# Міністерство освіти і науки, Молоді та спорту України Національний технічний університет «харківський політехнічний інститут»

кафедра Автоматизованих систем управління

Звіт до лабораторної роботи №1

з дисципліни «Компонентні програмні технології»

на тему:

«МОДЕЛЮВАННЯ КОМПОНЕНТ В CASE ЗАСОБАХ»

Виконав:

ст-ти гр. ІФ-30в:

Стрижов Д.В.

Пехуля Є. І.

Пантелеєв М. О.

Кудря В. А.

Перевірив:

Харків – 2012

## Цель работы:

Создать пакет необходимых диаграмм для заданной предметной области, созданных при помощи CASE-средств в нотации UML 2.0

Предметная область:

Почтовый сайт. Есть пользователи. У каждого пользователя есть следующие атрибуты: id, логин, пароль, ФИО, email, дата рожденья, роль (администратор, обычный пользователь и тп). Обычные пользователи могут просматривать/удалять свои письма (входящие и исходящие), отправлять письма любому пользователю. Администратор может просматривать/удалять письма любого пользователя. Письма делятся на категории: Входящие, Отправленные, Корзина.

## Ход работы:

1. Был изучен теоретический материал для выполнения работы
2. Были изучены методические указание по выполнению работы
3. В среде Visual Paradigm было создано диаграмму вариантов использования для заданной предметной области.

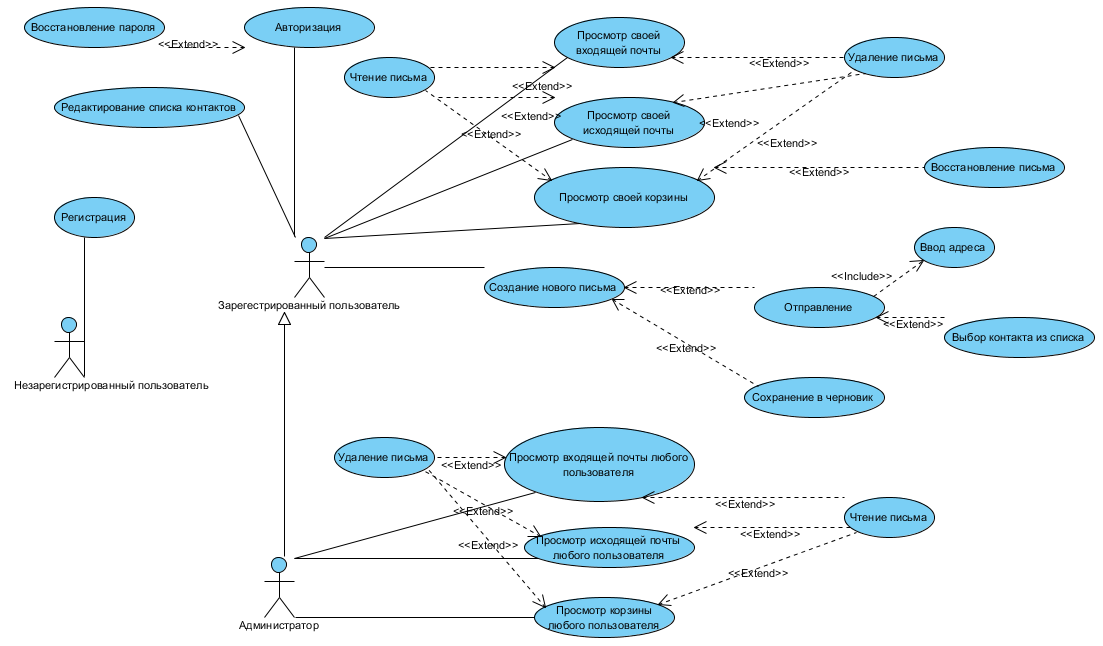


Рис. 1 – диаграмма вариантов использования

1. В соответствии с разработанной диаграммой вариантов использования, было созданно диаграммы последовательности для отображения взаимодействия основных объектов проектируемой системы.

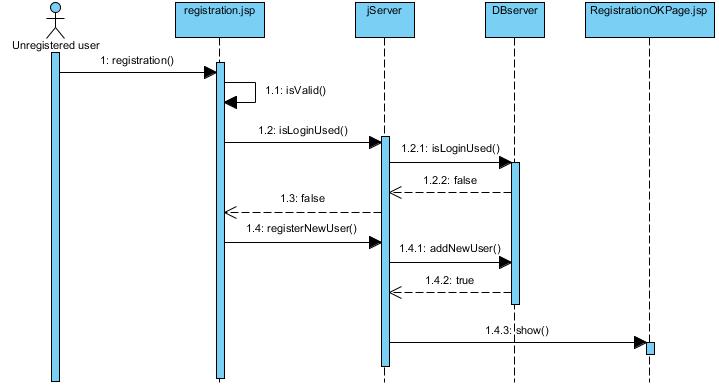


Рис. 2 - Диаграмма последовательности для варианта использования

«Регистрация»

