**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,   
СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»  
(СПбГУТ)**

**КУРСОВАЯ РАБОТА**

по дисциплине: «Разработка Java-приложений управления

телекоммуникациями»

На тему: «Контактный менеджер»

Курс: 3

Группа: ИКПИ-95

Студент: Крюков Н.М.

Работу принял: Зам. зав. каф. по научной работе Помогалова А. В.

Санкт-Петербург

2021

СОДЕРЖАНИЕ

[**Цель и задачи работы** 3](#_Toc92533618)

[**Выбор и обоснования инструментов разработки** 4](#_Toc92533619)

[**Разработка структуры клиент-серверного приложения** 5](#_Toc92533620)

[**Архитектура проекта** 7](#_Toc92533621)

[**Результат работы** 8](#_Toc92533622)

[СПИСОК ЛИТЕРАТУРЫ 11](#_Toc92533623)

[ПРИЛОЖЕНИЕ 12](#_Toc92533624)

# **Цель и задачи работы**

Цель работы: разработать графическое web-приложение на языке программирования Java по заданной теме.

В отчете должно быть описание используемого стека технологий, архитектура решения, описание функций, возможных на платформе.

Разрешено использование утилитарных библиотек и фреймворков.

Задачи:

1. Реализовать сервер, который будет удовлетворять запросам выбранной темы.
2. Реализовать клиент, который будет обращаться к API.

# **Выбор и обоснования инструментов разработки**

Для разработки серверной части будет использоваться библиотека Spring, а конкретно Spring Boot, Spring Security.

Spring Boot неявно упаковывает необходимые сторонние зависимости для каждого типа приложения на основе Spring и предоставляет их разработчику посредством так называемых starter-пакетов (spring-boot-starter-web, spring-boot-starter-data-jpa и т.д.)

Spring Security используется для реализации окна авторизации в приложении, методы которого переопределяется для замены встроенного окна авторизации на созданное представление в рамках проекта.

Реализация UI представлена фреймворком Vaadin Flow, представляющий собой инструмент для реализации графического интерфейса при помощи встроенных методов создания элементов в рамках DOM (Document Object Model) без использования HTML, CSS, JavaScript.

Для работы с базой данных будет использоваться СУБД PostgreSQL, так как инструменты для работы с БД предоставляется фреймворком Vaadin.

Для написания кода будет использована Intellj IDEA.

# **Разработка структуры клиент-серверного приложения**

Клиент – серверное приложение будет состоять из клиента и сервера соответственно.

Серверная часть подключена к базе данных, получает запросы обрабатывает их и отправляет результат.

Клиентская часть позволяет взаимодействовать с сервером через интерфейс (принимать информацию от пользователя и отправлять их на сервер, принимать информацию от сервера и предоставлять информацию пользователю в виде интерфейса).

Интерфейс представляет собой три представления в рамках пакета views.list : ContactForm, ListView, LoginView, объединенных в рамках класса MainLayout, который отвечает за корректное отображение интерфейса.

ContactForm реализует окно с полями для редактирования записей в БД.

LoginView реализует окно авторизации в приложении, которое переопределяет методы Spring Security и заменят собой стандартное представление окна авторизации, встроенное в Spring Security. В рамках проекта был реализован один пользователь с ролью «USER» при помощи инструментов фреймворка Vaadin flow. В дальнейшем при помощи встроенного в фреймворк Vaadin flow класса VaadinWebSecurityConfigurerAdapter возможно добавление неограниченного количества пользователей.

Разный уровень доступа к разным представлениям в рамках проекта возможно при помощи библиотеки javax.annotation.security.

ListView реализует вывод полей БД а также обработку событий при нажатии кнопок «Save», «Delete», «Cancel».

База данных состоит из нескольких моделей, предоставляемых фреймворком Vaadin Flow – CompanyRepository, ContactRepository, StatusRepository.

