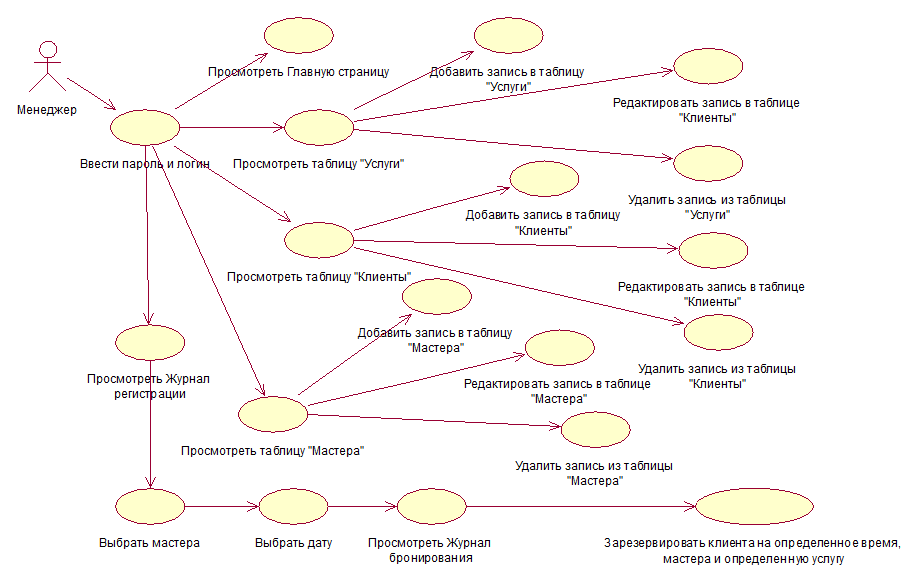
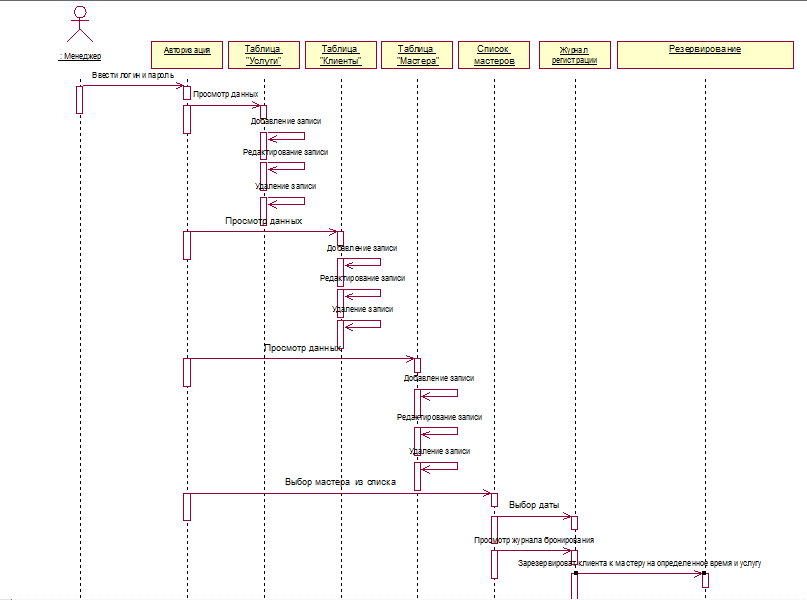
# Приложение A

