerService, employeeService, serviceService);  
  
 add(  
 getAllComponents()  
 );  
 }  
  
 private Component getMenu() {  
 Tab login = new Tab("Вход");  
 Image logo = new Image("images/logo.png", "Beauty Saloon");  
 logo.setWidth("65px");  
 logo.setHeight("34px");  
 Tab main = new Tab(logo);  
 Tab register = new Tab("Запись");  
  
 menu.add(login, main, register);  
 menu.setAutoselect(false);  
 menu.setSelectedIndex(1);  
  
 menu.addSelectedChangeListener(selectedChangeEvent -> {  
 if(selectedChangeEvent.getSelectedTab().getLabel().equals("Вход")) {  
 UI.*getCurrent*().navigate("login");  
 } else if(selectedChangeEvent.getSelectedTab().getLabel().equals("Запись")) {  
 registerDialog.open();  
 }  
 menu.setSelectedIndex(1);  
 });  
  
 HorizontalLayout header = new HorizontalLayout();  
 header.setWidthFull();  
 header.setJustifyContentMode(JustifyContentMode.*CENTER*);  
 header.add(menu);  
 return header;  
 }  
  
 private Component getBody() {  
 VerticalLayout body = new VerticalLayout();  
 HorizontalLayout blocksRow = new HorizontalLayout();  
 Image haircut = new Image("images/haircut.jpeg", "");  
 haircut.setWidth("150px");  
 haircut.setHeight("150px");  
 haircut.setTitle("Стрижка");  
  
 body.add(blocksRow);  
  
 return body;  
 }  
  
 private Component[] getAllComponents() {  
 return new Component[] {  
 getMenu(),  
 getBody()  
 };  
 }  
  
}

## **7.1.36 Листинг RegisterDialog.java**

package ru.pahanjes.beautysaloon.crm.UI.views;  
  
import com.vaadin.componentfactory.multiselect.MultiComboBox;  
import com.vaadin.flow.component.Component;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.button.ButtonVariant;  
import com.vaadin.flow.component.datepicker.DatePicker;  
import com.vaadin.flow.component.datetimepicker.DateTimePicker;  
import com.vaadin.flow.component.dialog.Dialog;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.orderedlayout.FlexComponent;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.EmailField;  
import com.vaadin.flow.component.textfield.TextField;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Customer;  
import ru.pahanjes.beautysaloon.crm.backend.entity.Service;  
import ru.pahanjes.beautysaloon.crm.backend.service.CustomerService;  
import ru.pahanjes.beautysaloon.crm.backend.service.EmployeeService;  
import ru.pahanjes.beautysaloon.crm.backend.service.ServiceService;  
  
import java.time.DayOfWeek;  
import java.time.LocalDateTime;  
import java.util.Arrays;  
import java.util.Optional;  
  
public class RegisterDialog extends Dialog {  
  
 private TextField firstName = new TextField("Имя");  
 private TextField lastName = new TextField("Фамилия");  
 private EmailField email = new EmailField("Электронная почта");  
 private TextField phoneNumber = new TextField("Номер телефона");  
 private MultiComboBox<Service> services = new MultiComboBox<>("Услуги");  
 private DateTimePicker timetable = new DateTimePicker();  
 private CustomerService customerService;  
 private EmployeeService employeeService;  
 private ServiceService serviceService;  
 private Customer customer;  
  
 private Button save = new Button("Записаться");  
 private Button close = new Button("Отменить");  
  
 public RegisterDialog(CustomerService customerService, EmployeeService employeeService, ServiceService serviceService) {  
 this.customerService = customerService;  
 this.employeeService = employeeService;  
 this.serviceService = serviceService;  
  
 configureDialog();  
 add(  
 getDialogForm()  
 );  
 }  
  
 private Component[] getDialogForm() {  
 HorizontalLayout name = new HorizontalLayout();  
 name.setWidthFull();  
 name.setJustifyContentMode(FlexComponent.JustifyContentMode.*CENTER*);  
 name.add(firstName, lastName);  
  
 HorizontalLayout contacts = new HorizontalLayout();  
 contacts.setWidthFull();  
 contacts.setJustifyContentMode(FlexComponent.JustifyContentMode.*CENTER*);  
 contacts.add(email, phoneNumber);  
  
 HorizontalLayout services = new HorizontalLayout();  
 services.setWidthFull();  
 services.setJustifyContentMode(FlexComponent.JustifyContentMode.*CENTER*);  
 services.add(this.services);  
  
 HorizontalLayout buttons = new HorizontalLayout();  
 buttons.setWidthFull();  
 buttons.setJustifyContentMode(FlexComponent.JustifyContentMode.*CENTER*);  
 buttons.add(save, close);  
  