В рамках проекта используется лишь ContactRepository с полями:

* firstName типа String,
* lastName типа String,
* company типа Company,
* status типа Status,
* email типа String

# **Архитектура проекта**

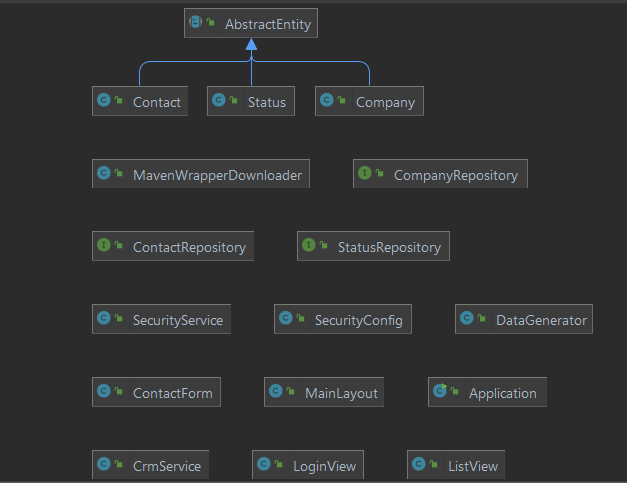


Рисунок 1 – Блок-схема взаимосвязей классов проекта

# **Результат работы**

Проект запускается с окна авторизации, в котором зарегистрированный пользователь вводит свои данные и заходит на сервис.

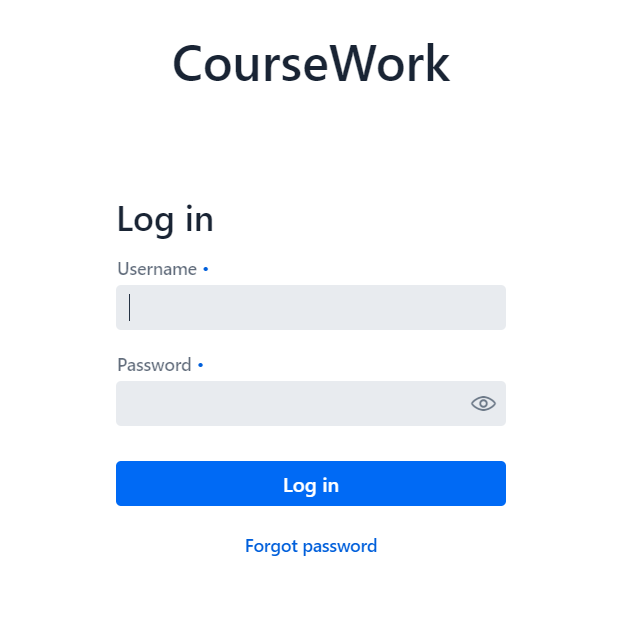


Рисунок 2 – Поле авторизации

Интерфейс приложения представляет собой таблицу с полями «First Name», «Last Name», «Email», «Status», «Company» а также поле с данными пользователя и кнопками «Save», «Delete», «Cancel», предназначенные для редактирования записей в таблице.

