

**BreadTrack:** Trabalho Integrado dos Componentes de Banco de Dados II, Engenharia de Software I e Programação II

Arthur Borger Kochem \*

Arthur Costa Gruber \*\*

Bruno Gabriel Konzen \*\*\*

Franciele Petry \*\*\*\*

Leonardo Agostini Costa \*\*\*\*\*

Otília Donato Barbosa \*\*\*\*\*

Roberson Junior Fernandes Alves \*\*\*\*\*

---

\* Discente do Curso de Ciência da Computação  
Unoesc - Campus de São Miguel do Oeste  
Rua Oiapoc, 2011. São Miguel do Oeste-SC  
arthurkochem12@gmail.com

\*\* Discente do Curso de Ciência da Computação  
Unoesc - Campus de São Miguel do Oeste  
Rua Oiapoc, 2011. São Miguel do Oeste-SC  
arthurcg21@gmail.com

\*\*\* Discente do Curso de Ciência da Computação  
Unoesc - Campus de São Miguel do Oeste  
Rua Oiapoc, 2011. São Miguel do Oeste-SC  
brunogkonzen@hotmail.com

\*\*\*\* Mestre em Informática  
Docente do Curso de Ciência da Computação  
Unoesc - Campus de São Miguel do Oeste  
Rua Oiapoc, 211. São Miguel do Oeste-SC  
franciele.petry@unoesc.edu.br

\*\*\*\*\* Discente do Curso de Ciência da Computação  
Unoesc - Campus de São Miguel do Oeste  
Rua Oiapoc, 2011. São Miguel do Oeste-SC  
Laccosta1242@gmail.com

\*\*\*\*\* Mestre em Engenharia Biomédica e  
Informática Industrial  
Docente do Curso de Ciência da Computação  
Unoesc - Campus de São Miguel do Oeste  
Rua Oiapoc, 2011. São Miguel do Oeste-SC  
otilia.barbosa@unoesc.edu.br

\*\*\*\*\* Mestre em Computação Aplicada (UEPG)  
Professor do Curso de Bacharelado em Ciência da  
Computação  
Unoesc - Campus de São Miguel do Oeste  
Rua Oiapoc, 211 – São Miguel do Oeste – SC  
roberson.alves@unoesc.edu.br

## Resumo

Este artigo apresenta um estudo de caso sobre a implementação de um Sistema de Gerenciamento de Banco de Dados (SGBD) em uma padaria, com o objetivo de aprimorar o gerenciamento de informações e processos relacionados ao estabelecimento. Inicialmente, foi realizada uma análise das necessidades da padaria, identificando os principais requisitos para o SGBD. Os requisitos incluíam o armazenamento e a organização de dados de clientes, produtos, estoque, vendas, pedidos de fornecedores e funcionários. A modelagem do banco de dados foi realizada utilizando-se a abordagem relacional, onde foram definidas as entidades, atributos e relacionamentos relevantes para a padaria. Com o banco de dados modelado, procedeu-se o estudo arquitetando o banco com a base da engenharia de software, seguidos de modelos ágeis que facilitam o entendimento do software em geral. Por fim, foi realizada a implementação do SGBD, criando as tabelas, estabelecendo as chaves primárias e estrangeiras e populando o banco com os dados existentes na padaria. Ao longo da implementação, foram realizados testes para verificar a eficácia do SGBD. Foram utilizadas ferramentas como Trello para planejar o projeto juntamente com o PostgreSQL, Eclipse, Visual Paradigm e DBeaver para modelagem e estruturação e o versionamento foi feito via GitHub. É crucial enfatizar a necessidade de uma abordagem robusta em engenharia de software, que compreenda aspectos como o controle de versionamento, para satisfazer as demandas em constante evolução de uma agência de eventos de renome.

**Palavras-chave:** Desenvolvedores. Engenharia. Software. Programação. Dados. Informações. Padaria. SGBD. Implementação.

## 1 INTRODUÇÃO

Com o avanço tecnológico, diversas áreas têm se beneficiado da automação e otimização de processos, em que a eficiência na gestão de informações desempenha um papel crucial, a implementação de soluções tecnológicas torna-se essencial para o sucesso e reconhecimento de estabelecimentos especializados. Este artigo discute o desenvolvimento de um Sistema Gerenciador de Banco de Dados para o controle de uma padaria. Neste contexto, surge a proposta de desenvolvimento do software "BreadTrack", dedicado a atender às demandas específicas de uma padaria renomada.

