CENTRO UNIVERSITÁRIO FAMETRO
CURSO DE GRADUAÇÃO TECNOLÓGICA EM
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CARLOS EDUARDO SERPA BRITO

DESENVOLVIMENTO BASEADO EM COMPONENTES

FORTALEZA - CE
2022.1

SUMÁRIO

1. Introdução :

Tendo em vista o constante aumento das demandas de softwares na nossa atualidade, práticas que proporcionam facilidade, economia de gastos e rapidez no desenvolvimento são cada vez mais necessárias e bem-vindas dentro dos projetos.

Dentre as mais diversas práticas existentes no mercado hoje, temos o Reuso de Software, que se resume a reutilização de partes de softwares já existentes dentro de novos projetos, por meio de técnicas como reuso de frameworks, requisitos, software baseada em componentes, arquitetura orientada a serviços, dentre outras, com o objetivo de se obter o mesmo resultado, economizando tempo e custos.

No entanto, nesse artigo iremos focar apenas no estudo do Software Baseado em Componentes, tratando das suas definições, aspectos técnicos, aplicação, dentre outros assuntos. Para realização desse estudo, trago como exemplo dois modelos de componentes, Enterprise Java Beans(EJB) e o JavaBeans, duas tecnologias bastante conhecidas, que apesar de terem um nome similar, possuem algumas diferenças que iremos tratar mais afundo a seguir.

2. Desenvolvimento :

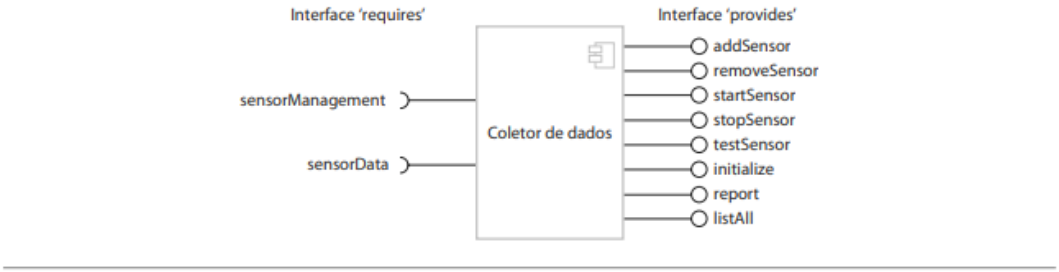
Assim como visto anteriormente, nesse tópico iremos tratar de forma mais detalhada sobre os componentes e os softwares baseados neles, trazendo definições, aspectos técnicos, construção, evolução, desenvolvimento, vantagens/desvantagens, aplicação e uma comparação dos modelos de exemplo, EJB e JavaBeans.

2.1 Definição:

Mas, afinal, o que são os componentes? Bom, sabemos que de maneira geral, componentes são partes de um todo, e de forma análoga isso também se aplica um pouco aos Componentes de Softwares, que apesar de ainda não existir uma definição tão concreta, podemos considerar como "Blocos de Códigos" (partes) modulares, independentes, altamente documentados e padronizados, implantáveis e reutilizáveis, que estão passíveis de composição a outro Software (todo). Ainda é válido destacar a importância das Interfaces, que são responsáveis por descrever e especificar a utilidade daquele determinado componente, além de permitir a comunicação dos componentes entre si.

exemplo de uma interface de componente segundo **Sommerville** [SOM2011],

Figura 17.2 Um modelo de componente coletor de dados



Quando tratamos da construção de um componente, ou seja, o seu desenvolvimento, precisamos levar em conta todas as características vistas no parágrafo anterior, sendo todas bem definidas e claras.

Visto isso, podemos definir também a Engenharia de Software Baseada em Componentes ou apenas Desenvolvimento Baseado em Componentes (DBC), que de forma similar a ES normal também passa pelos processos básicos de desenvolvimento de um software, como análise de requisitos, especificação, contato com cliente, mas com ênfase na reutilização de componentes funcionais e lógicos com interfaces claras e bem definidas.

- Enterprise JavaBeans: O EJB faz parte da plataforma Java EE (enterprise edition), pode ser considerado como uma arquitetura de componentes distribuídos voltada para o desenvolvimento corporativo. Ele permite criar aplicações modulares, ou seja, distribuídas, de forma rápida e eficiente. Sua arquitetura é cliente/servidor, ou seja, os componentes estão do lado do servidor, sendo acessados de forma remota pelo cliente.

