Servlet Reserve

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        String path = (String) request.getPathInfo().substring(1);
        Usuario user = (Usuario) request.getSession().getAttribute("user");
        if(user!=null) {
                try {
                        switch (path) {
                        case "start": {
                                LinkedList<ComputersSpecification> pcs = this.ctrl.GetPcsAvailable();
                                request.setAttribute("pcs", pcs);
                                request.getRequestDispatcher("/WEB-INF/Views/Reserve/reservation.jsp").forward(request, response);
                                break;
                        }
                        case "cancel": {
                                Reserva r = (Reserva) request.getSession().getAttribute("reserva");
                                request.getSession().removeAttribute("forUser");
                                request.getSession().removeAttribute("reserva");
                                request.getSession().removeAttribute("para");
                                request.getSession().removeAttribute("pc");
                                this.ctrl.changeState(r.getIdComputadora(), "disponible");
                                response.sendRedirect("../bookings.jsp");
                        }
                        default:
                        }
                } catch (IllegalStateException e) {
                        response.sendRedirect("../login.jsp");
                }
        } else {
                response.sendRedirect("../login.jsp");
        }
}
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        String path = (String) request.getPathInfo().substring(1);
        Usuario user = (Usuario) request.getSession().getAttribute("user");
        if(user!=null) {
                        switch (path) {
                        case "selected": {
```

```java
				Reserva reserve = new Reserva();
				String dia = (String) request.getParameter("reserva_para");
				String type = (String) request.getParameter("tipo");

				int idpc = this.ctrl.selectToReserve(type);

				reserve.setIdComputadora(idpc);
				reserve.setIdUsuario(user.getId());

				reserve.setFecha_de_reserva(LocalDate.now());
				if(dia.contains("mañana")&&LocalTime.now().getHour()!=0) {
					reserve.setFecha_a_reservar(LocalDate.now().plusDays(1));
				} else {
					reserve.setFecha_a_reservar(LocalDate.now());
				}
				request.getSession().setAttribute("para", dia);
				request.getSession().setAttribute("pc", type);
				request.getSession().setAttribute("reserva", reserve);
				request.getRequestDispatcher("/WEB-INF/Views/Reserve/saving.jsp").forward(request, response);
				break;
			}
			case "resume" : {
				String hdesde = (String) request.getParameter("horadesde");
				String hhasta = (String) request.getParameter("horahasta");

				if(hdesde.equals("Desde") || hhasta.equals("Hasta") || LocalTime.parse(hdesde).getHour()>LocalTime.parse(hhasta).getHour()) {
					request.setAttribute("error", "Por favor, especifique correctamente las horas");
					request.getRequestDispatcher("/WEB-INF/Views/Reserve/saving.jsp").include(request, response);
				} else {

					Reserva reserve = (Reserva) request.getSession().getAttribute("reserva");

					reserve.setHoraDesde(LocalTime.parse(hdesde));
					reserve.setHoraHasta(LocalTime.parse(hhasta));

					int price = this.ctrl.obtenerPrecioAlDia((String)request.getSession().getAttribute("pc"));
					request.setAttribute("precio", price);

					int monto = this.ctrl.calcularMonto(reserve.getHoraDesde(), reserve.getHoraHasta(), price);
					reserve.setImporte(monto);
```

```
                    if(completeReserveByType(request, reserve)==null) {
                        request.getRequestDispatcher("/WEB-INF/Views/Reserve/saving.jsp").include(request, response);
                    } else {

                        request.getRequestDispatcher("/WEB-INF/Views/Reserve/resume.jsp").forward(request, response);
                    }
                }
                break;
            }
            case "save" : {
                Reserva reserve = (Reserva) request.getSession().getAttribute("reserva");
                reserve.setEstado("solicitada");
                try {
                    this.ctrl.save(reserve);
                    this.ctrl.sendMail(user, reserve, (String)request.getSession().getAttribute("pc"));
                    request.getRequestDispatcher("/WEB-INF/Views/Reserve/success.jsp").forward(request, response);
                } catch (AddressException e) {
                    System.out.println("address exception");
                    e.printStackTrace();
                } catch (MessagingException e) {
                    System.out.println("messaging excepction");
                    e.printStackTrace();
                } catch (SQLIntegrityConstraintViolationException e1) {
                    e1.printStackTrace();
                    request.setAttribute("error", "Ya realizo una reserva.");
                    this.ctrl.changeState(reserve.getIdComputadora(), "disponible");
                    response.sendError(400);
                }
                break;
            }
            default: {
            }
            }
    } else {
        response.sendRedirect("../login.jsp");
    }
}
```

ControladorReserva