A padaria moderna não se limita apenas à produção de pães e doces; ela representa um ambiente dinâmico onde a eficiência operacional é crucial. O sistema visa oferecer uma solução integrada, otimizando desde o controle de estoque e gestão de fornecedores até o acompanhamento de pedidos e preferências dos clientes.

Assim, diante desse contexto, o artigo apresentará a abordagem do BreadTrack, destacando sua proposta inovadora para a gestão de pedidos em padarias, abrangendo desde a modelagem inicial até os resultados obtidos durante o desenvolvimento do sistema.

Uma solução abrangente para otimizar a gestão da padaria "BreadTrack" seria a implementação de um sistema integrado que englobe não apenas o controle de estoques, pedidos e vendas, mas também considere aspectos como a experiência do cliente, a eficiência operacional e a adaptação às demandas dinâmicas do setor alimentício, tudo isso integrado em uma gestão de pedidos eficiente juntamente com uma análise de relatórios em um sistema integrado e intuitivo.

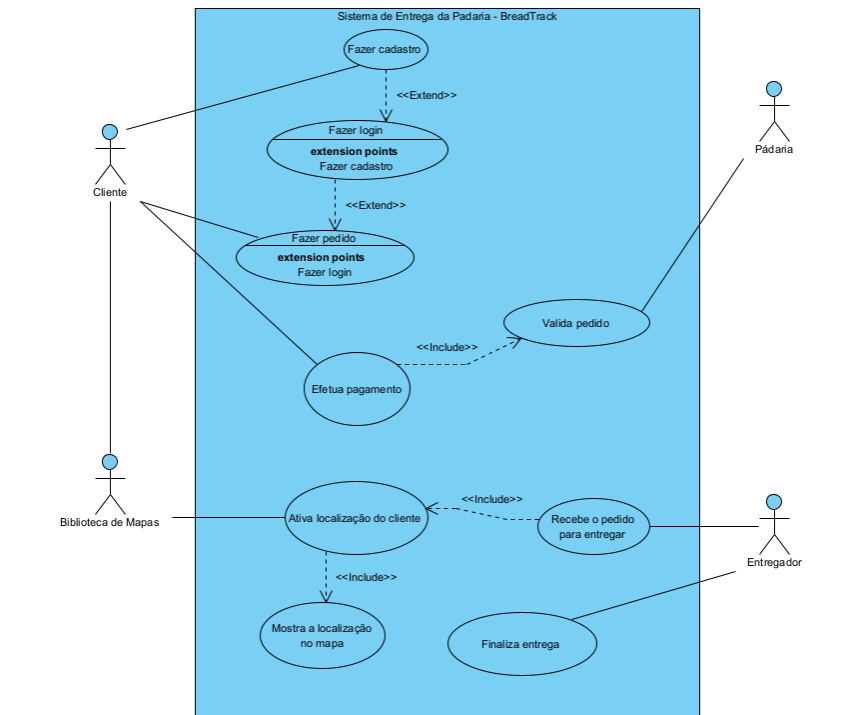
## 2 MATERIAIS E MÉTODOS

Para a realização deste trabalho utilizamos várias ferramentas, sendo elas o DBBeaver para a criação da base de dados em PostgreSQL, o Visual Paradigm para realização modelagem do banco de dados, além disso foi utilizado também para confecção do modelo de caso de uso, diagrama de sequência, diagrama de atividade, diagrama de estado e diagrama de classe, no ambiente de programação foi utilizado o github, onde realizamos métodos de criação, exclusão, listagem, busca e atualização, além do fluxo de vendas, foi utilizada a linguagem de programação Java, e as bibliotecas Jakarta, Java Persistence API, e o framework Springboot, utilizamos ainda o github como ferramenta para o versionamento. Trello para organização de cartões e tarefas realizadas pelo grupo, e o BRModelo para realização do modelo conceitual.