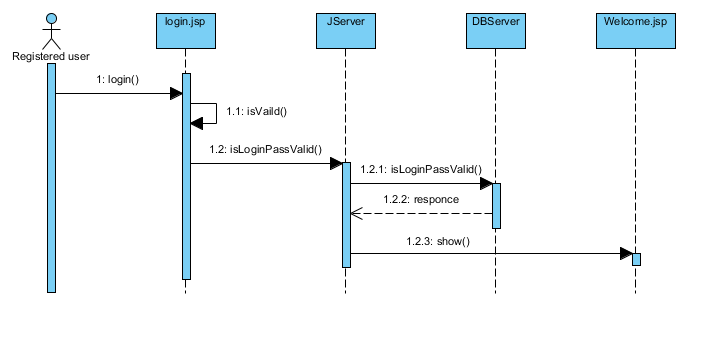


Рис. 3 - Диаграмма последовательности для варианта использования

«Авторизация»

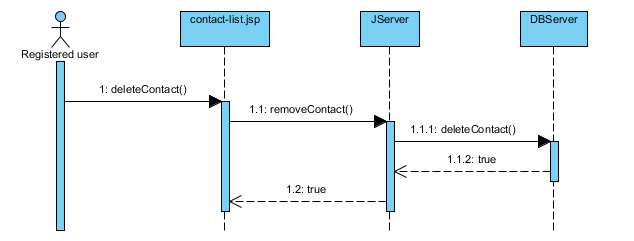


Рис. 4 - Диаграмма последовательности для варианта использования

«Удаление элемента из списка контактов»

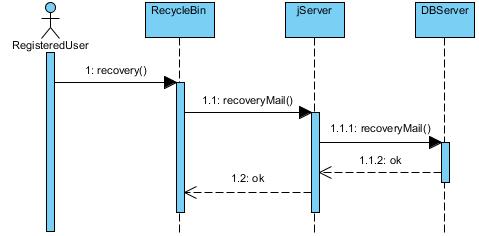


Рис. 5 - Диаграмма последовательности для варианта использования

«Восстановление пользователем письма из корзины»

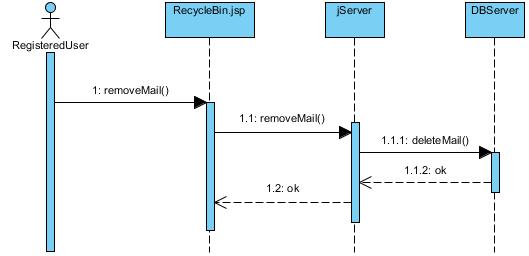


Рис. 6 - Диаграмма последовательности для варианта использования

«Удаление пользователем письма из корзины»

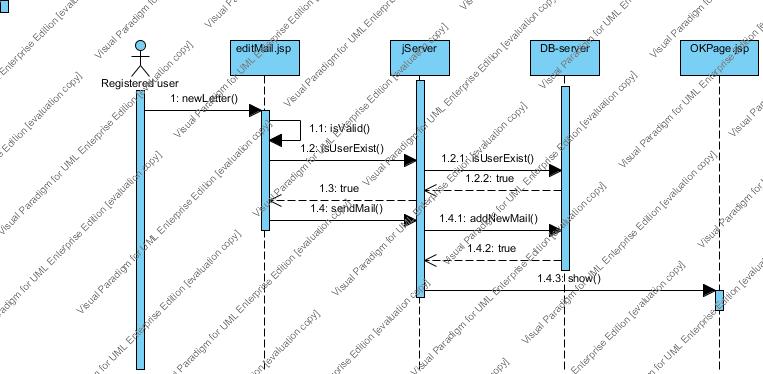


Рис. 7 - Диаграмма последовательности для варианта использования

«Отправка письма»

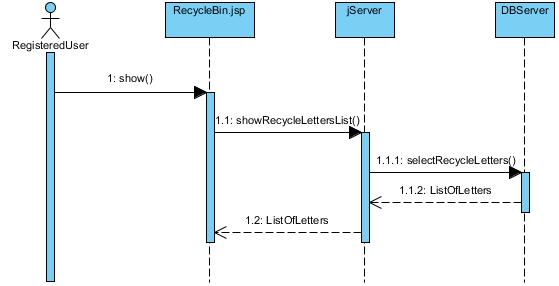


Рис. 8 - Диаграмма последовательности для варианта использования

«Просмотр своей корзины»