```
public class ControladorReserva {
        private DataPc pcdao;
        private DataReservas rdao;
        private DataPrecios pdao;
        private DataDescuentos ddao;
        private DataUsuarios userdao;

public ControladorReserva() {
        this.pcdao = new DataPc();
        this.rdao = new DataReservas();
        this.pdao = new DataPrecios();
        this.ddao = new DataDescuentos();
        this.userdao = new DataUsuarios();
}
public boolean finish(String code) {

        return rdao.finish(code);
}
public LinkedList<Streamers> getStreamersList() {

        return rdao.getStreamersList();
}
public LinkedList<ReserveList> getAll() {

        return rdao.getAll();
}
public ReserveSpecification cancelarReserva(String code) {

        return rdao.cancel(code);
}
public Usuario getUserByUsername(String username) {

        return userdao.getByUsername(username);
}
public LinkedList<ComputersSpecification> GetPcsAvailable () {

        return pcdao.GetPcsAvailable();
}
public int selectToReserve(String tpc) {

        int id = pcdao.getIdAvailable(tpc);
        changeState(id, "seleccionada");
        return id;
}
public void changeState(int id, String estado) {

        pcdao.setEstado(id, estado);
}
public int obtenerPrecioAlDia(String tpc) {

        return pdao.getPrice(tpc);
}
public Descuento obtenerDescuento(int cantHoras) {

        return ddao.getOne(cantHoras);
}
public int calcularMonto(LocalTime d, LocalTime h, int price) {

        int precio = price;
        int monto = 0;
        int submonto = 0;
        int cantHoras = h.getHour() - d.getHour();
        if(cantHoras>=4) {
                Descuento desc = obtenerDescuento(cantHoras);
                double porcentaje = desc.getPorcentaje();
                submonto = cantHoras * precio;
                monto = (int) (submonto - submonto*porcentaje);
        } else {
                monto = cantHoras * precio;
        }

        return monto;
}
public Reserva save(Reserva r) throws SQLIntegrityConstraintViolationException {

        return rdao.save(r);
}
public ReserveSpecification validate(String code) {

        return rdao.get(code);
}
public String confirm(String code) {
		return rdao.confirm(code);
	}
	public void sendMail(Usuario u, Reserva r, String pc) throws AddressException, MessagingException {
		try {
		 final Properties props;
		 int port = 465;
		 String to = u.getEmail();
		 String subject = "CiberRosario - Reserva ";
		 String content =
					 "Hola"+" "+u.getNombre().toUpperCase()+", "+"su reserva se ha realizado con éxito."
					+ "\nLe adjuntamos la informacion de su reserva: "
					+ "\n\t>> Computadora: "+pc.toUpperCase()+" ."
					+ "\n\t>> Reserva hecha el: "+r.getFecha_de_reserva()+" ."
					+ "\n\t>> Para el dia: "+r.getFecha_a_reservar()+" ."
					+ "\n\t>> Desde las "+r.getHoraDesde()+", "+"hasta las "+r.getHoraHasta()+" ."
					+ "\n\t>> Por un total de: $"+r.getImporte()+" ."
					+ "\nPara proseguir, al momento de llegada la reserva, presente el siguiente codigo al recepcionista. Recuerde que el total debe abonarse en
EFECTIVO en el local."
					+ ""
					+ "\n\n\tCÓDIGO: "+r.getCod_reserva().toUpperCase()+
					"\n\n"

					+"Muchas gracias. Nos vemos viciando!"
					+"\nCiberRosario";
		 props = new Properties();
		 props.put("mail.smtp.auth", "true");
		 props.put("mail.smtp.ssl.enable", "true");
		 props.put("mail.smtp.host", "smtp.gmail.com");
		 props.put("mail.smtp.port", port);
		 props.put("from", "ciberrosariopc@gmail.com");
		 props.put("username", "ciberrosariopc@gmail.com");
		 props.put("password", "rosfossvhymupehd");
		 Session sesion = Session.getInstance(props, new Authenticator() {
			protected PasswordAuthentication getPasswordAuthentication() {
				return new PasswordAuthentication(props.getProperty("username"), props.getProperty("password"));
			}
		});
		 Message mensaje = new MimeMessage(sesion);
		 mensaje.setFrom(new InternetAddress(props.getProperty("from")));
		 mensaje.setRecipients(Message.RecipientType.TO, InternetAddress.parse(to));
```