- JavaBeans: O JavaBeans é um modelo de componentes, ou também pode ser considerado como os componentes de softwares em si, que tem como objetivo fornecer unidades modulares, independentes e reutilizáveis para que possam ser manipuladas de forma visual por meio de uma ferramenta de desenvolvimento.

2.2 Aspectos Técnicos:

Para um melhor entendimento dos componentes em si, nesse tópico irei tratar de forma mais detalhada das suas características vistas anteriormente, usando a definição apresentada por Sametinger [SJO01]:

- Autocontido: característica do componente de ser “reusável” sem que exista uma necessidade/dependência de outros componentes.

- Identificação: componentes devem ter sua identificação clara e guardadas em vez de juntas com outros artefatos.

- Funcionalidade: componentes devem ter suas funções bem especificadas e claras, entrando a necessidade de alta documentação.

- Interface: assim como visto antes, a interface é de extrema importância para um componente, logo ela precisa ser clara e objetiva, indicando como aquele componente pode ser reusado e o que ele depende/necessita para funcionamento.

- Documentação: a documentação é essencial e indispensável para os componentes e seu reuso.

- Condição de reuso: componentes devem ser tratados de uma maneira que não afete suas condições de reuso, ou seja, deve-se preservar o reuso.

2.3 Construção de Componentes:

Para o desenvolvimento dos componentes em si, iremos usar como parâmetro o modelo sugerido por Leonardo Silveira [LSI01], o qual sugere o processo de construção através das seguintes etapas:

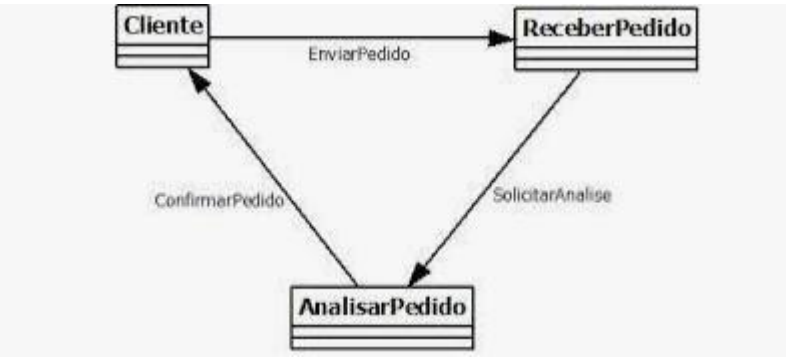
- 1. Identificação das Classes relacionadas ao problema do projeto.

- 2. Identificação das Classes relacionadas infraestrutura do projeto.

- 3. Elaboração das Classes de projeto que não são obtidas como componentes reutilizáveis.

- 3.1 Especificação de mensagens quando classes ou componentes colaboram entre si.

exemplo diagrama de colaboração (Wikipédia),

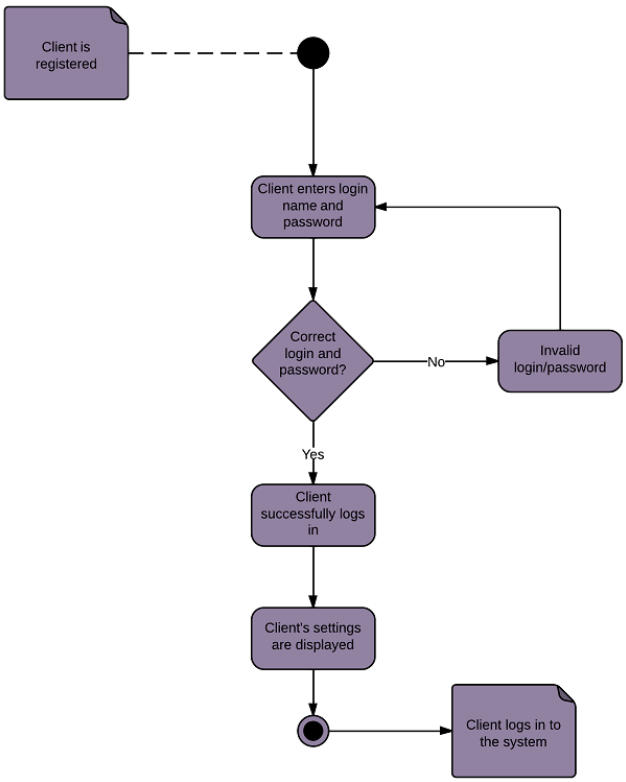


- 3.2 Identificação das interfaces de cada componente. (exemplo de interface no tópico 2.1)

- 3.3 Elaboração de atributos, tipos de dados e estrutura necessária para a implementação.