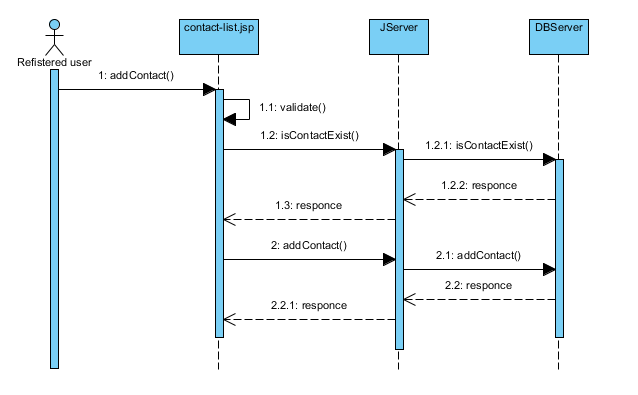


Рис. 9 - Диаграмма последовательности для варианта использования

«Добавление элемента в список контактов»

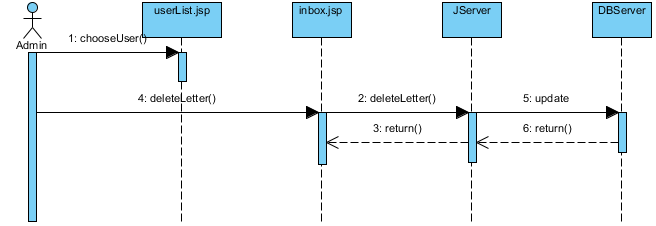


Рис. 10 - Диаграмма последовательности для варианта использования

«Удаление администратором входящего письма пользователя»

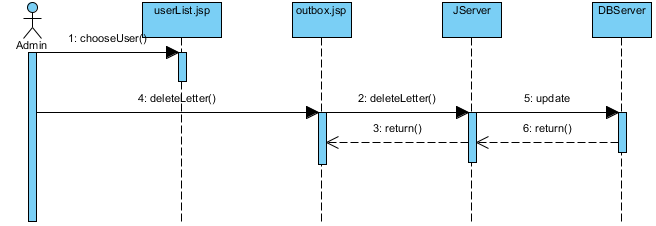


Рис. 11 - Диаграмма последовательности для варианта использования

«Удаление администратором исходящего письма пользователя»

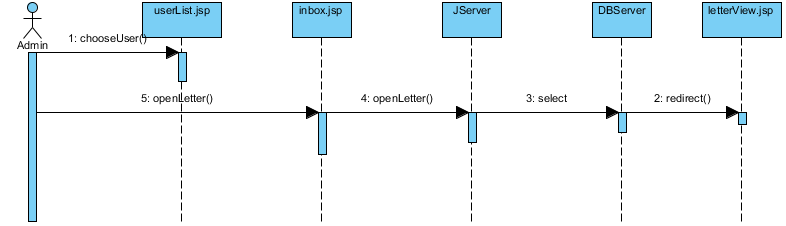


Рис. 12 - Диаграмма последовательности для варианта использования

«Чтение администратором входящего письма пользователя»

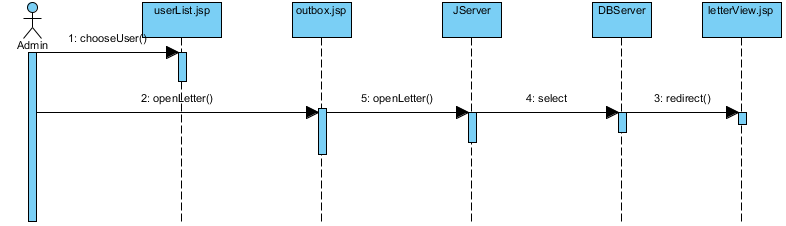


Рис. 13 - Диаграмма последовательности для варианта использования

«Чтение администратором исходящего письма пользователя»

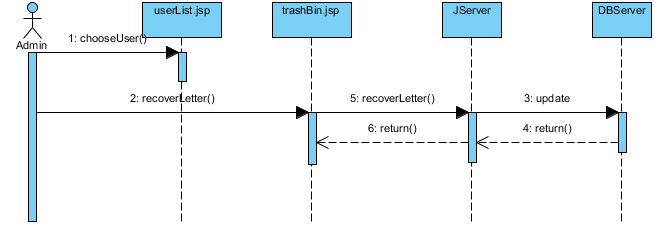


Рис. 14 - Диаграмма последовательности для варианта использования

«Восстановление администратором письма из корзины пользователя»

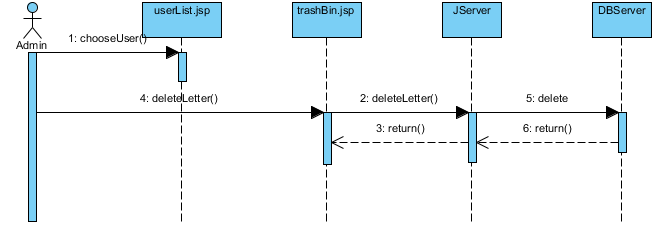


Рис. 15 - Диаграмма последовательности для варианта использования

«Удаление администратором письма из корзины пользователя»