# Диаграмма вариантов использования



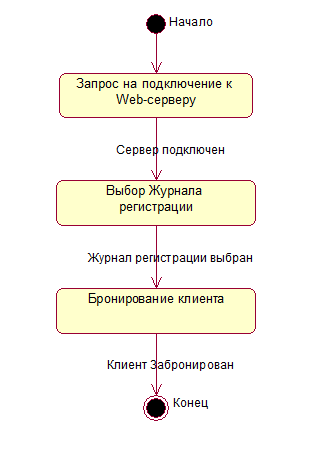
# Приложение Б

# Диаграмма последовательности



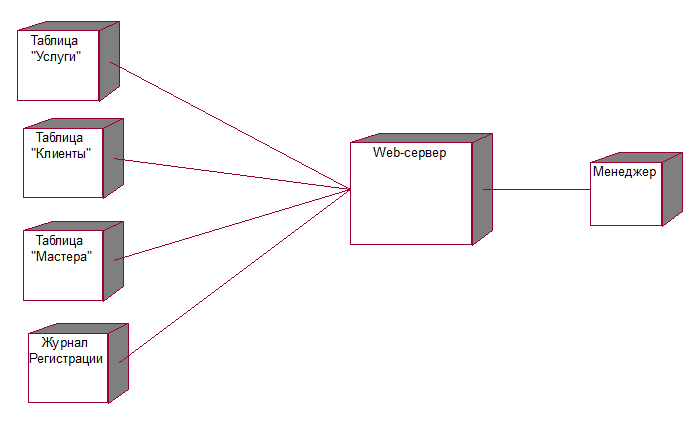
# Приложение В

# Диаграмма состояния



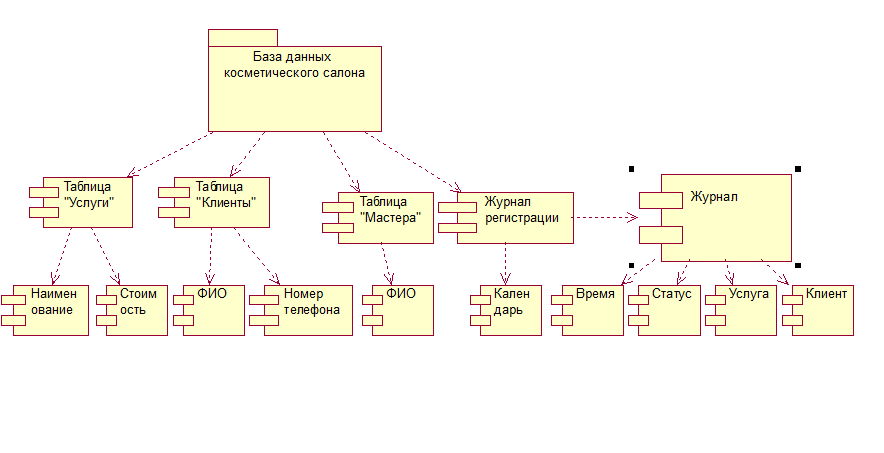
# Приложение Г

# Диаграмма развертывания



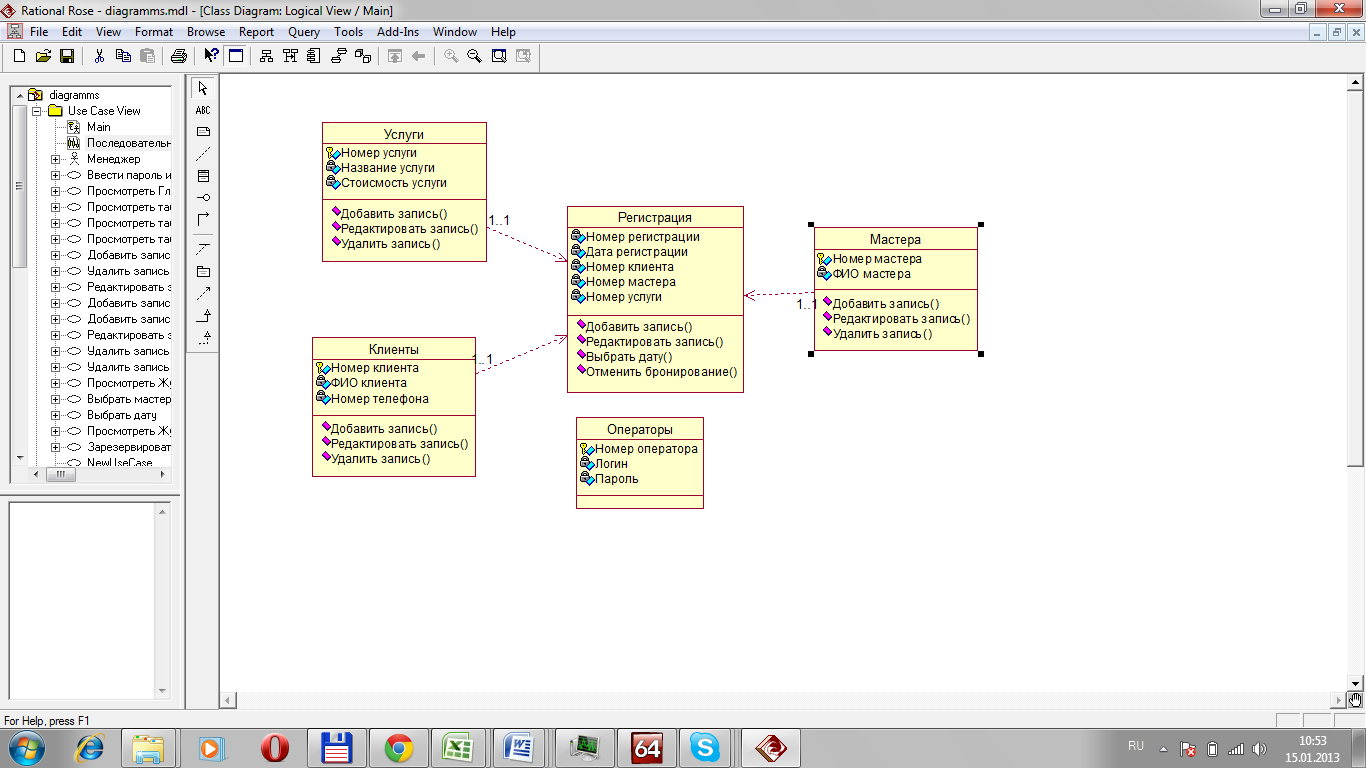
# Приложение Д

# Диаграмма компонентов



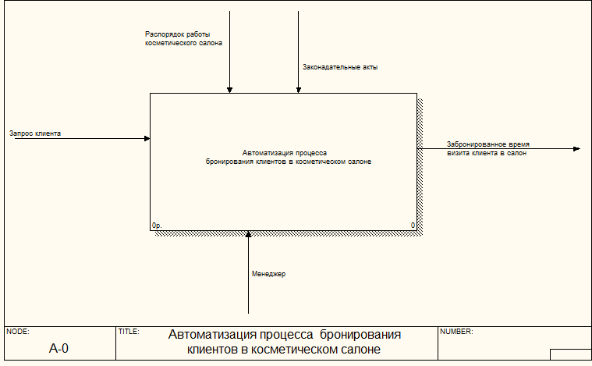
# Приложение Е

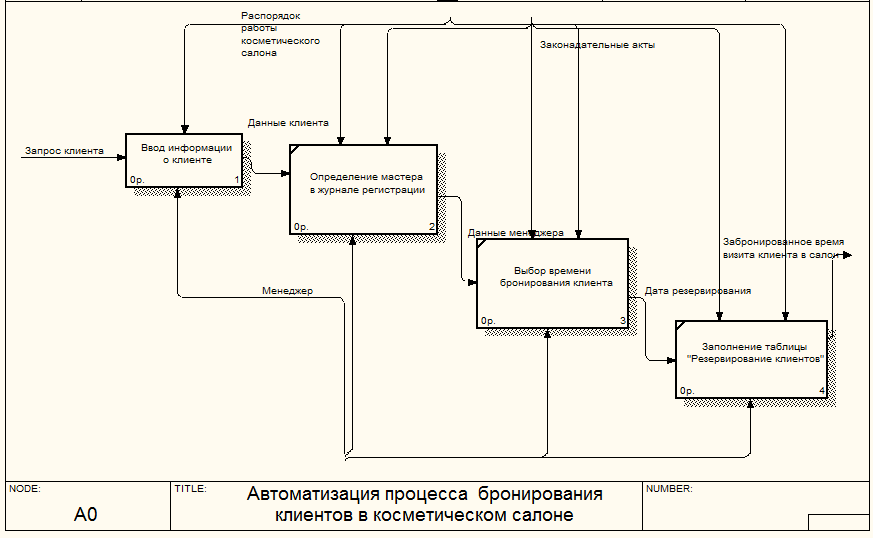
# Диаграмма классов

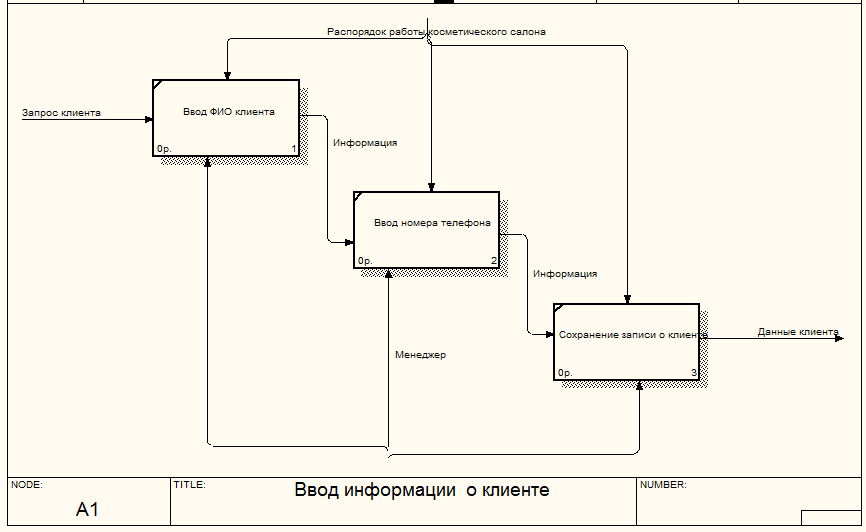


# Приложение Ж

# Диаграммы методологии IDEF0







# Приложение З

# Листинг программы

//RegisterController.java

package ru.data.controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.\*;

import ru.data.model.bo.\*;

import ru.data.servicies.ClientService;

import ru.data.servicies.MasterService;

import ru.data.servicies.PriceService;

import ru.data.servicies.RegisterService;

import java.io.UnsupportedEncodingException;

import java.sql.SQLException;

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import java.util.Map;

@Controller

public class RegisterController {

@Autowired

private MasterService masterService;

@Autowired

private RegisterService registerService;

@Autowired

private PriceService priceService;

@Autowired

private ClientService clientService;

@RequestMapping(value = "/register")

public String register(Map<String, Object> map) throws SQLException {

map.put("masterList", masterService.getAllMaster());