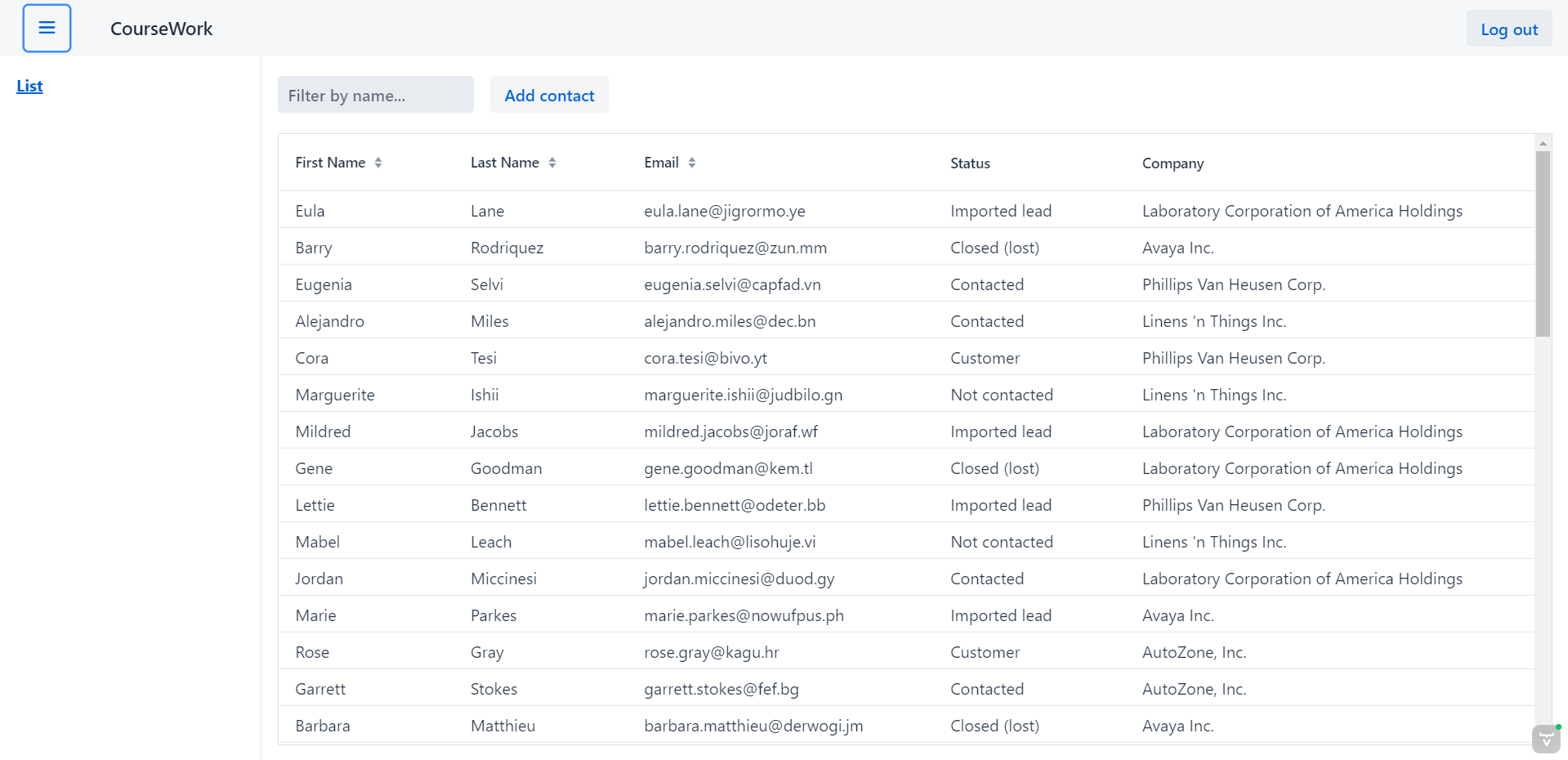


Рисунок 3 – Интерфейс веб-приложения

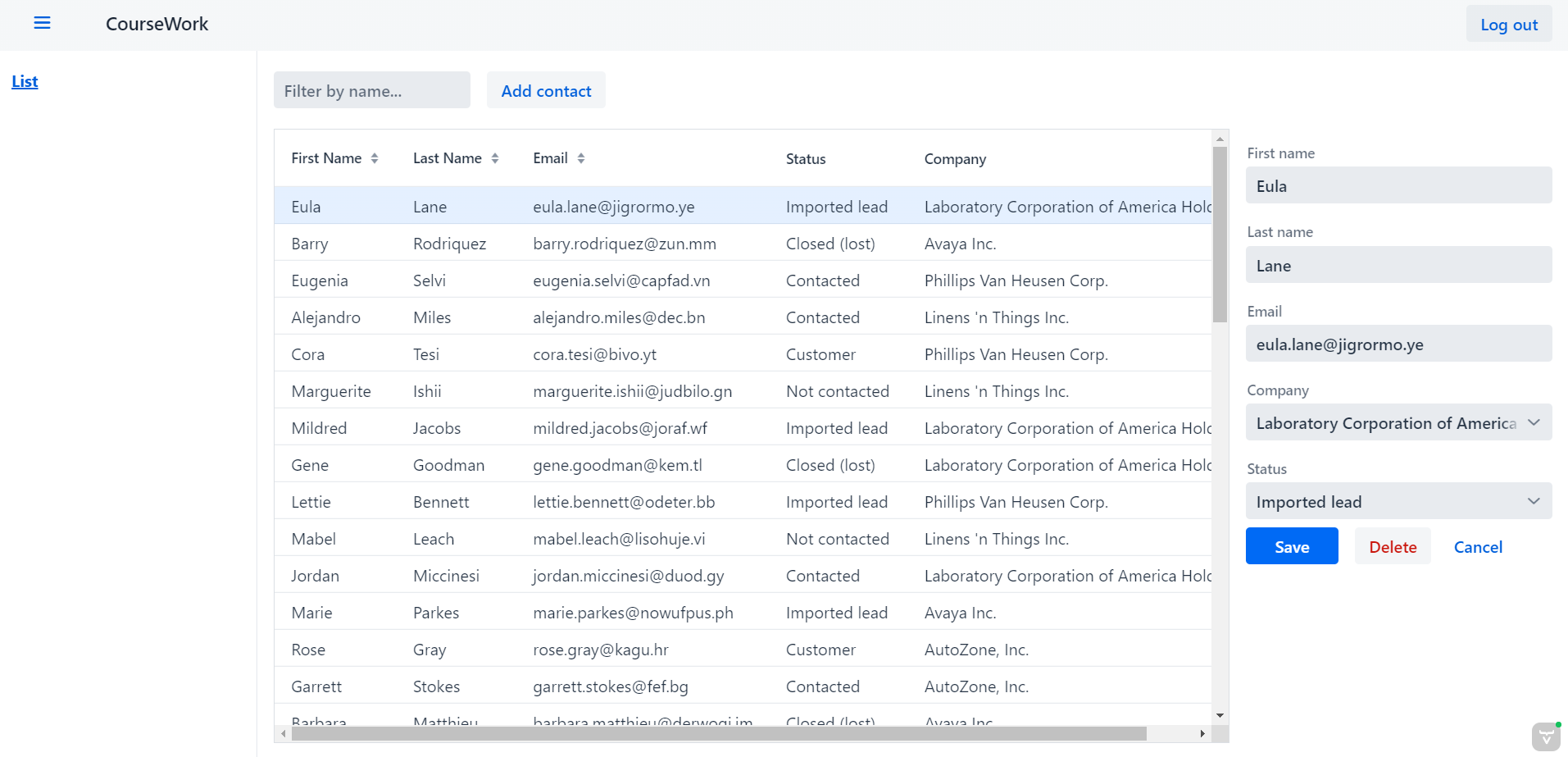


Рисунок 4 – Интерфейс веб-приложения с формой для редактирования записей

**Вывод**

1. Был произведен поиск информации по заданной теме и на основе собранного материала было сформировано клиент-серверное приложение.
2. Поставленные задачи были выполнены.
3. Требования к оформлению были выполнены, веб-приложение удовлетворяет запросу о единстве стилевого оформления, удобстве навигации, структурирование текста, наличие анимации.
4. Были использованы все обязательные и необязательные средства: Java, библиотека Java Spring, Spring Boot, фреймворк Vaadin flow.

# СПИСОК ЛИТЕРАТУРЫ

1. Альфред, В. Ахо Компиляторы. Принципы, технологии и инструментарий / Альфред В. Ахо и др. - М.: Вильямс, 2015. - 689 c.
2. Берд, Барри Java для чайников / Барри Берд. - М.: Диалектика / Вильямс, 2013. - 521 c.