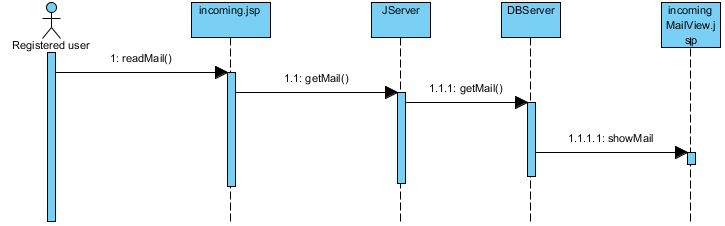


Рис. 16 - Диаграмма последовательности для варианта использования

«Чтение входящего письма»

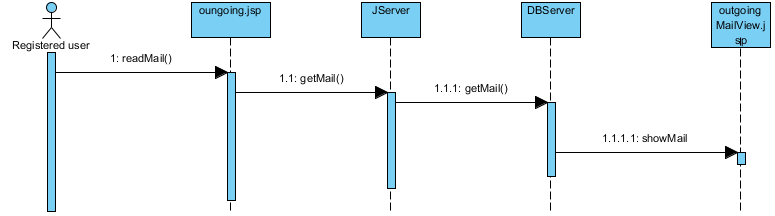


Рис. 17 - Диаграмма последовательности для варианта использования

«Чтение исходящего письма»

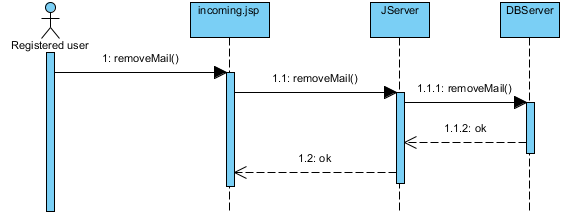


Рис. 18 - Диаграмма последовательности для варианта использования

«Удаление входящего письма»

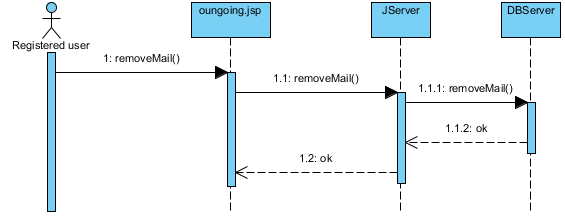


Рис. 19 - Диаграмма последовательности для варианта использования

«Удаление исходящего письма»

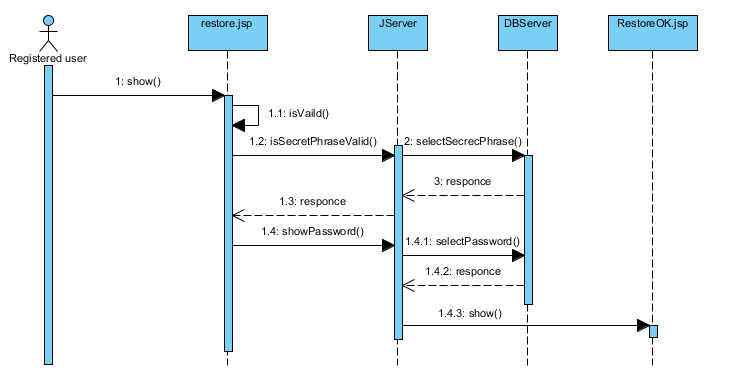


Рис. 20 - Диаграмма последовательности для варианта использования

«Восстановление пароля»

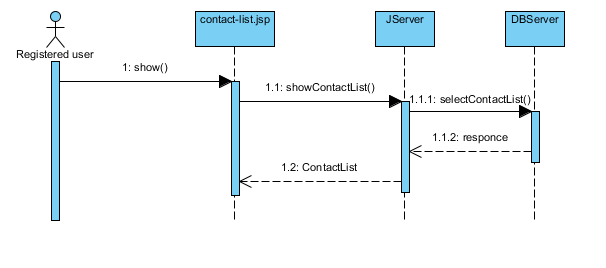


Рис. 21 - Диаграмма последовательности для варианта использования

«Просмотр списка контактов»