 VerticalLayout timetableAndButtons = new VerticalLayout();  
 timetableAndButtons.setWidthFull();  
 timetableAndButtons.setJustifyContentMode(FlexComponent.JustifyContentMode.*CENTER*);  
 timetableAndButtons.add(timetable, buttons);  
  
 return new Component[] {  
 name,  
 contacts,  
 services,  
 timetableAndButtons  
 };  
 }  
  
 private void configureDialog() {  
 firstName.setPlaceholder("Виктория");  
 firstName.setRequired(true);  
 firstName.setRequiredIndicatorVisible(true);  
 firstName.setErrorMessage("Введите имя");  
  
 lastName.setPlaceholder("Павловна");  
 lastName.setRequired(true);  
 lastName.setRequiredIndicatorVisible(true);  
 lastName.setErrorMessage("Введите фамилию");  
  
 email.setPlaceholder("angelina\_pavlovla@gmail.com");  
 email.setRequiredIndicatorVisible(true);  
 email.setErrorMessage("Введите корректный адрес электронной почты");  
  
 phoneNumber.setPlaceholder("+7-xxx-xxx-xx-xx");  
 phoneNumber.setRequired(true);  
 phoneNumber.setRequiredIndicatorVisible(true);  
 phoneNumber.setErrorMessage("Введите номер телефона");  
  
 services.setItems(serviceService.findAll());  
 services.setItemLabelGenerator(Service::getService);  
 services.setRequired(true);  
 services.setRequiredIndicatorVisible(true);  
 services.setPlaceholder("Стрижка");  
 services.setErrorMessage("Выберите услуги");  
 services.setI18n(new MultiComboBox.MultiComboBoxI18n()  
 .setClear("Очистить")  
 .setSelect("Выбрать все"));  
  
 timetable.setMin(LocalDateTime.*now*().plusDays(1L));  
 timetable.setMax(LocalDateTime.*now*().plusMonths(1L));  
 timetable.setDatePickerI18n(new DatePicker.DatePickerI18n()  
 .setWeek("Неделя").setCalendar("Календарь").setClear("Очистить")  
 .setToday("Сегодня").setCancel("Отмена").setFirstDayOfWeek(1)  
 .setMonthNames(Arrays.*asList*(  
 "Январь", "Февраль", "Март", "Апрель",  
 "Май", "Июнь", "Июль", "Август", "Сентябрь",  
 "Октябрь", "Ноябрь", "Декабрь"))  
 .setWeekdays(Arrays.*asList*(  
 "Воскресенье", "Понедельник", "Вторник", "Среда",  
 "Четверг", "Пятница", "Суббота"))  
 .setWeekdaysShort(Arrays.*asList*(  
 "Вс", "Пн", "Вт", "Ср", "Чт", "Пт", "Сб"))  
 );  
 timetable.setRequiredIndicatorVisible(true);  
 timetable.setErrorMessage("Выберите дату и время");  
  
 save.addClickListener(save -> saveCustomer());  
 save.addThemeVariants(ButtonVariant.*LUMO\_PRIMARY*);  
  
 close.addClickListener(close -> close());  
 close.addThemeVariants(ButtonVariant.*LUMO\_TERTIARY*);  
  
 }  
  
 private void saveCustomer() {  
 if (firstName.getValue().isEmpty()) {  
 Notification notification = new Notification("Введите имя", 3000, Notification.Position.*TOP\_CENTER*);  
 notification.open();  
 return;  
 } else if (lastName.getValue().isEmpty()) {  
 Notification notification = new Notification("Введите фамилию", 3000, Notification.Position.*TOP\_CENTER*);  
 notification.open();  
 return;  
 } else if (email.getValue().isEmpty()) {  
 Notification notification = new Notification("Введите адрес электронной почты", 3000, Notification.Position.*TOP\_CENTER*);  
 notification.open();  
 return;  
 } else if (phoneNumber.getValue().isEmpty()) {  
 Notification notification = new Notification("Введите номер", 3000, Notification.Position.*TOP\_CENTER*);  
 notification.open();  
 return;  
 } else if (services.getValue().size() == 0) {  
 Notification notification = new Notification("Выберите услуги", 3000, Notification.Position.*TOP\_CENTER*);  
 notification.open();  
 return;  
 } else if(timetable.getValue().getDayOfWeek() == DayOfWeek.*SUNDAY* || timetable.getValue().getHour() > 21 || timetable.getValue().getHour() < 8) {  
 Notification notification = new Notification("Запись может быть произведена только с понедельника по субботу с 8:00 до 21:00",  
 3000, Notification.Position.*TOP\_CENTER* );  
 notification.open();  
 return;  
 }  
 Optional<Service> notActiveService = services.getValue().stream().findFirst().filter(service -> !service.isActive());  
 if(notActiveService.isPresent()) {  
 Notification notification = new Notification("Услуга \"" + notActiveService.get().getService() + "\" временно недоступна",  
 3000, Notification.Position.*TOP\_CENTER*);  
 notification.open();  
 return;  
 }  
  