- 3.4 Descrição detalhada do fluxo de processo obtido em todas as operações.

exemplo diagrama de atividades (**Lucidchart**),



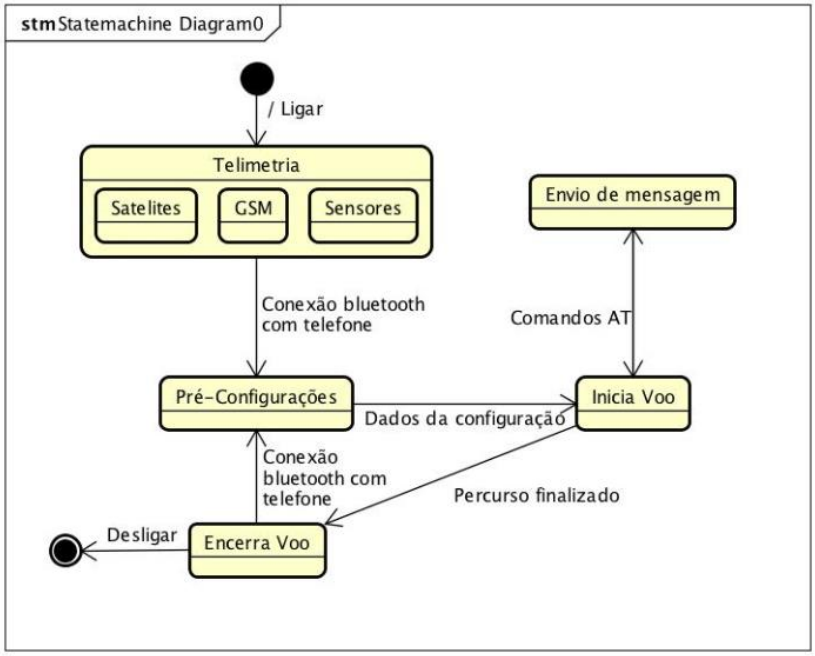
- 4. Descrição das fontes de dados persistentes (BD) e identificação das classes necessárias para gerenciá-las.

exemplo classe DAO responsável por gerenciar o BD em uma aplicação de sobremesas.

```java
package dao;

import java.util.List;

public class SobremesaDAO {

    public static void salvar(Sobremesa s) {
        EntityManager em  =  JpaUtil.criarEntityManager();
        em.getTransaction().begin();
        em.persist(s);
        em.getTransaction().commit();
        em.close();
    }


    public static void editar(Sobremesa s) {
        EntityManager em  =  JpaUtil.criarEntityManager();
        em.getTransaction().begin();
        em.merge(s);
        em.getTransaction().commit();
        em.close();
    }

    public static void excluir(Sobremesa s) {
        EntityManager em  =  JpaUtil.criarEntityManager();
        em.getTransaction().begin();
        em.remove(s);
        em.getTransaction().commit();
        em.close();

    }
```