### 2.1 MODELAGEM

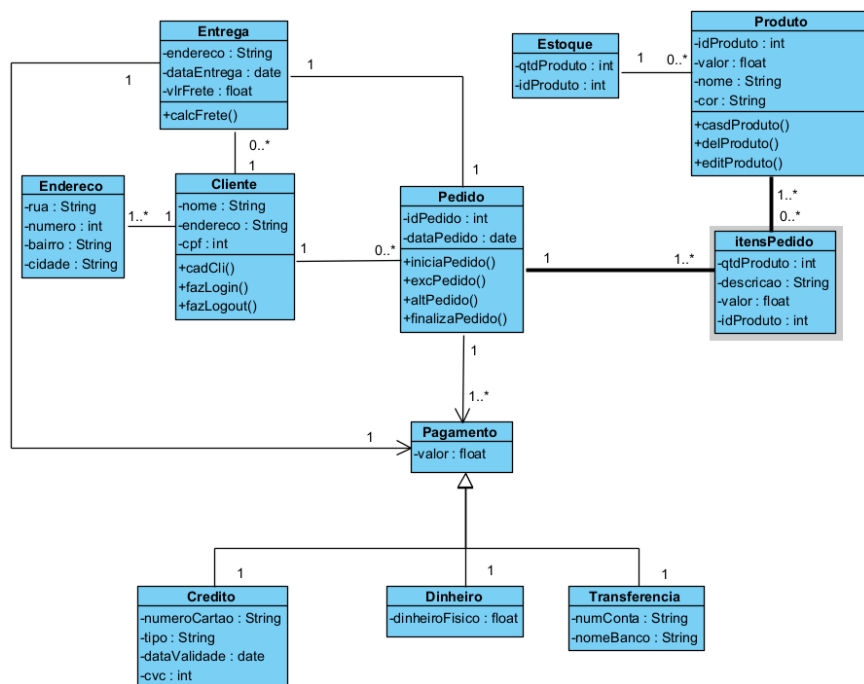
A partir dos requisitos já levantados, foi dado início a construção dos diagramas de casos de uso e de classes utilizando a ferramenta Visual Paradigm. Abaixo as Figuras 1 e 2 mostram os diagramas.

Figura 1 - Diagrama de Caso de Uso



Fonte: Autoria própria (2023).

Figura 2 - Diagrama de Classes

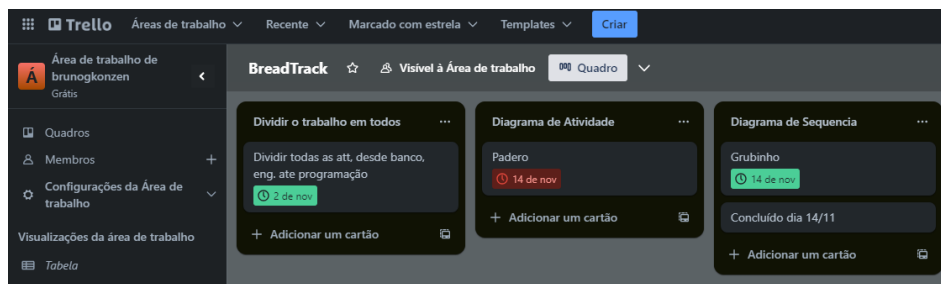


Fonte: Autoria própria (2023).

### 3 DESENVOLVIMENTO

Inicialmente foi criado um KANBAN na ferramenta Trello, para organizar as tarefas e decidir quais atividades cada integrante ficou responsável. A Figura 3 mostra o KANBAN inicialmente em seus primeiros dias.

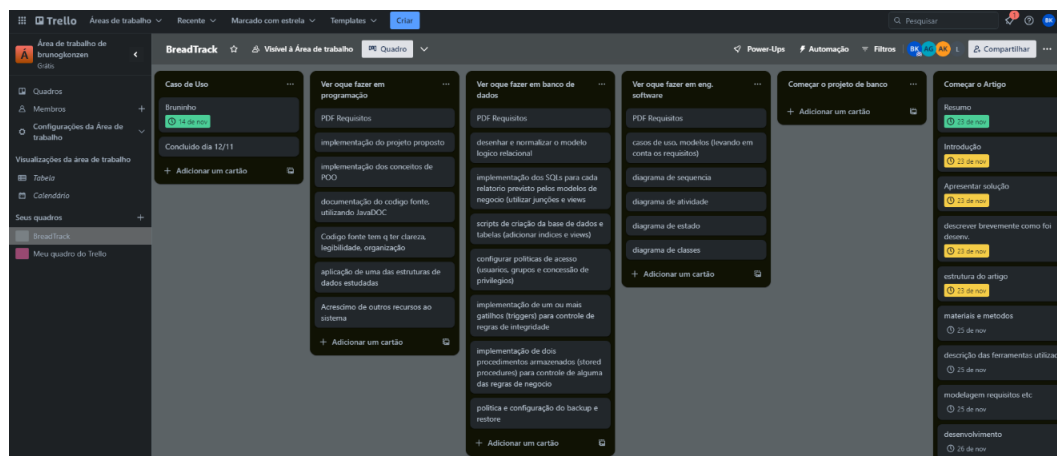
Figura 3 - Estado inicial do KANBAN



Fonte: Autoria própria (2023).