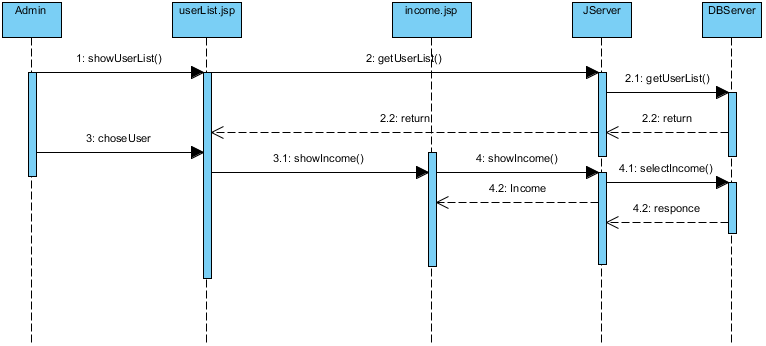


Рис. 22 - Диаграмма последовательности для варианта использования

«Просмотр администратором входящих писем пользователя»

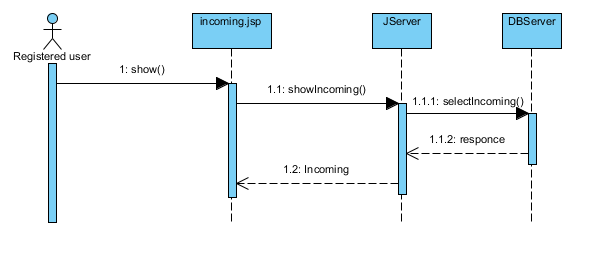


Рис. 23 - Диаграмма последовательности для варианта использования

«Просмотр входящих писем»

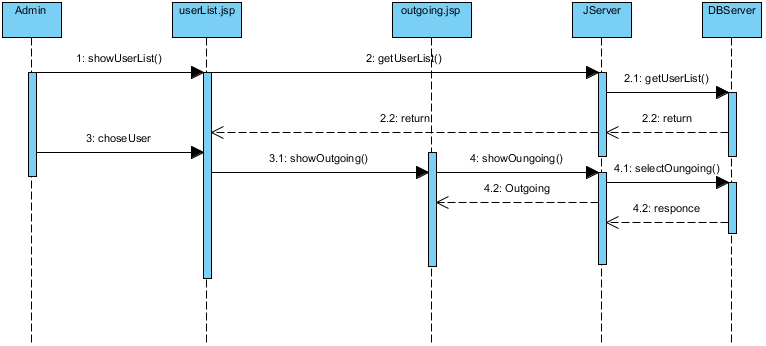


Рис. 24 - Диаграмма последовательности для варианта использования

«Просмотр администратором исходящих пользователя»

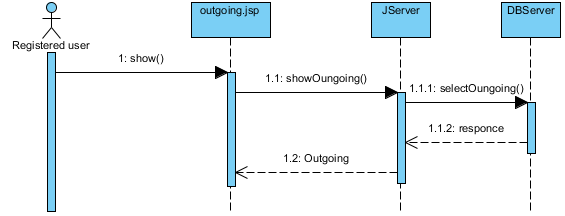


Рис. 25 - Диаграмма последовательности для варианта использования

«Просмотр исходящих писем»

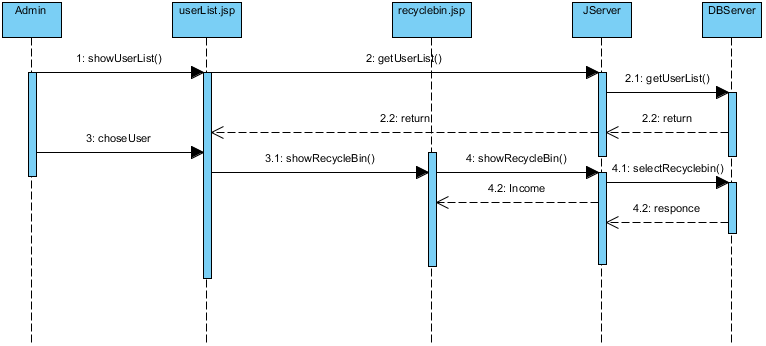


Рис. 26 - Диаграмма последовательности для варианта использования

«Просмотр администратором корзины пользователя»

1. Была разработана концептуальную модель данных, реляционную модель данных, и диаграмму таблиц базы данных, необходимой для содержания данных о предметной области

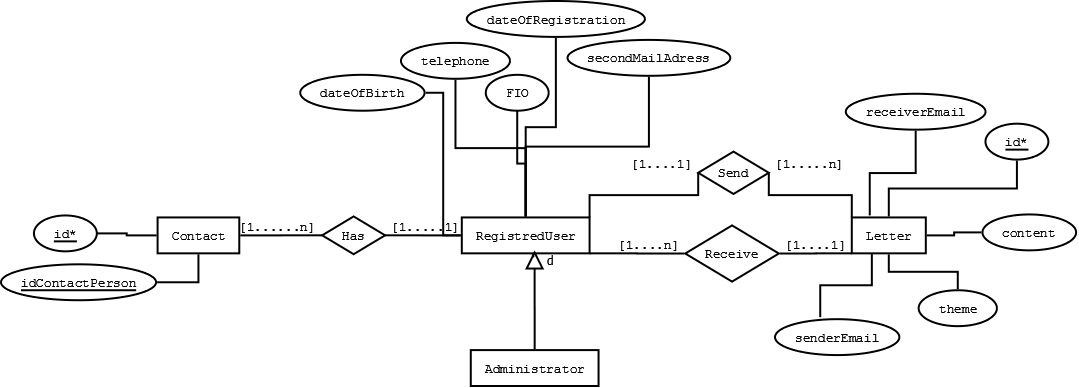


Рис. 27 – Концептуальная модель данных в нотации EER

Реляционная модель данных имеет следующий вид:

**R1 RegistredUser** (id\*, FIO, telephone dateOfBirth, dateOfRegistration, secondMailAdress);

**R2 Administrator** (id\*, idRegistredUser);

**R3 Letter** (id\*, receiverEmail, senderEmail, content, theme);

**R4 Letter\_Sender\_Receiver** (id\*, idSender, idReceiver, idLetter, isRead, isSenderTrash, isRecTrash);

**R5 Contact** (id\*, idContactHolder, idContactPerson).

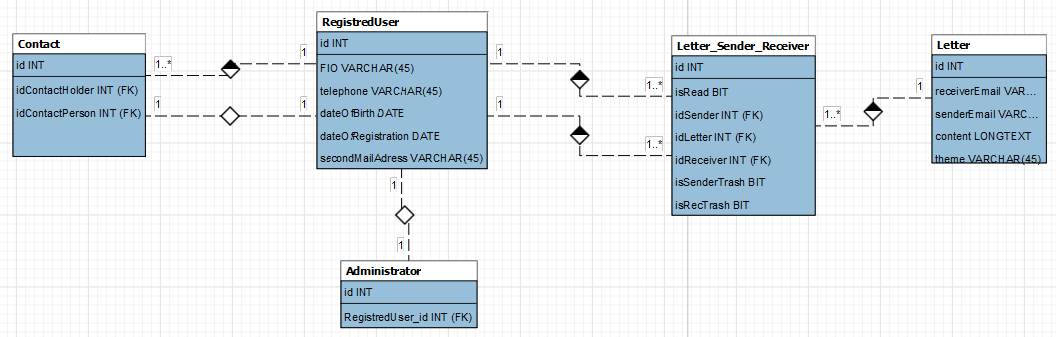


Рис. 28 – Диаграмма таблиц БД в нотации IDEF1X

1. При помощи автоматизированных утилит CASE-средства Visual Paradigm, по составленным диаграммам было сгенерировано sql-код для создания базы данных.

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL,ALLOW\_INVALID\_DATES';

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET latin1 COLLATE latin1\_swedish\_ci ;

USE `mydb` ;

-- -----------------------------------------------------

-- Table `mydb`.`RegistredUser`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`RegistredUser` ;

CREATE TABLE IF NOT EXISTS `mydb`.`RegistredUser` (

`id` INT NOT NULL AUTO\_INCREMENT ,

`FIO` VARCHAR(45) NULL ,

`telephone` VARCHAR(45) NULL ,

`dateOfBirth` DATE NULL ,

`dateOfRegistration` DATE NULL ,

`secondMailAdress` VARCHAR(45) NULL ,

PRIMARY KEY (`id`) )

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Administrator`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Administrator` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Administrator` (

`id` INT NOT NULL AUTO\_INCREMENT ,

`RegistredUser\_id` INT NOT NULL ,

PRIMARY KEY (`id`) ,

INDEX `fk\_Administrator\_RegistredUser1\_idx` (`RegistredUser\_id` ASC) ,

CONSTRAINT `fk\_Administrator\_RegistredUser1`

FOREIGN KEY (`RegistredUser\_id` )

REFERENCES `mydb`.`RegistredUser` (`id` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Letter`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Letter` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Letter` (

`id` INT NOT NULL AUTO\_INCREMENT ,

`receiverEmail` VARCHAR(45) NULL ,

`senderEmail` VARCHAR(45) NULL ,

`content` LONGTEXT NULL ,

`theme` VARCHAR(45) NULL ,

`date&time` DATETIME NULL ,

PRIMARY KEY (`id`) )

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Letter\_Sender\_Receiver`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Letter\_Sender\_Receiver` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Letter\_Sender\_Receiver` (

