Documento sobre os padrões utilizados.

- Iterator

```java
public Paciente buscar(Paciente usuario) {
    String sql = "SELECT * FROM usuarios WHERE id=?";
    Paciente retorno = null;
    try {
        PreparedStatement stmt = connection.prepareStatement(sql);
        stmt.setInt(1, usuario.getId());
        ResultSet resultado = stmt.executeQuery();
        if (resultado.next()) {
            retorno = new Paciente(Integer.parseInt(resultado.getString("id")), resultado.getString("nome"),
                    resultado.getString("cpf"),
                    resultado.getString("rg"),
                    resultado.getString("sexo"),
                    resultado.getString("login"),
                    resultado.getString("email"),
                    "",
                    LocalDate.parse(resultado.getString("dataNascimento")),
                    resultado.getString("endereco"),
                    resultado.getString("historico"));
            retorno = usuario;
        }
    } catch (SQLException ex) {
        Logger.getLogger(usuariosDAO.class.getName()).log(Level.SEVERE, null, ex);
    }
    return retorno;
}
```

- Prototype

```java
public abstract class LicitacaoPrototype {

    protected long codigo;
    protected String referencia;
    protected double valor;

    public abstract String exibirInfo();

    public abstract LicitacaoPrototype clonar();

    public long getCodigo() {
        return codigo;
    }

    public void setCodigo(long codigo) {
        this.codigo = codigo;
    }

    public String getReferencia() {
        return referencia;
    }

    public void setReferencia(String referencia) {
        this.referencia = referencia;
    }

    public double getValor() {
        return valor;
    }

    public void setValor(double valor) {
        this.valor = valor;
    }

}
```

```java
public class MedicamentosPrototype extends LicitacaoPrototype {

    private long codigo;
    private String referencia;
    private double valor;

    protected MedicamentosPrototype(MedicamentosPrototype mp) {
        this.codigo = mp.getCodigo();
        this.referencia = mp.getReferencia();
        this.valor = mp.getValor();
    }

    public MedicamentosPrototype(long codigo, String referencia, double valor) {
        this.codigo = codigo;
        this.referencia = referencia;
        this.valor = valor;
    }

    @Override
    public String exibirInfo() {
        return "Licitação de Medicamentos\n"
                + "Código: " + getCodigo() + " Referencia: " + getReferencia() + "Valor: R$"
                + getValor();
    }

    @Override
    public MedicamentosPrototype clonar() {
        return new MedicamentosPrototype(this);
    }

}
```

- Decorator

```java
public class Contabilidade {
    private int id;
    private String referencia;
    private double valor;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getReferencia() {
        return referencia;
    }

    public void setReferencia(String referencia) {
        this.referencia = referencia;
    }

    public double getValor() {
        return valor;
    }

    public void setValor(double valor) {
        this.valor = valor;
    }

}
```

```java
public abstract class ContabilidadeDecorator extends Contabilidade {

    Contabilidade contabil;

    public ContabilidadeDecorator(Contabilidade contabil) {
        this.contabil = contabil;
    }

    @Override
    public int getId() {
        return contabil.getId();
    }

    @Override
    public String getReferencia() {
        return contabil.getReferencia();
    }

    @Override
    public double getValor() {
        return contabil.getValor();
    }

}
```

```java
public class Salario extends ContabilidadeDecorator {

    Contabilidade contabil;

    public Salario(Contabilidade contabil, int id, String referencia, double valor) {
        super(contabil);
        this.contabil = contabil;
        contabil.setId(id);
        contabil.setReferencia(referencia);
        contabil.getValor();

    }

    @Override
    public int getId() {
        return contabil.getId();
    }

    @Override
    public String getReferencia() {
        return contabil.getReferencia();
    }

    @Override
    public double getValor() {
        return contabil.getValor();
    }

}
```