 customer = new Customer();  
 customer.setFirstName(firstName.getValue());  
 customer.setLastName(lastName.getValue());  
 customer.setEmail(email.getValue());  
 customer.setPhoneNumber(phoneNumber.getValue());  
 customer.setServices(services.getValue());  
 customer.setTimetable(timetable.getValue());  
 customer.setStatus(Customer.Status.*SignedUp*);  
 customerService.save(customer);  
 Notification.*show*(firstName.getValue() + ", Вы были успешно записаны");  
 close();  
 }  
  
}

## **7.1.37 Листинг Constants.java**

package ru.pahanjes.beautysaloon.crm.util;  
  
import java.time.LocalDateTime;  
import java.time.Month;  
  
public final class Constants {  
 public static final LocalDateTime *COMPANY\_FOUNDATION\_DATE* = LocalDateTime.*of*(2020, Month.*DECEMBER*, 25, 0, 0, 0);  
}

## **7.1.38 Листинг Application.java**

package ru.pahanjes.beautysaloon.crm;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.boot.autoconfigure.web.servlet.error.ErrorMvcAutoConfiguration;  
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;  
  
@SpringBootApplication(exclude = ErrorMvcAutoConfiguration.class)  
public class Application extends SpringBootServletInitializer {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(Application.class, args);  
 }  
  
}

## **7.1.39 Листинг application.properties**

server.port=${PORT:8080}  
# Ensure application is run in Vaadin 14/npm mode  
vaadin.compatibilityMode = false  
logging.level.org.atmosphere = warn  
  
# Database settings  
spring.jpa.hibernate.ddl-auto=update  
spring.datasource.url=jdbc:mysql://${MYSQL\_HOST:localhost}:3306/db\_beauty\_saloon  
spring.datasource.username=bs\_user  
spring.datasource.password=sd\*sa2!l  
  
# LiveReload settings  
spring.devtools.restart.poll-interval=2s  
spring.devtools.restart.quiet-period=1s  
  
# Suppress WARN c.v.f.s.c.ReturnChannelHandler - Ignoring update for disabled return channel:  
logging.level.com.vaadin.flow.server.communication.ReturnChannelHandler=ERROR  
# Suppress WARN c.v.f.s.c.rpc.MapSyncRpcHandler - Property update request for disabled element is received from the client side.  
logging.level.com.vaadin.flow.server.communication.rpc.MapSyncRpcHandler=ERROR  
logging.level.com.vaadin.flow.server.communication.rpc.AbstractRpcInvocationHandler=ERROR

## **7.1.40 Листинг shared-styles.css**

/\* Main layout \*/  
  
a[highlight] {  
 font-weight: bold;  
 text-decoration: underline;  
}  
  
.header {  
 padding: 0 var(--lumo-space-m);  
}  
  
.header h1.logo {  
 font-size: 1em;  
 margin: var(--lumo-space-m);  
}  
  
/\* Customer list view\*/  
.customer-list-view .customer-content {  
 display: flex;  
}  
  
.customer-list-view .customer-grid {  
 flex: 2;  
}  
  
.customer-list-view .customer-form {  
 flex: 1;  
 padding: var(--lumo-space-m);  
}  
  
/\* Employee list view\*/  
.employee-list-view .employee-content {  
 display: flex;  
}  
  
.employee-list-view .employee-grid {  
 flex: 2;  
}  
  
.employee-list-view .employee-form {  
 flex: 1;  
 padding: var(--lumo-space-m);  
}  
  
@media all and (max-width: 1100px) {  
 .customer-list-view.editing .toolbar,  
 .customer-list-vew.editing .customer.grid {  
 display: none;  
 }  
  
 .employee-list-view.editing .toolbar,  
 .employee-list-view.editing .employee.grid {  
 display: none;  
 }  
}  
  
/\*  
.dashboard-view .contact-stats {  
 font-size: 4em;  
 margin: 1em 0;  
}\*/