3. Гарнаев, Андрей WEB-программирование на Java и JavaScript / Андрей Гарнаев , Сергей Гарнаев. - Москва: СПб. [и др.] : Питер, 2017. - 718 c.
4. Гонсалвес, Энтони Изучаем Java EE 7 / Энтони Гонсалвес. - М.: Питер, 2016. - 640 c.
5. Гупта, Арун Java EE 7. Основы / Арун Гупта. - М.: Вильямс, 2014. - 336 c.
6. Монахов, В. Язык программирования Java и среда NetBeans (+ CD-ROM) / В. Монахов. - М.: БХВ-Петербург, 2012. - 720 c
7. Савитч, Уолтер Язык Java. Курс программирования / Уолтер Савитч. - М.: Вильямс, 2015. - 928 c.
8. Хабибуллин, Ильдар Самоучитель Java / Ильдар Хабибуллин. - М.: БХВ-Петербург, 2014. - 768 c.
9. Шилдт, Герберт Java 8. Руководство для начинающих / Герберт Шилдт. - М.: Вильямс, 2015. - 720 c.
10. Алексеев А.. Введение в Web-дизайн. Учебное пособие. — М.: ДМК Пресс, 2019. — 184 c.
11. Гарретт Д. Веб-дизайн. Элементы опыта взаимодействия / Д. Гарретт. — СПб.: Символ-плюс, 2015. — 192 c.
12. Гарретт Джесс. Веб-дизайн. Элементы опыта взаимодействия. — М.: Символ-Плюс, 2020. — 285 c.