- Observer

```java
public class Horarios {
    private final String[] atendimentosnaTela;

    public Horarios(String[] atendimentosnaTela) {
        this.atendimentosnaTela = atendimentosnaTela;
    }
}
```

```java
public class HorariosSubject {

    protected ArrayList<HorariosObserver> observers;
    protected Horarios horarios;

    public HorariosSubject() {
        observers = new ArrayList<>();
    }

    public void attach(HorariosObserver observer) {
        observers.add(observer);
    }

    public void detach(int indice) {
        observers.remove(indice);
    }

    public void setState(Horarios horarios) {
        this.horarios = horarios;
        notifyObservers();
    }

    private void notifyObservers() {
        for (HorariosObserver observer : observers) {
            observer.update();
        }
    }

    public Horarios getState() {
        return horarios;
    }
}
```

```java
public abstract class HorariosObserver {

    protected HorariosSubject horarios;

    public HorariosObserver(HorariosSubject horarios) {
        this.horarios = horarios;
    }

    public abstract void update();
}
```

```java
public class FXMLHorariosObserver extends HorariosObserver implements Initializable {

    @FXML
    private TableView<Horarios> tableViewHorarios;

    @FXML
    private TableColumn<Horarios, String> tableColumnSala;
    @FXML
    private TableColumn<Horarios, String> tableColumnMedico;
    @FXML
    private TableColumn<Horarios, String> tableColumnHorario;

    private final List<Horarios> listClientes = new ArrayList<>();
    ;
    private ObservableList<Horarios> observableListClientes;

    public FXMLHorariosObserver(HorariosSubject horarios) {
        super(horarios);
    }

    @Override
    public void update() {

        tableColumnSala.setCellValueFactory(new PropertyValueFactory<>("sala"));
        tableColumnMedico.setCellValueFactory(new PropertyValueFactory<>("medido"));
        tableColumnHorario.setCellValueFactory(new PropertyValueFactory<>("horario"));

        observableListClientes = FXCollections.observableArrayList(listClientes);
        tableViewHorarios.setItems(observableListClientes);

    }

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO

    }

}
```

- Proxy

```java
public class Relatorio {

    private final int quantidadeDePascientes;
    private final int funciocariosTrabalhando;
    private final int funciocariosDeFerias;

    public Relatorio(int quantidadeDePascientes, int funciocariosTrabalhando, int funciocariosDeFerias) {
        this.quantidadeDePascientes = quantidadeDePascientes;
        this.funciocariosTrabalhando = funciocariosTrabalhando;
        this.funciocariosDeFerias = funciocariosTrabalhando;
    }

    public String getNumPascientes() {
        return "Total de Pascientes: " + quantidadeDePascientes;
    }

    public String getNumFTrabalhando() {
        return "Funcionarios trabalhando: " + funciocariosTrabalhando;
    }
    public String getNumFFerias() {
        return "Funcionarios de ferias: " + funciocariosTrabalhando;
    }

}
```

```java
public class RelatorioProxy  extends Relatorio{

    private final Usuario user;

    public RelatorioProxy(Usuario user, int quantidadeDePascientes, int funciocariosTrabalhando, int funciocariosDeFerias) {
        super(quantidadeDePascientes, funciocariosTrabalhando, funciocariosDeFerias);
        this.user = user;
    }

    @Override
    public String getNumPascientes() {
        if(temPermissaoDeAcesso()){
            return super.getNumPascientes();
        }
        return null;
    }

    @Override
    public String getNumFTrabalhando() {
        if(temPermissaoDeAcesso()){
            return super.getNumFTrabalhando();
        }
        return null;
    }
    @Override
    public String getNumFFerias() {
        if(temPermissaoDeAcesso()){
            return super.getNumFFerias();
        }
        return null;
    }

    private boolean temPermissaoDeAcesso() {
        return user.getLogin().equalsIgnoreCase("admin") && user.getSenha().equalsIgnoreCase("admin12345");
    }
}
```

- Builder

```java
public class Prontuario {

    private int ID;
    private int IDPaciente;
    private String historico;
```

```java
public abstract class ProntuarioBuilder {
    protected Prontuario prontuario;

    public ProntuarioBuilder() {
        this.prontuario = new Prontuario();
    }

    public abstract void buildId(int id);

    public abstract void buildIDPaciente(int id);

    public abstract void buildhistorico(String historico);


    public Prontuario getProntuario() {
        return this.prontuario;
    }
}
```

```java
public class CirugiaBuilder extends ProntuarioBuilder {

    private Prontuario prontuario;

    @Override
    public void buildId(int id) {
       prontuario.setID(id);
    }

    @Override
    public void buildIDPaciente(int id) {
        prontuario.setIDPaciente(id);
    }

    @Override
    public void buildhistorico(String historico) {
        prontuario.setHistorico(historico);
    }


}
```

```java
 */
public class MedicoBuilderC {
    protected ProntuarioBuilder prontuario;

    public MedicoBuilderC(ProntuarioBuilder prontuario) {
        this.prontuario = prontuario;
    }

    public void construirProntuario(int id, int idP, String historico) {
        prontuario.buildId(id);
        prontuario.buildIDPaciente(idP);
        prontuario.buildhistorico(historico);
    }

    public Prontuario getProntuario() {
        return this.prontuario.getProntuario();
    }
}
```