```
                    mensaje.setSubject(subject);
                    mensaje.setText(content);
                    Transport.send(mensaje);
                    System.out.println("mail enviado.");
                } catch (Exception e) {
                        System.out.println("No se pudo mandar el mail.");
                        e.printStackTrace();
                }
        }
}
```

DataReserve

```
        public Reserva save(Reserva r ) throws SQLIntegrityConstraintViolationException {

                PreparedStatement stmt = null;
                try {
                stmt = DbConnector.getInstancia().getConn().prepareStatement("INSERT INTO reservas (cod_reserva, fecha_de_reserva, fecha_a_reservar, horaDesde,
horaHasta, idUsuario, idComputadora, importe, plataforma_stream, name_stream, link_stream, rubro_work, empresa_work, descripcion_work, estado) VALUES
(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)");
                        UUID uuid = UUID.randomUUID();
              String cod = uuid.toString().substring(0, 5);
              r.setCod_reserva(cod);
                        stmt.setString(1, cod);
                        stmt.setObject(2, r.getFecha_de_reserva());
                        stmt.setObject(3, r.getFecha_a_reservar());
                        stmt.setObject(4, r.getHoraDesde());
                        stmt.setObject(5, r.getHoraHasta());
                        stmt.setInt(6, r.getIdUsuario());
                        stmt.setInt(7, r.getIdComputadora());
                        stmt.setInt(8, r.getImporte());
                        stmt.setString(15, r.getEstado());
                        if((r.getPlataforma_stream()!=null)&&(r.getName_stream()!=null)&&(r.getLink_stream()!=null)) {
                                stmt.setString(9, r.getPlataforma_stream());
                                stmt.setString(10, r.getName_stream());
                                stmt.setString(11, r.getLink_stream());
                        } else {
                                stmt.setString(9, null);
                                stmt.setString(10, null);
                                stmt.setString(11, null);
                        }
                        if((r.getRubro_work()!=null)&&(r.getEmpresa_work()!=null)&&(r.getDescripcion_work()!=null)) {
```

```
                    stmt.setString(12, r.getRubro_work());
                    stmt.setString(13, r.getEmpresa_work());
                    stmt.setString(14, r.getDescripcion_work());
                } else {
                    stmt.setString(12, null);
                    stmt.setString(13, null);
                    stmt.setString(14, null);
                }
                stmt.executeUpdate();
        } catch (SQLException e) {
                e.printStackTrace();
                throw new SQLIntegrityConstraintViolationException();
        } finally {
                try {
                    if(stmt!=null) {stmt.close();}
                    DbConnector.getInstancia().releaseConn();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
        }
        return r;
}
```

DataPc

```
public LinkedList<ComputersSpecification> GetPcsAvailable() {

        ResultSet rs = null;
        Statement stmt = null;
        LinkedList<ComputersSpecification> pcs = new LinkedList<ComputersSpecification>();
        try {
                stmt = DbConnector.getInstancia().getConn().createStatement();
                rs = stmt.executeQuery("with pc_cant as (select tpc.idTipoComputadora, count(*) cant from computadoras pc "
                                + "inner join tipo_computadora tpc "
                                + "on pc.idTipoComputadora = tpc.idTipoComputadora "
                                + "where pc.estado = 'disponible' "
                                + "group by 1) "
                                  + "select distinct pc.placa_madre, pc.placa_de_video, pc.ram, pc.procesador, pc.almacenamiento, pc.idTipoComputadora,
                                                tp.descripcion, ifnull(pc_cant.cant, 0) cant "
                                + "from computadoras pc "
                                + "left join pc_cant "
```