# ПРИЛОЖЕНИЕ

**ContactRepository.java**

package com.example.application.data.repository;

import com.example.application.data.entity.Contact;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

import java.util.List;

public interface ContactRepository extends JpaRepository<Contact, Long> {

@Query("select c from Contact c " +

"where lower(c.firstName) like lower(concat('%', :searchTerm, '%')) " +

"or lower(c.lastName) like lower(concat('%', :searchTerm, '%'))")

List<Contact> search(@Param("searchTerm") String searchTerm);

}

**CrmService.java**

package com.example.application.data.service;

import com.example.application.data.entity.Company;

import com.example.application.data.entity.Contact;

import com.example.application.data.entity.Status;

import com.example.application.data.repository.CompanyRepository;

import com.example.application.data.repository.ContactRepository;

import com.example.application.data.repository.StatusRepository;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CrmService {

private final ContactRepository contactRepository;

private final CompanyRepository companyRepository;

private final StatusRepository statusRepository;

public CrmService(ContactRepository contactRepository,

CompanyRepository companyRepository,

StatusRepository statusRepository) {

this.contactRepository = contactRepository;

this.companyRepository = companyRepository;

this.statusRepository = statusRepository;

}

public List<Contact> findAllContacts(String stringFilter) {

if (stringFilter == null || stringFilter.isEmpty()) {

return contactRepository.findAll();

} else {

return contactRepository.search(stringFilter);

}

}

public long countContacts() {

return contactRepository.count();

}

public void deleteContact(Contact contact) {

contactRepository.delete(contact);

}

public void saveContact(Contact contact) {

if (contact == null) {

System.err.println("Contact is null. Are you sure you have connected your form to the application?");

return;

}

contactRepository.save(contact);

}

public List<Company> findAllCompanies() {

return companyRepository.findAll();

}

public List<Status> findAllStatuses(){

return statusRepository.findAll();

}

}

**SecurityConfig.java**

package com.example.application.security;

import com.example.application.views.list.LoginView;

import com.vaadin.flow.spring.security.VaadinWebSecurityConfigurerAdapter;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

@EnableWebSecurity

@Configuration

public class SecurityConfig extends VaadinWebSecurityConfigurerAdapter {

@Override

protected void configure(HttpSecurity http) throws Exception {

super.configure(http);

setLoginView(http,LoginView.class);

}

@Override

public void configure(WebSecurity web) throws Exception {

web.ignoring().antMatchers("/images/\*\*");

super.configure(web);

}

@Bean

@Override

protected UserDetailsService userDetailsService() {

return new InMemoryUserDetailsManager(User.withUsername("user").password("{noop}userpass").roles("USER").build());

}}

**SecurityService.java**

package com.example.application.security;

import com.vaadin.flow.component.UI;

import com.vaadin.flow.server.VaadinServletRequest;

import org.springframework.security.web.authentication.logout.SecurityContextLogoutHandler;

import org.springframework.stereotype.Component;

@Component

public class SecurityService {

public void logout(){

UI.getCurrent().getPage().setLocation("/");

SecurityContextLogoutHandler logoutHandler = new SecurityContextLogoutHandler();

logoutHandler.logout(VaadinServletRequest.getCurrent().getHttpServletRequest(),null,null);

}

}

**ContactForm.java**

package com.example.application.views.list;

import com.example.application.data.entity.Company;

import com.example.application.data.entity.Contact;

import com.example.application.data.entity.Status;

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.combobox.ComboBox;

import com.vaadin.flow.component.formlayout.FormLayout;

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.binder.ValidationException;

import com.vaadin.flow.shared.Registration;

import java.util.List;

public class ContactForm extends FormLayout {

Binder<Contact> binder = new BeanValidationBinder<>(Contact.class);

TextField firstName = new TextField("First name");

TextField lastName = new TextField("Last name");

EmailField email = new EmailField("Email");

ComboBox<Status> status = new ComboBox<>("Status");

ComboBox<Company> company = new ComboBox<>("Company");

Button save = new Button("Save");

Button delete = new Button("Delete");