## **7.1.41 Листинг service-view.css**

.service-view .service-content {  
 display: flex;  
}  
  
.service-view .service-grid {  
 flex: 2;  
}  
  
.service-view .service-form {  
 flex: 1;  
 padding: var(--lumo-space-m);  
}

## **7.1.42 Листинг employee-view.css**

/\* Employee list view\*/  
.employee-list-view .employee-content {  
 display: flex;  
}  
  
.employee-list-view .employee-grid {  
 flex: 2;  
}  
  
.employee-list-view .employee-form {  
 flex: 1;  
 padding: var(--lumo-space-m);  
}

## **7.1.43 Листинг customer-view.css**

/\* Customer list view\*/  
.customer-list-view .customer-content {  
 display: flex;  
}  
  
.customer-list-view .customer-grid {  
 flex: *2*;  
}  
  
.customer-list-view .customer-form {  
 flex: *1*;  
 padding: var(--lumo-space-m);  
}

## **7.1.44 Листинг change-view.css**

.change-view .user-info {  
 display: flex;  
}  
  
.change-view .user-form {  
 flex: 2;  
}  
  
.change-view .employee-form {  
 flex: 1;  
 padding: var(--lumo-space-m);  
}

## **7.1.45 Листинг cabinet-view.css**

#header {  
 height: var(--lumo-size-xl);  
}  
#header h1 {  
 font-size: var(--lumo-font-size-xxl);  
 margin-top: var(--lumo-space-m);  
}  
#header h5{  
 margin-right: var(--lumo-space-m);  
 margin-bottom: var(--lumo-space-m);  
}  
#anchor {  
 font-size: var(--lumo-font-size-xl);  
 margin-right: var(--lumo-space-m);  
 color: var(--lumo-secondary-text-color);  
}  
#logo {  
 box-sizing: border-box;  
 box-shadow: inset 0 -1px var(--lumo-contrast-10pct);  
 padding: var(--lumo-space-s) var(--lumo-space-m);  
}  
#logo img {  
 height: calc(var(--lumo-size-s) \* 1.2);  
}  
#logo h1 {  
 font-size: var(--lumo-font-size-xxxl);  
 font-weight: 600;  
 margin: 0 var(--lumo-space-s);  
}  
  
vaadin-tab {  
 font-size: var(--lumo-font-size-s);  
 height: var(--lumo-size-l);  
 font-weight: 600;  
 color: var(--lumo-body-text-color);  
}  
  
vaadin-tab:hover {  
 background-color: var(--lumo-contrast-5pct);  
}  
  
vaadin-tab[selected] {  
 background-color: var(--lumo-primary-color-10pct);  
 color: var(--lumo-primary-text-color);  
}

## **7.1.45 Листинг pom.xml**

<?xml version="1.0" encoding="UTF-8"?><project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <groupId>com.vaadin.tutorial.crm</groupId>  
 <artifactId>vaadin-crm</artifactId>  
 <name>Vaadin CRM</name>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <properties>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  
  
 <vaadin.version>14.4.4</vaadin.version>  
  
 </properties>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>2.3.2.RELEASE</version>  
 </parent>  
  
 <repositories>  
 <!-- The order of definitions matters. Explicitly defining central here to make sure it has the highest priority. -->  
  
 <!-- Main Maven repository -->  
 <repository>  
 <id>central</id>  
 <url>https://repo.maven.apache.org/maven2</url>  
 <snapshots>  
 <enabled>false</enabled>  
 </snapshots>  
 </repository>  
 <!-- Repository used by many Vaadin add-ons -->  
 <repository>  
 <id>vaadin-addons</id>  
 <url>http://maven.vaadin.com/vaadin-addons</url>  
 </repository>  
 </repositories>  
  
 <pluginRepositories>  
 <!-- Main Maven repository -->  
 <pluginRepository>  
 <id>central</id>  
 <url>https://repo.maven.apache.org/maven2</url>  
 <snapshots>  
 <enabled>false</enabled>  
 </snapshots>  
 </pluginRepository>  
 </pluginRepositories>  
  
 <dependencyManagement>  
 <dependencies>  
 <dependency>  
 <groupId>com.vaadin</groupId>  
 <artifactId>vaadin-bom</artifactId>  
 <version>${vaadin.version}</version>  
 <type>pom</type>  
 <scope>import</scope>  
 </dependency>  
 </dependencies>  
 </dependencyManagement>  
  