return "register";

}

@RequestMapping(value = "/register/{masterId}")

public String registerByMaster(@PathVariable("masterId") Long id, Map<String, Object> map) throws SQLException, ParseException {

map.put("viewsList", getViews(id, null));

map.put("master", masterService.getMasterById(id));

return "register";

}

@RequestMapping(value = "/showreg")

public String registerByMasterView(

@RequestParam(value = "id") Long id,

@RequestParam(value = "date") String date,

Map<String, Object> map) throws SQLException, ParseException {

map.put("viewsList", getViews(id, date));

map.put("master", masterService.getMasterById(id));

return "register";

}

@RequestMapping(value = "/reserve")

public String reserve(

@RequestParam(value = "resId") Long id,

@RequestParam(value = "resDate") String date,

@RequestParam(value = "resHour") Integer hour,

Map<String, Object> map) throws SQLException, ParseException{

Date fullDate = getDate(date, hour);

map.put("fullDate", convertFullDateToStr(fullDate));

map.put("master", masterService.getMasterById(id));

map.put("clientsList", clientService.getAllClients());

map.put("priceList", priceService.getAllPrices());

map.put("register", new RegisterCreation());

return "reserve";

}

@RequestMapping(value = "/saveReg", method = RequestMethod.POST)

public String saveReg(

@ModelAttribute("register") RegisterCreation register,

BindingResult result) throws SQLException, UnsupportedEncodingException, ParseException {

Register reg = new Register();