Button close = new Button("Cancel");

private Contact contact;

public ContactForm(List<Company> companies, List<Status> statuses) {

addClassName("contact-form");

binder.bindInstanceFields(this);

company.setItems(companies);

company.setItemLabelGenerator(Company::getName);

status.setItems(statuses);

status.setItemLabelGenerator(Status::getName);

add(firstName,

lastName,

email,

company,

status,

createButtonsLayout());

}

public void setContact(Contact contact){

this.contact = contact;

binder.readBean(contact);

}

private HorizontalLayout createButtonsLayout() {

save.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);

delete.addThemeVariants(ButtonVariant.LUMO\_ERROR);

close.addThemeVariants(ButtonVariant.LUMO\_TERTIARY);

save.addClickListener(event ->validateAndSave());

delete.addClickListener(event -> fireEvent(new DeleteEvent(this,contact)));

close.addClickListener(event -> fireEvent(new CloseEvent(this)));

save.addClickShortcut(Key.ENTER);

close.addClickShortcut(Key.ESCAPE);

return new HorizontalLayout(save, delete, close);

}

private void validateAndSave() {

try {

binder.writeBean(contact);

fireEvent(new SaveEvent(this,contact));

} catch (ValidationException e) {

e.printStackTrace();

}

}

// Events Перегружаем дефолтные ивенты встроенной кнопки

public static abstract class ContactFormEvent extends ComponentEvent<ContactForm> {

private Contact contact;

protected ContactFormEvent(ContactForm source, Contact contact) {

super(source, false);

this.contact = contact;

}

public Contact getContact() {

return contact;

}

}

public static class SaveEvent extends ContactFormEvent {

SaveEvent(ContactForm source, Contact contact) {

super(source, contact);

}

}

public static class DeleteEvent extends ContactFormEvent {

DeleteEvent(ContactForm source, Contact contact) {

super(source, contact);

}

}

public static class CloseEvent extends ContactFormEvent {

CloseEvent(ContactForm source) {

super(source, null);

}

}

public <T extends ComponentEvent<?>> Registration addListener(Class<T> eventType,

ComponentEventListener<T> listener) {

return getEventBus().addListener(eventType, listener);

}

}

**ListView.java**

package com.example.application.views.list;

import com.example.application.data.entity.Contact;

import com.example.application.data.service.CrmService;

import com.vaadin.flow.component.Component;

import com.vaadin.flow.component.button.Button;

import com.vaadin.flow.component.grid.Grid;

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 javax.annotation.security.PermitAll;

import java.util.Collections;

@Route(value = "", layout = MainLayout.class)

@PageTitle("CourseWork | Vaadin CRM")

@PermitAll

public class ListView extends VerticalLayout {

Grid<Contact> grid = new Grid<>(Contact.class);

TextField filterText = new TextField();

ContactForm form;

private CrmService service;

public ListView(CrmService service) {

this.service = service;

addClassName("list-view");

setSizeFull();

configureGrid();

configureForm();

add(getToolbar(), getContent());

updateList();

closeEditor();

}

private void closeEditor() {

form.setContact(null);

form.setVisible(false);

removeClassName("editing");

}

private void updateList() {

grid.setItems(service.findAllContacts(filterText.getValue()));

}

private Component getContent() {

HorizontalLayout content = new HorizontalLayout(grid,form);

content.setFlexGrow(2,grid);

content.setFlexGrow(1, form);

content.addClassName("content");

content.setSizeFull();

return content;

}

private void configureForm() {

form = new ContactForm(service.findAllCompanies(),service.findAllStatuses());

form.setWidth("25em");

form.addListener(ContactForm.SaveEvent.class, this::saveContact);

form.addListener(ContactForm.DeleteEvent.class, this::deleteContact);

form.addListener(ContactForm.CloseEvent.class, e-> closeEditor());

}

private void saveContact(ContactForm.SaveEvent event){

service.saveContact(event.getContact());

updateList();

closeEditor();

}

private void deleteContact(ContactForm.DeleteEvent event){

service.deleteContact(event.getContact());

updateList();

closeEditor();