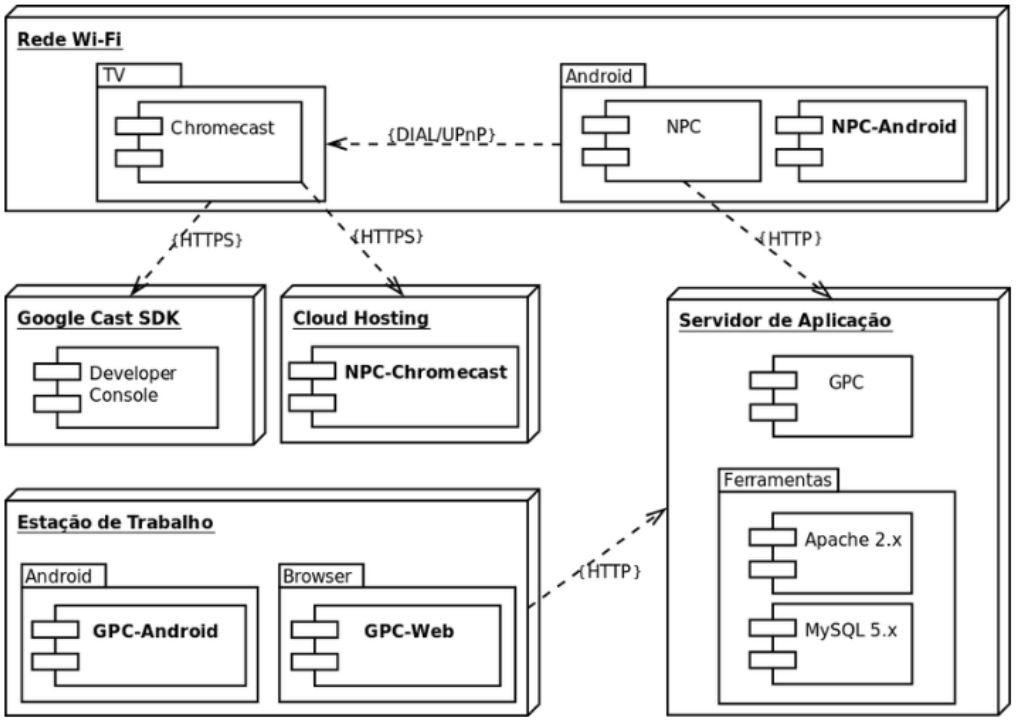
- 5. Desenvolvimento e elaboração das representações comportamentais para cada componente/classe.

Exemplo diagrama de estado (ResearchGate),



- 6. Elaboração dos diagramas de implantação para detalhar a implementação.

exemplo diagrama de implantação (ResearchGate),



2.4 Evolução de Componentes:

Assim como visto anteriormente, sabemos que a “reusabilidade” é uma característica muito importante e essencial dentro de um componente, e esse conceito já vem desde os anos 1960, onde se existiam bibliotecas científicas que reutilizavam algoritmos bem definidos para diversas aplicações, mas com uma certa limitação. No entanto, é válido salientar que apenas em 1990 aproximadamente, surgiu a possibilidade da criação de novas aplicações por meio do reuso de componentes para os engenheiros de software. E essa prática vem se aperfeiçoando desde então, seja nas metodologias que surgiram nos anos após, assim como novas tecnologias.

2.5 Desenvolvimento com Componentes:

Antes de iniciarmos esse tópico, é importante compreendermos duas perspectivas que apesar de parecerem iguais, são distintas, primeiro temos o Desenvolvimento de Componentes, que está relacionado a construção e implementação do componente em si, junto a sua documentação, interface, codificação, visto no tópico 2.3. Em seguida, chegamos ao Desenvolvimento com Componentes, o que se resume a utilização do reuso de componentes para o desenvolvimento de novos softwares, o qual iremos tratar nesse tópico.

Além das atividades básicas já vistas na engenharia de software padrão, como a análise de requisitos, especificações do sistema e os feedbacks do cliente, o desenvolvimento com componentes possui algumas etapas adicionais que podem mudar de autor para autor. O modelo que iremos tratar é o sugerido por Brown e Keit [BRKT97], os quais destacam as 5 atividades essenciais para utilização dos componentes no desenvolvimento como: Seleção, Qualificação, Adaptação, Composição e Atualização.

- Seleção: a seleção é a primeira e uma das mais essenciais atividades do processo, aqui é onde serão buscados os componentes que tem potencial para fazer parte do projeto, e após um estudo e investigação sobre os componentes buscados e suas caraterísticas, é realizado a seleção dos mesmos.

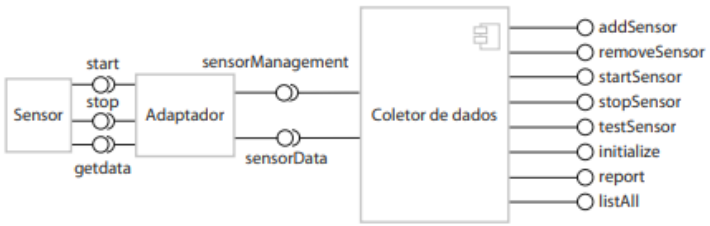
- Qualificação: uma vez selecionado os componentes potenciais para o projeto, é preciso garantir que eles realmente irão executar todas as funções necessárias, portanto, nessa etapa é realizado uma análise mais detalhada, separando e selecionando de forma mais concreta apenas os componentes que realmente são úteis.