reg.setMasterId(register.getMasterId());

reg.setClientId(register.getClientId());

reg.setDate(convertFullDateToDate(register.getDate()));

reg.setPriceId(register.getPriceId());

registerService.addRegister(reg);

return "redirect:/cosmSalon/register/"+register.getMasterId().toString();

}

@RequestMapping(value = "/delRegister")

public String deleteClient(

@RequestParam(value = "regId") Long regId) throws SQLException {

Register reg = registerService.getRegisterById(regId);

Long masterId = reg.getMasterId();

registerService.deleteRegister(reg);

return "redirect:/cosmSalon/register/"+masterId;

}

private ArrayList<RegisterView> getViews(Long id, String date) throws SQLException, ParseException{

Date fDate = new Date();

fDate.setHours(0);

fDate.setMinutes(0);

fDate.setSeconds(0);

if(date!=null)

fDate = convertToDate(date);

ArrayList<RegisterView> viewList = new ArrayList<RegisterView>();

ArrayList<Register> list = registerService.getRecordsByMasterId(id);

Date iterDate = fDate;

for(int i = 9; i < 18; i++){

RegisterView view = new RegisterView();

iterDate.setHours(i);

view.setDate(iterDate);

view.setDateStr(convertTimeToStr(iterDate));

view.setHour(i);

boolean setted = false;

for(Register reg: list){

if(convertFullDateToStr(reg.getDate()).equals(convertFullDateToStr(iterDate))){

Clients client = clientService.getClientById(reg.getClientId());

view.setClientName(client.getLastName()+" "+client.getName());

view.setClientId(client.getId());