```
                    + "on pc.idTipoComputadora = pc_cant.idTipoComputadora "
                    + "left join tipo_computadora tp "
                    + "on tp.idTipoComputadora = pc.idTipoComputadora");
            if(rs!=null) {
                while(rs.next()) {
                    ComputersSpecification pca = new ComputersSpecification();
                    TypePc type = new TypePc();
                    type.setIdTipoComputadora(rs.getString("idTipoComputadora"));
                    type.setDescripcion(rs.getString("descripcion"));
                    pca.setMotherboard(rs.getString("placa_madre"));
                    pca.setVideocard(rs.getString("placa_de_video"));
                    pca.setRam(rs.getString("ram"));
                    pca.setCore(rs.getString("procesador"));
                    pca.setStorage(rs.getString("almacenamiento"));
                    pca.setAmount(rs.getInt("cant"));
                    pca.setType(type);
                    pcs.add(pca);
                }
                return pcs;
            }
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }finally {
            try {
                if(rs!=null) {rs.close();}
                if(stmt!=null) {stmt.close();}
                DbConnector.getInstancia().releaseConn();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        return pcs;
    }
    public void setEstado(int id, String estado) {

        PreparedStatement stmt = null;
        try {
            stmt = DbConnector.getInstancia().getConn().prepareStatement("UPDATE computadoras SET estado = ? WHERE idComputadora = ?");
            stmt.setString(1, estado);
            stmt.setInt(2, id);
				stmt.executeUpdate();
			} catch (Exception e) {
				// TODO: handle exception
			}finally {
				try {
					if(stmt!=null) {stmt.close();}
					DbConnector.getInstancia().releaseConn();
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
		}
		public int getIdAvailable(String type) {

			ResultSet rs = null;
			PreparedStatement stmt = null;
			int id = 0;
			try {
			stmt = DbConnector.getInstancia().getConn().prepareStatement("select pc.idComputadora id from computadoras pc inner join tipo_computadora tpc on
pc.idTipoComputadora = tpc.idTipoComputadora where tpc.descripcion = ? and pc.estado = 'disponible' limit 1;");
				stmt.setString(1, type);
				rs = stmt.executeQuery();
				if(rs!=null&&rs.next()) {
					id = rs.getInt("id");
					System.out.println(id);
					return id;
				}
			} catch (Exception e) {
				e.printStackTrace();
			} finally {
				try {
					if(rs!=null) {rs.close();}
					if(stmt!=null) {stmt.close();}
					DbConnector.getInstancia().releaseConn();
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
			return id;
		}
```

DataPrice

```
public int getPrice(String type) {

	CallableStatement cstmt = null;
	int p = 0 ;
	try {
		cstmt = DbConnector.getInstancia().getConn().prepareCall("{CALL get_last_price_for_pc(?, ?)}");
		cstmt.setString(1, type);
		cstmt.registerOutParameter(2, Types.INTEGER);
		cstmt.execute();
		p = cstmt.getInt(2);
		return p;
	} catch (Exception e) {
		e.printStackTrace();
	} finally {
		try {
			if(cstmt!=null) {cstmt.close();}
			DbConnector.getInstancia().releaseConn();
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}
	return p;
}
```

DataDescuento

```
public Descuento getOne(int cantHoras) {
	Descuento desc = null;
	PreparedStatement stmt = null;
	ResultSet rs = null;
	String query;
	try {
		if(cantHoras < 6) {
			query = "SELECT * FROM descuentos WHERE horas_minimas <= ?";
			stmt = DbConnector.getInstancia().getConn().prepareStatement(query);
			stmt.setInt(1, cantHoras);
		} else {
			query = "with hora as (select max(horas_minimas) horamax from descuentos where horas_minimas <= ?) "
					+ "select d.horas_minimas, d.porcentaje "
					+ "from descuentos d "
					+ "inner join hora h "
```

```
                        + "on h.horamax = d.horas_minimas ";
                stmt = DbConnector.getInstancia().getConn().prepareStatement(query);
                stmt.setInt(1, cantHoras);
            }
            rs = stmt.executeQuery();
            if(rs!=null&&rs.next()) {
                desc = new Descuento();
                desc.setHoras_minimas(rs.getInt("horas_minimas"));
                desc.setPorcentaje(rs.getDouble("porcentaje"));
            }
        } catch (Exception e) {
            // TODO: handle exception
        } finally {
            try {
                if(rs!=null) {rs.close();}
                if(stmt!=null) {stmt.close();}
                DbConnector.getInstancia().releaseConn();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        return desc;
    }
```

asd