}

private void configureGrid() {

grid.addClassNames("contact-grid");

grid.setSizeFull();

grid.setColumns("firstName", "lastName", "email");

grid.addColumn(contact -> contact.getStatus().getName()).setHeader("Status");

grid.addColumn(contact -> contact.getCompany().getName()).setHeader("Company");

grid.getColumns().forEach(col -> col.setAutoWidth(true));

grid.asSingleSelect().addValueChangeListener(e -> editContact(e.getValue()));

}

private void editContact(Contact contact) {

if(contact == null){

closeEditor();

} else {

form.setContact(contact);

form.setVisible(true);

addClassName("editing");

}

}

private HorizontalLayout getToolbar() {

filterText.setPlaceholder("Filter by name...");

filterText.setClearButtonVisible(true);

filterText.setValueChangeMode(ValueChangeMode.LAZY);

filterText.addValueChangeListener(e -> updateList());

Button addContactButton = new Button("Add contact");

addContactButton.addClickListener(e -> addContact());

HorizontalLayout toolbar = new HorizontalLayout(filterText, addContactButton);

toolbar.addClassName("toolbar");

return toolbar;

}

private void addContact() {

grid.asSingleSelect().clear();

editContact(new Contact());

}

}

**LoginView.java**

package com.example.application.views.list;

import com.vaadin.flow.component.html.H1;

import com.vaadin.flow.component.login.LoginForm;

import com.vaadin.flow.component.orderedlayout.VerticalLayout;

import com.vaadin.flow.router.BeforeEnterEvent;

import com.vaadin.flow.router.BeforeEnterListener;

import com.vaadin.flow.router.PageTitle;

import com.vaadin.flow.router.Route;

@Route("login")

@PageTitle("Login | CourseWork")

public class LoginView extends VerticalLayout implements BeforeEnterListener {

private LoginForm login = new LoginForm();

public LoginView() {

addClassName("login-view");

setSizeFull();

setAlignItems(Alignment.CENTER);

setJustifyContentMode(JustifyContentMode.CENTER);

login.setAction("login");

add(

new H1("CourseWork"),login

);

}

@Override

public void beforeEnter(BeforeEnterEvent beforeEnterEvent) {

if (beforeEnterEvent.getLocation().getQueryParameters().getParameters().containsKey("error")){

login.setError(true);

}

}

}

**MainLayout.java**

package com.example.application.views.list;

import com.example.application.security.SecurityService;

import com.vaadin.flow.component.applayout.AppLayout;

import com.vaadin.flow.component.applayout.DrawerToggle;

import com.vaadin.flow.component.button.Button;

import com.vaadin.flow.component.html.H1;

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.router.HighlightCondition;

import com.vaadin.flow.router.HighlightConditions;

import com.vaadin.flow.router.RouterLink;

import java.awt.\*;

public class MainLayout extends AppLayout {

private SecurityService securityService;

public MainLayout(SecurityService securityService) {

this.securityService = securityService;

createHeader();

createDrawer();

}

private void createHeader() {

H1 logo = new H1("CourseWork");

logo.addClassNames("text-l", "m-m");

Button logOut = new Button("Log out", e -> securityService.logout());

HorizontalLayout header = new HorizontalLayout(new DrawerToggle(), logo, logOut);

header.setDefaultVerticalComponentAlignment(FlexComponent.Alignment.CENTER);

header.expand(logo);

header.setWidthFull();

header.addClassNames("py-0","px-m");

addToNavbar(header);

}

private void createDrawer(){

RouterLink listView = new RouterLink("List", ListView.class);

listView.setHighlightCondition(HighlightConditions.sameLocation());

addToDrawer( new VerticalLayout(

listView

));

}

}

**Application.java**

package com.example.application;

import com.vaadin.flow.component.dependency.NpmPackage;

import com.vaadin.flow.component.page.AppShellConfigurator;

import com.vaadin.flow.server.PWA;

import com.vaadin.flow.theme.Theme;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

/\*\*

\* The entry point of the Spring Boot application.

\*

\* Use the @PWA annotation make the application installable on phones, tablets

\* and some desktop browsers.

\*

\*/

@SpringBootApplication

@Theme(value = "flowcrmtutorial")

@PWA(name = "Flow CRM Tutorial", shortName = "Flow CRM Tutorial", offlineResources = {"images/logo.png"})

@NpmPackage(value = "line-awesome", version = "1.3.0")

public class Application extends SpringBootServletInitializer implements AppShellConfigurator {

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

}