- Adaptação: nessa etapa é realizado correções de "compatibilidade" entre os componentes selecionados e o sistema a ser desenvolvido, visto que muitas vezes ambos não serão compatíveis nativamente, sendo necessário realizar técnicas de adaptação, como alteração de código, herança, wrapping, as quais não vamos entrar afundo nesse artigo, para que possibilite o funcionamento.

- Composição: também pode ser considerada como integração, essa atividade busca permitir a interação e comunicação de todos os componentes e frameworks de componentes entre si dentro do sistema.

exemplo de uma composição segundo Sommerville [SOM2011],

Figura 17.10 Um adaptador conectando o coletor de dados e um sensor



- Atualização: e por último, chegamos à atualização, que visa a substituição de componentes antigos por componentes com versões mais atuais e melhores, sem alterar nas funções do sistema.

2.6 Tecnologias do EJB e JavaBeans:

Ainda que anteriormente, o EJB e o JavaBeans se mostraram similares, na tabela a seguir podemos perceber que se trata de duas tecnologias diferentes, com aspectos, propósitos e estruturas bastante distintas.

tabela de comparação segundo Spagnoli [SPA01],

Tabela 5 - Características de JB e EJB

Aspecto	JB	EJB
Componente	Componente que pode ser manipulado visualmente	Componente do lado servidor acessados remotamente
Tipos	Não apresenta (JB 1.0)	*Session bean, entity bean* e *message-driven bean*
Visibilidade	Podem ser objetos visuais ou não	Não são objetos visuais
Implementação	Classes Java que apresentam métodos `get/set`	Classes Java que implementam interfaces específicas (`javax.ejb`)
Aplicações desenvolvidas	Baseadas em eventos (lado cliente)	Distribuídas (lado servidor)
Persistência	Sim	*Stateful session bean*
Ambiente de execução	Não é necessário	Necessário

2.7 Vantagens e Desvantagens:

Ainda que, a Engenharia de Software Baseada em Componentes possua um propósito bastante positivo para o desenvolvimento, é importante destacar alguns dos seus pontos negativos para que sejam avaliado antes de sua utilização, assim como para as tecnologias EJB e JavaBeans. É importante destacar que muitos dos prós e contras do DBC também se aplicam as tecnologias EJB e JavaBeans, não sendo necessária a repetição.

- Desenvolvimento Baseado em Componentes (DBC):

Vantagens: reutilização de componentes, economia de custos, maior rapidez no desenvolvimento, redução de manutenção, desenvolvimento em paralelo.

Desvantagens: dificuldade na escolha dos componentes, fraca cultura de reuso, qualidade dos componentes (fator externo), custo de tempo para desenvolver um componente (visto que ele necessita ser feito de forma genérica)

- EJB:

Vantagens: desenvolvimento simples e rápido graças a grande quantidade de recursos disponíveis, não existe uma necessidade tão real de entender o que o EJB realmente faz, apenas saber utilizá-lo, acesso remoto a outros componentes de forma simples por meio das API's e soluções integradas do EJB, fornece serviços de transações, segurança, persistência e outros.

Desvantagens: necessita da utilização de um Container EJB para funcionamento, a aplicação de Dependência é limitada ao uso de anotações, o Container disponibiliza uma gama muito grande de recursos, que muitas vezes não serão utilizado, tornando-o pesado para a aplicação.

- JavaBeans:

Vantagens: é possível controlar todas as propriedades, métodos e eventos de um Bean, JavaBeans podem ser configurados por meio de softwares externos, um bean recebe e registra eventos derivados de outros objetos, gerando novos eventos que serão enviados a outros objetos, permite armazenamento persistente das configurações de um Bean.

Desvantagens: permite que uma classe com um construtor nulo possa ser instanciada contendo um estado inválido, e caso esse problema não seja percebido pelo desenvolvedor, ela pode passar desapercebida pelo compilador. A necessidade de criação de getter/setter para todas as propriedades de um Bean, pode gerar duplicação de código.

2.8 Aplicação e Uso:

Nesse tópico, irei mostrar algumas aplicações de uso exemplos tanto do JavaBeans quanto do EJB, sem entrar tão afundo na parte técnica de Java e programação, visto que esse não é o propósito do estudo.