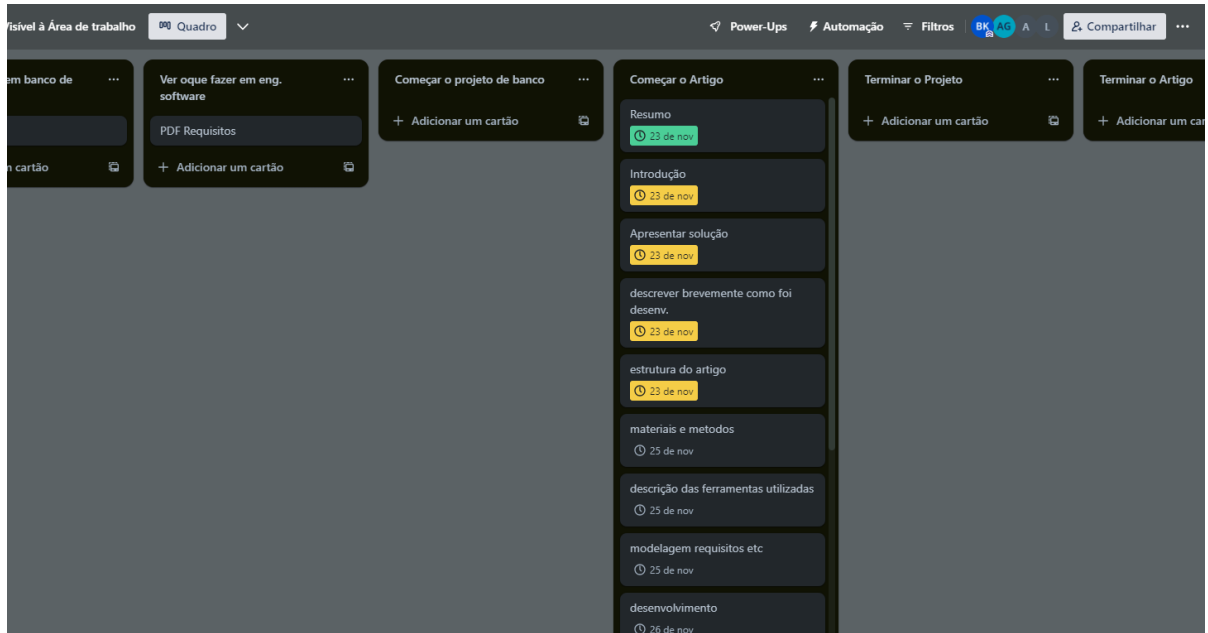
É perceptível que no começo as atividades eram básicas e com o objetivo de organização do projeto, somente mais tarde que as modelagens e implementações foram adicionadas como atividades como mostra a Figura 4 e 5.

Figura 4 - KANBAN das atividades



Fonte: Autoria própria (2023).

Figura 5 - Estágio final do KANBAN



Fonte: Autoria própria (2023).