 <dependencies>  
 <dependency>  
 <groupId>com.vaadin</groupId>  
 <!-- Replace artifactId with vaadin-core to use only free components -->  
 <artifactId>vaadin</artifactId>  
 <exclusions>  
 <!-- Webjars are only needed when running in Vaadin 13 compatibility mode -->  
 <exclusion>  
 <groupId>com.vaadin.webjar</groupId>  
 <artifactId>\*</artifactId>  
 </exclusion>  
 <exclusion>  
 <groupId>org.webjars.bowergithub.insites</groupId>  
 <artifactId>\*</artifactId>  
 </exclusion>  
 <exclusion>  
 <groupId>org.webjars.bowergithub.polymer</groupId>  
 <artifactId>\*</artifactId>  
 </exclusion>  
 <exclusion>  
 <groupId>org.webjars.bowergithub.polymerelements</groupId>  
 <artifactId>\*</artifactId>  
 </exclusion>  
 <exclusion>  
 <groupId>org.webjars.bowergithub.vaadin</groupId>  
 <artifactId>\*</artifactId>  
 </exclusion>  
 <exclusion>  
 <groupId>org.webjars.bowergithub.webcomponents</groupId>  
 <artifactId>\*</artifactId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
 <dependency>  
 <groupId>com.vaadin</groupId>  
 <artifactId>vaadin-spring-boot-starter</artifactId>  
 <exclusions>  
 <!-- Excluding so that webjars are not included. -->  
 <exclusion>  
 <groupId>com.vaadin</groupId>  
 <artifactId>vaadin-core</artifactId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
 <!-- Bean validation implementation-->  
 <dependency>  
 <groupId>org.hibernate.validator</groupId>  
 <artifactId>hibernate-validator</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>com.vaadin</groupId>  
 <artifactId>vaadin-testbench</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>io.github.bonigarcia</groupId>  
 <artifactId>webdrivermanager</artifactId>  
 <version>3.8.1</version>  
 <scope>test</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>com.vaadin</groupId>  
 <artifactId>flow-server-production-mode</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.security</groupId>  
 <artifactId>spring-security-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.security</groupId>  
 <artifactId>spring-security-config</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>com.vaadin.componentfactory</groupId>  
 <artifactId>multi-combo-box-flow</artifactId>  
 <version>0.4.0</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.vaadin</groupId>  
 <artifactId>textfieldformatter</artifactId>  
 <version>5.2.0</version>  
 </dependency>  
  
 </dependencies>  
  
 <build>  
 <defaultGoal>spring-boot:run</defaultGoal>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 <!-- Clean build and startup time for Vaadin apps sometimes may exceed  
 the default Spring Boot's 30sec timeout. -->  
 <configuration>  
 <wait>500</wait>  
 <maxAttempts>240</maxAttempts>  
 </configuration>  
 </plugin>  
  
 <!--  
 Take care of synchronizing java dependencies and imports in  
 package.json and main.js files.  
 It also creates webpack.config.js if not exists yet.  
 -->  
 <plugin>  
 <groupId>com.vaadin</groupId>  
 <artifactId>vaadin-maven-plugin</artifactId>  
 <version>${vaadin.version}</version>  
 <executions>  
 <execution>  
 <goals>  
 <goal>prepare-frontend</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
 </plugins>  
 </build>  
  
 <profiles>  
 <profile>  
 <!-- Production mode is activated using -Pproduction -->  
 <id>production</id>  
 <properties>  
 <vaadin.productionMode>true</vaadin.productionMode>  
 </properties>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 <configuration>  
 <jvmArguments>-Dvaadin.productionMode</jvmArguments>  
 </configuration>  
 </plugin>  
 <plugin>  
 <groupId>com.vaadin</groupId>  
 <artifactId>vaadin-maven-plugin</artifactId>  
 <version>${vaadin.version}</version>  
 <executions>  
 <execution>  
 <goals>  
 <goal>build-frontend</goal>  
 </goals>  
 <phase>compile</phase>  
 </execution>  
 </executions>  
 </plugin>  
 </plugins>  
 </build>  
 </profile>  
  
 <profile>  
 <id>it</id>  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 <executions>  
 <execution>  
 <id>start-spring-boot</id>  
 <phase>pre-integration-test</phase>  
 <goals>  
 <goal>start</goal>  
 </goals>  
 </execution>  
 <execution>  
 <id>stop-spring-boot</id>  
 <phase>post-integration-test</phase>  
 <goals>  
 <goal>stop</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
  
 <!-- Runs the integration tests (\*IT) after the server is started -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-failsafe-plugin</artifactId>  
 <executions>  
 <execution>  
 <goals>  
 <goal>integration-test</goal>  
 <goal>verify</goal>  
 </goals>  
 </execution>  
 </executions>  
 <configuration>  
 <trimStackTrace>false</trimStackTrace>  
 <enableAssertions>true</enableAssertions>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
 </profile>  
  
 </profiles>  
</project>