- Interpreter

```
public class LocalDate extends LocalDateInterpreter {

    java.util.Date dataFormatada;
    @Override
    public String Data(String dataSistema) {
        String dataBanco = "";
        try {
            dataFormatada = new SimpleDateFormat("dd/MM/yyyy").parse(dataSistema);
            dataBanco = new SimpleDateFormat("yyyy-MM-dd").format(dataFormatada);
        } catch (ParseException ex) {
            Logger.getLogger(LocalDate.class.getName()).log(Level.SEVERE, null, ex);
        }
        return dataBanco;
    }

    @Override
    public String Hora(String horaSistema) {
        String dataBanco = "";
        try {
            dataFormatada = new SimpleDateFormat("hh:mm").parse(horaSistema);
            dataBanco = new SimpleDateFormat("hh:mm").format(dataFormatada);
        } catch (ParseException ex) {
            Logger.getLogger(LocalDate.class.getName()).log(Level.SEVERE, null, ex);
        }
        return dataBanco;
    }

    @Override
    public String DataeHora(String dataEhora) {
        String dataBanco = "";
        try {
            dataFormatada = new SimpleDateFormat("dd/MM/yyyy hh:mm").parse(dataEhora);
            dataBanco = new SimpleDateFormat("yyyy-MM-dd  hh:mm").format(dataFormatada);
        } catch (ParseException ex) {
            Logger.getLogger(LocalDate.class.getName()).log(Level.SEVERE, null, ex);
        }
        return dataBanco;
    }

}
```

```
 */
public abstract class LocalDateInterpreter {

    public abstract String Data(String data);

    public abstract String Hora(String hora);

    public abstract String DataeHora(String dataEhora);

}
```

- Visitor

```java
public interface Documento {
    public Object aceitar(DocumentoRelatorio dr);
}
```

```java
public interface DocumentoRelatorio {

    public Object visitar(JSon j);

    public Object visitar(XML x);

    public Object visitar(CSV c);
}
```

```java
public class JSon implements Documento {

    @Override
    public Object aceitar(DocumentoRelatorio dr) {
        return dr.visitar(this);
    }

    public void gerarJson(Relatorio[] json) {

        try {

            try (FileWriter file = new FileWriter("/relatorios/"+System.currentTimeMillis()+".json")) {
                file.write(Arrays.toString(json));
                file.flush();
            }

        } catch (IOException e) {
        }
    }
}
```

```java
public class XML implements Documento {

    @Override
    public Object aceitar(DocumentoRelatorio dr) {
        return dr.visitar(this);
    }

    public void gerarXML(Relatorio[] xml) {
        try {

            try (FileWriter file = new FileWriter("/relatorios/"+System.currentTimeMillis()+".xml")) {
                file.write(Arrays.toString(xml));
                file.flush();
            }

        } catch (IOException e) {
        }
    }

}
```

```java
public class CSV implements Documento{

    @Override
    public Object aceitar(DocumentoRelatorio dr) {
        return dr.visitar(this);
    }

    public void gerarCSV(Relatorio[] csv) {
        try {

            try (FileWriter file = new FileWriter("/relatorios/"+System.currentTimeMillis()+".csv")) {
                file.write(Arrays.toString(csv));
                file.flush();
            }

        } catch (IOException e) {
        }

    }

}
```

```java
public class ExportarGelatorio implements DocumentoRelatorio {

    Relatorio[] relt;

    public ExportarGelatorio(Relatorio[] relt) {
        this.relt = relt;
    }

    @Override
    public Object visitar(JSon j) {
        j.gerarJson(relt);
        return null;
    }

    @Override
    public Object visitar(XML x) {
        x.gerarXML(relt);
        return null;
    }

    @Override
    public Object visitar(CSV c) {
        c.gerarCSV(relt);
        return null;
    }

}
```

```java
public class ValidateDoc implements DocumentoRelatorio {

    @Override
    public Object visitar(JSon j) {
        return true;
    }

    @Override
    public Object visitar(XML x) {
        return true;
    }

    @Override
    public Object visitar(CSV c) {
        return true;
    }

}
```