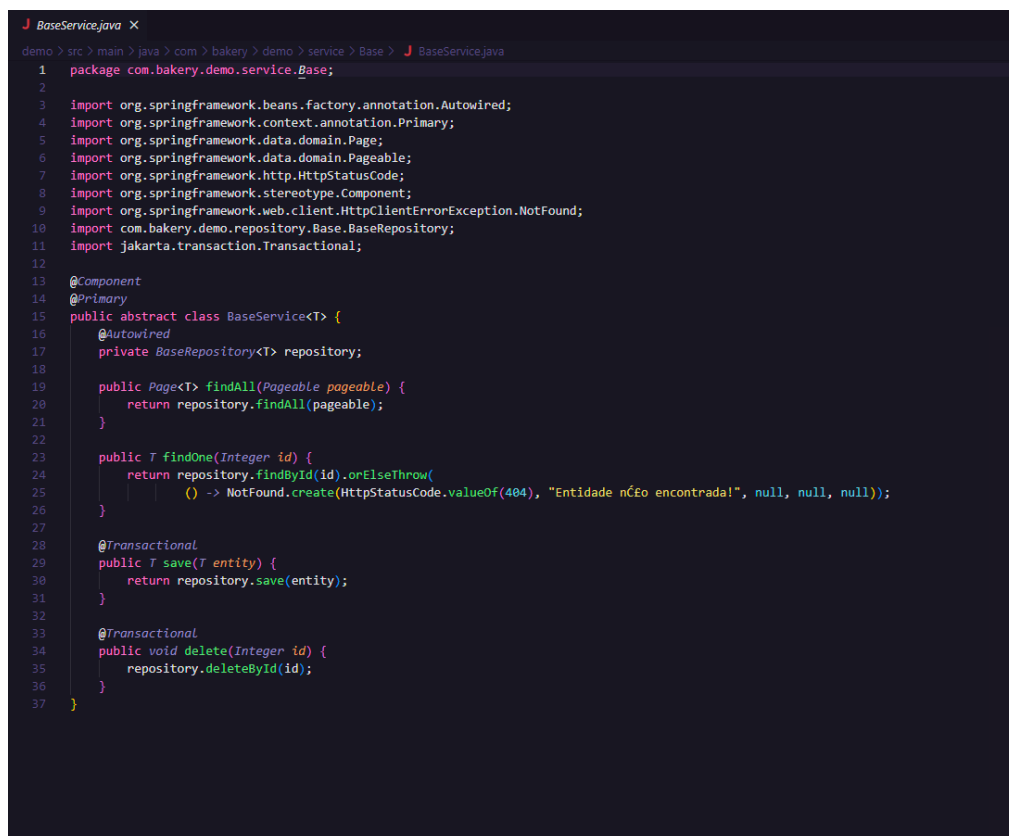
## 4 SISTEMA BREADTRACK

Com a modelagem concluída e tarefas já estabelecidas, foi dado início a parte de Banco de Dados, onde a estrutura principal já foi elaborada semestre passado, sendo acrescentados pequenos ajustes nas colunas, relatórios e script de inserts do banco de dados, além disso foram criados grupos e usuários com seus próprios privilégios, índices de colunas das principais tabelas do banco, views para facilitar pesquisa dos relatórios já estabelecidos, e implementado dois gatilhos, e criado políticas e configuração do backup e restore.

### 4.1 CÓDIGOS

Segue Figuras 6, 7, 8 e 9 dos códigos realizados em Java para implementação dos códigos:

Figura 3 - BaseService



```

1 package com.bakery.demo.service.Base;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.context.annotation.Primary;
5 import org.springframework.data.domain.Page;
6 import org.springframework.data.domain.Pageable;
7 import org.springframework.http.HttpStatus;
8 import org.springframework.stereotype.Component;
9 import org.springframework.web.client.HttpClientErrorException.NotFound;
10 import com.bakery.demo.repository.Base.BaseRepository;
11 import jakarta.transaction.Transactional;
12
13 @Component
14 @Primary
15 public abstract class BaseService<T> {
16     @Autowired
17     private BaseRepository<T> repository;
18
19     public Page<T> findAll(Pageable pageable) {
20         return repository.findAll(pageable);
21     }
22
23     public T findOne(Integer id) {
24         return repository.findById(id).orElseThrow(
25             () -> NotFound.create(HttpStatus.valueOf(404), "Entidade não encontrada!", null, null, null));
26     }
27
28     @Transactional
29     public T save(T entity) {
30         return repository.save(entity);
31     }
32
33     @Transactional
34     public void delete(Integer id) {
35         repository.deleteById(id);
36     }
37 }

```

Fonte: Autoria própria (2023).



Figura 4 - BaseController

```

J BaseController.java X
demo > src > main > java > com > bakery > demo > controller > base > J BaseController.java
1 package com.bakery.demo.controller.base;
2
3 import org.springframework.data.domain.Pageable;
4 import org.springframework.http.ResponseEntity;
5 import org.springframework.web.bind.annotation.DeleteMapping;
6 import org.springframework.web.bind.annotation.GetMapping;
7 import org.springframework.web.bind.annotation.PathVariable;
8 import org.springframework.web.bind.annotation.PostMapping;
9 import org.springframework.web.bind.annotation.PutMapping;
10 import org.springframework.web.bind.annotation.RequestBody;
11 import org.springframework.web.bind.annotation.RestController;
12
13 import com.bakery.demo.service.Base.BaseService;
14
15 @RestController
16 public abstract class BaseController<T> {
17     private final BaseService<T> service;
18
19     public BaseController(BaseService<T> service) {
20         this.service = service;
21     }
22
23     @GetMapping("")
24     public ResponseEntity<Iterable<T>> getPage(Pageable pageable){
25         return ResponseEntity.ok(service.findAll(pageable));
26     }
27
28     @GetMapping("/{id}")
29     public ResponseEntity<T> getOne(@PathVariable Integer id){
30         return ResponseEntity.ok(service.findOne(id));
31     }
32
33     @PutMapping("")
34     public ResponseEntity<T> update(@RequestBody T updated){
35         return ResponseEntity.ok(service.save(updated));
36     }
37
38     @PostMapping("")
39     public ResponseEntity<T> create(@RequestBody T created){
40         return ResponseEntity.ok(service.save(created));
41     }
42
43     @DeleteMapping("/{id}")
44     public ResponseEntity<String> delete(@PathVariable Integer id){
45         service.delete(id);
46         return ResponseEntity.ok("Ok");
47     }
48 }