`id` INT NOT NULL AUTO\_INCREMENT ,

`isRead` BIT NULL DEFAULT 0 ,

`idSender` INT NOT NULL ,

`idLetter` INT NOT NULL ,

`idReceiver` INT NOT NULL ,

`isSenderTrash` BIT NULL DEFAULT 0 ,

`isRecTrash` BIT NULL DEFAULT 0 ,

PRIMARY KEY (`id`) ,

INDEX `fk\_Letter\_Sender\_Receiver\_RegistredUser1\_idx` (`idSender` ASC) ,

INDEX `fk\_Letter\_Sender\_Receiver\_Letter1\_idx` (`idLetter` ASC) ,

INDEX `fk\_Letter\_Sender\_Receiver\_RegistredUser2\_idx` (`idReceiver` ASC) ,

CONSTRAINT `fk\_Letter\_Sender\_Receiver\_RegistredUser1`

FOREIGN KEY (`idSender` )

REFERENCES `mydb`.`RegistredUser` (`id` )

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Letter\_Sender\_Receiver\_Letter1`

FOREIGN KEY (`idLetter` )

REFERENCES `mydb`.`Letter` (`id` )

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Letter\_Sender\_Receiver\_RegistredUser2`

FOREIGN KEY (`idReceiver` )

REFERENCES `mydb`.`RegistredUser` (`id` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Contact`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Contact` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Contact` (

`id` INT NOT NULL AUTO\_INCREMENT ,

`idContactHolder` INT NOT NULL ,

`idContactPerson` INT NOT NULL ,

PRIMARY KEY (`id`) ,

INDEX `fk\_Contact\_RegistredUser\_idx` (`idContactHolder` ASC) ,

INDEX `fk\_Contact\_RegistredUser1\_idx` (`idContactPerson` ASC) ,

CONSTRAINT `fk\_Contact\_RegistredUser`

FOREIGN KEY (`idContactHolder` )

REFERENCES `mydb`.`RegistredUser` (`id` )

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Contact\_RegistredUser1`

FOREIGN KEY (`idContactPerson` )

REFERENCES `mydb`.`RegistredUser` (`id` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

1. Были автоматически сгенерированы java-классы по разработанной диаграмме

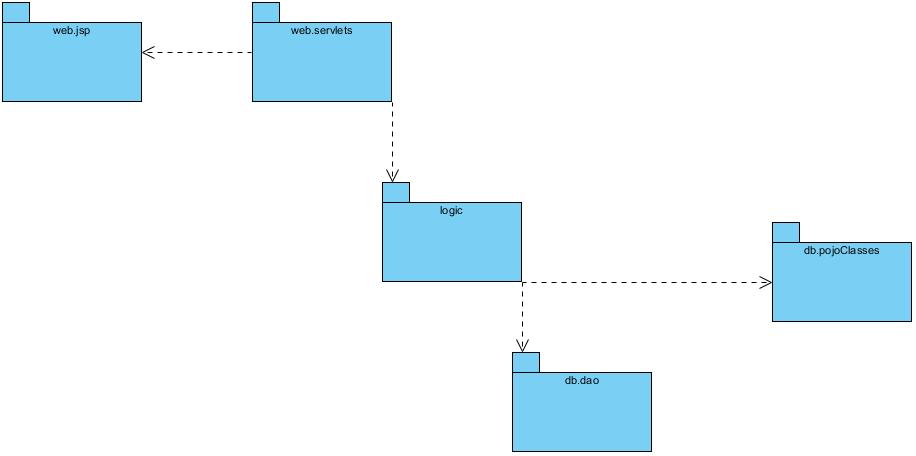


Рис. 29 – Диаграмма пакетов

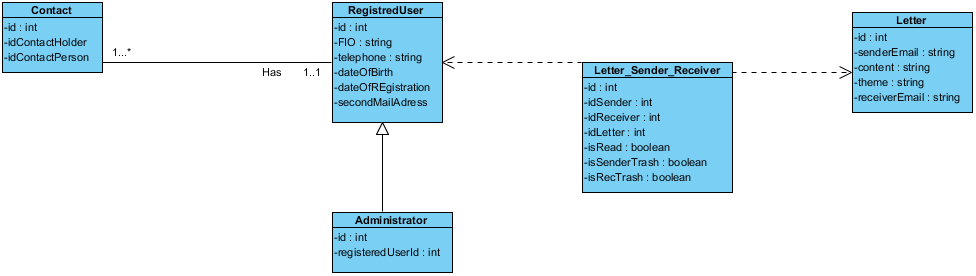


Рис. 30 – Диаграмма классов

Сгенерированный Java-код из диаграммы классов с помощью case – средства Visual Paradigm for UML:

package logic;

public class Administrator extends RegistredUser {

private int \_id;

private RegistredUser \_registeredUserId;

public Administrator() {

super();

// TODO Auto-generated constructor stub

}

public int get\_id() {

return \_id;

}

public void set\_id(int \_id) {

this.\_id = \_id;

}

public RegistredUser get\_registeredUserId() {

return \_registeredUserId;

}

public void set\_registeredUserId(RegistredUser \_registeredUserId) {

this.\_registeredUserId = \_registeredUserId;