Price price = priceService.getPriceById(reg.getPriceId());

view.setCost(price.getCost());

view.setPriceName(price.getName());

view.setRegId(reg.getId());

view.setStatus("Забронировано");

setted = true;

break;

}

}

if(!setted)

view.setStatus("Свободно");

viewList.add(view);

}

return viewList;

}

private Date convertToDate(String date) throws ParseException{

DateFormat format = new SimpleDateFormat("dd-MM-yyyy");

return format.parse(date);

}

private String convertFullDateToStr(Date date){

DateFormat format = new SimpleDateFormat("dd-MM-yyyy HH:mm");

return format.format(date);

}

private Date convertFullDateToDate(String date) throws ParseException{

DateFormat format = new SimpleDateFormat("dd-MM-yyyy HH:mm");

return format.parse(date);

}

private String convertToStr(Date date){

DateFormat format = new SimpleDateFormat("dd-MM-yyyy");

return format.format(date);

}

private String convertTimeToStr(Date date){

DateFormat format = new SimpleDateFormat("HH:mm");

return format.format(date);

}

private Date getDate(String date, Integer hour) throws ParseException{

Date res = convertToDate(date);

res.setHours(hour);

return res;

}

}

//Encoder.java

package ru.data.servicies;

import ru.data.model.bo.Clients;

import ru.data.model.bo.Master;

import ru.data.model.bo.Price;

import java.io.UnsupportedEncodingException;

public class Encoder {

public static Clients clientEncoder(Clients client) throws UnsupportedEncodingException {

if(client!=null)

{

client.setPhoneNum(strEncoding(client.getPhoneNum()));

client.setLastName(strEncoding(client.getLastName()));

client.setName(strEncoding(client.getName()));

client.setSurName(strEncoding(client.getSurName()));

}

return client;

}

public static Master masterEncoder(Master master) throws UnsupportedEncodingException {

if(master!=null)

{

master.setLastName(strEncoding(master.getLastName()));

master.setName(strEncoding(master.getName()));

master.setSurName(strEncoding(master.getSurName()));

}

return master;

}

public static String strEncoding(String str) throws UnsupportedEncodingException {

String newStr = null;

if (str != null) {

newStr = new String(str.getBytes("ISO-8859-1"), "UTF-8");

}

return newStr;

}

public static Price priceEncoder(Price price) throws UnsupportedEncodingException {

if(price!=null)

{

price.setName(strEncoding(price.getName()));

}

return price;

}

}

//HelloTag.java

package ru.data.tags;

import ru.data.servicies.Settings;

import javax.servlet.jsp.tagext.\*;

import javax.servlet.jsp.\*;

import java.io.\*;

import java.util.Date;

public class HelloTag extends SimpleTagSupport {

public void doTag() throws JspException, IOException {

JspWriter out = getJspContext().getOut();

out.println("Оператор: "+ Settings.user);

}

}

//MasterDAOImpl.java

package ru.data.DAO;

import org.hibernate.SessionFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Repository;

import ru.data.model.bo.Master;

import java.sql.SQLException;

import java.util.ArrayList;

@Repository

public class MasterDAOImpl implements MasterDAO {

@Autowired

private SessionFactory sessionFactory;

@Override

public void addMaster(Master master) throws SQLException {

sessionFactory.getCurrentSession().save(master);

}

@Override

public void updateMaster(Master master) throws SQLException {

if(null!= master)

{

sessionFactory.getCurrentSession().clear();

sessionFactory.getCurrentSession().update(master);

}

}

@Override

public void deleteMaster(Long id) throws SQLException {

Master master = (Master) sessionFactory.getCurrentSession().load(Master.class, id);

if (null != master) {

sessionFactory.getCurrentSession().delete(master);

}

}

@Override

public ArrayList<Master> getAllMaster() throws SQLException {

return (ArrayList<Master>) sessionFactory.getCurrentSession().createQuery("from Master").list();

}

@Override

public Master getMasterById(Long id) throws SQLException {

Master master = new Master();

String query = "from Master where id="+id.toString();

ArrayList<Master> list = (ArrayList<Master>) sessionFactory.getCurrentSession().createQuery(query).list();

try

{

master=list.get(0);

}

catch (IndexOutOfBoundsException e)

{

master = null;

}

return master;

}

}