- Implementando um Componente do tipo Sessão com EJB:

```java
import java.text.SimpleDateFormat;
import java.util.Date;

import javax.ejb.LocalBean;
import javax.ejb.Stateless;

@Stateless
@Local
public class ConversorBean {

	public String formatarData(Date data){

		SimpleDateFormat formata = new SimpleDateFormat("MM/dd/yyyy");
		return formata.format(data);
	}
}
```

exemplo Devmedia [DVM02]

As anotações “@Stateless” e “@Local”, indicam o tipo do componente (Bean) e seu acesso limitado apenas para dentro do Container EJB, respectivamente.

- Implementando uma Classe Bean no Eclipse com JavaBeans:

```java
package bean;

public class User implements java.io.Serializable {

    private String id;
    private String senha;

    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getSenha() {
        return senha;
    }
    public void setSenha(String senha) {
        this.senha = senha;
    }

    public User() {
        this.id="Admin";
        this.senha="123";
    }

}
```

Para que uma classe seja considerada como Bean dentro do JavaBeans ele precisa ter alguns requisitos como a implementação do “Serializable” (possibilita a persistência), construtor sem argumentos, métodos “Get’s” e “Set’s”, assim como visto na foto.

3. Conclusão :

Portanto, conclui-se que o conceito de reuso vem crescendo bastante nos últimos anos, sendo uma prática bastante positiva, tendo seus pontos positivos e negativos que devem ser levados em conta na hora de escolher o desenvolvimento baseado em componentes como modelo para seu projeto, dando destaque ao aumento de produtividade e redução de custos e prazos com seu uso. Em relação as tecnologias EJB e JavaBeans, podemos concluir que não existem diretamente uma relação de melhor ou pior, mas sim a mais adequada para a aplicação que você deseja.

4. Bibliografia :

- Desenvolvimento Baseado em Componentes. **Devmedia**, 2018. Disponível em:https://www.devmedia.com.br/desenvolvimento-baseado-em-componentes-revista-java-magazine 110/26550. Acesso em: 29 de maio de 2022.

- [LSI01] SILVEIRA, Leonardo. Métodos Avançados de Programação. 01 de maio de 2022 Apresentação de Power Point. Disponível em: http://educacaoonline.unifametro.edu.br/mod/resource/view.php?id=293113&redirect=1. Acesso em: 29 de maio de 2022.

- [SPA01] SPAGNOLI, Luciana de Araújo; BECKER, Karin. Estudo Sobre o Desenvolvimento Baseado em Componentes. Maio de 2003. Disponível em: https://www.pucrs.br/facin-prov/wp-content/uploads/sites/19/2016/03/tr026.pdf. Acesso em: 31 de maio de 2022.

- KLEEMANN, Jean Wagner. Engenharia de Componentes. Disponível em: http://www.linhadecodigo.com.br/artigo/3119/engenharia-de-componentes-parte-1.aspx. Acesso em 1 de junho de 2022.

- Engenharia de Software Baseada em Componentes. **StringFixer**. Disponível em: https://stringfixer.com/pt/Component-based_software_engineering. Acesso em: 1 de junho de 2022.

- [DVM02] Enterprise JavaBeans. **Devmedia**. Disponível em: https://www.devmedia.com.br/enterprise-javabeans/26402. Acesso em 1 de junto de 2022.

- [BRKT97] BROWN, Alan W.; KEITH, Short; 1997. On Components and Objects: The Foundation of Component-Based Development. Disponível em: https://www.researchgate.net/publication/3698622_On_components_and_objects_the_foundations_of_component-based_development. Acesso em: 1 de junho de 2022.

- [SOM2011] SOOMERVILLE, Ian. 2011. Software Engeneering 9th Edition. Disponível em: https://www.facom.ufu.br/~william/Disciplinas%202018-2/BSI-GSI030-EngenhariaSoftware/Livro/engenhariaSoftwareSommerville.pdf. Acesso em 1 de junho de 2022.

- AZEVEDO, Douglas José Peixoto. Evolução do Software. **Batebyte.** Disponível em: http://www.batebyte.pr.gov.br/Pagina/Evolucao-de-Software. Acesso em 1 de junho de 2022.

- JavaBeans. **Wikipédia**. Disponível em: https://pt.wikipedia.org/wiki/JavaBeans#:~:text=JavaBeans%20são%20componentes%20de%20software,de%20uma%20ferramenta%20de%20desenvolvimento". Acesso em 1 de junho de 2022.

- [SJO01] SAMETINGER, Johannes. Software Engineering with Reusable Components. Disponível em: https://www.researchgate.net/publication/220691072_Software_Engineering_with_Reusable_Components. Acesso em 1 de junto de 2022.