}

}

package logic;

public class Contact {

private int \_id;

private RegistredUser \_ContactHolder;

private RegistredUser \_ContactPerson;

public Contact() {

super();

// TODO Auto-generated constructor stub

}

public int get\_id() {

return \_id;

}

public void set\_id(int \_id) {

this.\_id = \_id;

}

public RegistredUser get\_ContactHolder() {

return \_ContactHolder;

}

public void set\_ContactHolder(RegistredUser \_ContactHolder) {

this.\_ContactHolder = \_ContactHolder;

}

public RegistredUser get\_ContactPerson() {

return \_ContactPerson;

}

public void set\_ContactPerson(RegistredUser \_ContactPerson) {

this.\_ContactPerson = \_ContactPerson;

}

}

package logic;

public class Letter\_Sender\_Receiver {

private int \_id;

private RegistredUser \_Sender;

private RegistredUser \_Receiver;

private Letter \_Letter;

private boolean \_isRead;

private boolean \_isSenderTrash;

private boolean \_isRecTrash;

public Letter\_Sender\_Receiver() {

super();

// TODO Auto-generated constructor stub

}

public int get\_id() {

return \_id;

}

public void set\_id(int \_id) {

this.\_id = \_id;

}

public RegistredUser get\_Sender() {

return \_Sender;

}

public void set\_Sender(RegistredUser \_Sender) {

this.\_Sender = \_Sender;

}

public RegistredUser get\_Receiver() {

return \_Receiver;

}

public void set\_Receiver(RegistredUser \_Receiver) {

this.\_Receiver = \_Receiver;

}

public Letter get\_Letter() {

return \_Letter;

}

public void set\_Letter(Letter \_Letter) {

this.\_Letter = \_Letter;

}

public boolean is\_isRead() {

return \_isRead;

}

public void set\_isRead(boolean \_isRead) {

this.\_isRead = \_isRead;

}

public boolean is\_isSenderTrash() {

return \_isSenderTrash;

}

public void set\_isSenderTrash(boolean \_isSenderTrash) {

this.\_isSenderTrash = \_isSenderTrash;

}

public boolean is\_isRecTrash() {

return \_isRecTrash;

}

public void set\_isRecTrash(boolean \_isRecTrash) {

this.\_isRecTrash = \_isRecTrash;

}

}

package logic;

public class Letter {

private int \_id;

private String \_senderEmail;

private String \_content;

private String \_theme;

private String \_receiverEmail;

public Letter() {

super();

// TODO Auto-generated constructor stub

}

public int get\_id() {

return \_id;

}

public void set\_id(int \_id) {

this.\_id = \_id;

}

public String get\_senderEmail() {

return \_senderEmail;

}

public void set\_senderEmail(String \_senderEmail) {

this.\_senderEmail = \_senderEmail;

}

public String get\_content() {

return \_content;

}

public void set\_content(String \_content) {

this.\_content = \_content;

}

public String get\_theme() {

return \_theme;

}

public void set\_theme(String \_theme) {

this.\_theme = \_theme;

}

public String get\_receiverEmail() {

return \_receiverEmail;

}

public void set\_receiverEmail(String \_receiverEmail) {

this.\_receiverEmail = \_receiverEmail;

}

}

package logic;

import java.util.Date;

public class RegistredUser {

private int \_id;

private String \_fIO;

private String \_telephone;

private Date \_dateOfBirth;

private Date \_dateOfREgistration;

private String \_secondMailAdress;

public RegistredUser() {

super();

// TODO Auto-generated constructor stub

}

public int get\_id() {

return \_id;

}

public void set\_id(int \_id) {

this.\_id = \_id;

}

public String get\_fIO() {

return \_fIO;

}

public void set\_fIO(String \_fIO) {

this.\_fIO = \_fIO;

}

public String get\_telephone() {

return \_telephone;

}

public void set\_telephone(String \_telephone) {

this.\_telephone = \_telephone;

}

public Date get\_dateOfBirth() {

return \_dateOfBirth;

}

public void set\_dateOfBirth(Date \_dateOfBirth) {

this.\_dateOfBirth = \_dateOfBirth;

}

public Date get\_dateOfREgistration() {

return \_dateOfREgistration;

}

public void set\_dateOfREgistration(Date \_dateOfREgistration) {

this.\_dateOfREgistration = \_dateOfREgistration;

}

public String get\_secondMailAdress() {

return \_secondMailAdress;

}

public void set\_secondMailAdress(String \_secondMailAdress) {

this.\_secondMailAdress = \_secondMailAdress;

}

}

# ВЫВОД

В ходе выполнения лабораторной роботы, были изучены возможности проектирования программного обеспечения при помощи CASE-средства Visual Paradigm для заданной предметной области «Почтовый сайт»