```

Fonte: Autoria própria (2023).

Figura 5 - BaseRepository

```

J BaseRepository.java X
demo > src > main > java > com > bakery > demo > repository > Base > J BaseRepository.java
1 package com.bakery.demo.repository._Base;
2 import org.springframework.data.repository.NoRepositoryBean;
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5 @NoRepositoryBean
6 public interface BaseRepository<T> extends JpaRepository<T, Integer> {
7 }
8

```

Fonte: Autoria própria (2023).

Figura 6 - Exemplo de Fluxo de Vendas

```

46
47 @Transactional
48 public VendaDTO comprarProdutos(FluxoVendaPayloadDTO controllerPayload) {
49     Cliente cliente = cs.findOne(controllerPayload.getClientId());
50     Set<Produto> produtoList = new HashSet<Produto>();
51
52     for (ProdutoFluxoVendaDTO produto : controllerPayload.getProdutos()) {
53         Produto databaseProduct = ps.findOne(produto.getIdProduto());
54
55         if (databaseProduct.getQtdest() - produto.getQtdProduto() < 0) {
56             throw new Error("Produto sem estoque: " + databaseProduct.getNomepro() + ". Quantidade em estoque: " + databaseProduct.getQtdest());
57         }
58
59         databaseProduct.setQtdest(databaseProduct.getQtdest() - produto.getQtdProduto());
60         ps.save(databaseProduct);
61
62         produtoList.add(databaseProduct);
63     }
64
65     Venda newVenda = new Venda();
66     newVenda.setCliente(cliente);
67     newVenda.setDatven(LocalDate.now());
68     newVenda.setQtdven(produtoList.size());
69     newVenda.setVlrven(BigDecimal.valueOf(produtoList.stream().collect(Collectors.summingDouble(pd -> pd.getVlrpro(), doubleValue()))).doubleValue());
70     newVenda.setQtdven(controllerPayload.getProdutos().stream().collect(Collectors.summingInt(pd -> pd.getQtdProduto())));
71     newVenda.setPadaria(padariaS.findOne(controllerPayload.getCnpjPadaria()));
72
73     Venda venda = vs.save(newVenda);
74
75     for (Produto produto : produtoList) {
76         ProdutoVenda pv = new ProdutoVenda();
77         ProdutoVendaKey pvk = new ProdutoVendaKey();
78
79         pvk.setCodpro(produto.getCodpro());
80         pvk.setCodven(venda.getCodven());
81
82         pv.setId(pvk);
83         pv.setProduto(produto);
84         pv.setVenda(venda);
85
86         pvs.save(pv);
87     }
88
89     return vs.findWithProducts(venda.getCodven());
90 }
91
92
93
94
95

```

Fonte: Autoria própria (2023).

## **5 CONCLUSÃO**

Conclui-se com o desenvolvimento deste projeto, tivemos um sistema que possibilita cadastro, atualização, leitura e outras funções, para um gerenciamento de uma padaria chamada “BreadTrack”, para a conclusão desse projeto foi necessário os conhecimentos adquiridos durante o semestre em Banco de Dados II, Programação II e Engenharia de Software I, além do trabalho em equipe realizado entre os estudantes, é de importante salientar que o sistema é essencial para manter um bom ambiente de trabalho em uma padaria, tendo em vista que existem funções que agilizam e facilitam os processos realizados no ambiente de trabalho.

## ***BreadTrack (DBMS)***

### *Abstract*

*This article presents a case study on the implementation of a Database Management System (DBMS) in a bakery, aiming to enhance the management of information and processes related to the establishment. Initially, an analysis of the bakery's needs was conducted, identifying the key requirements for the DBMS. These requirements included the storage and organization of customer data, product information, inventory, sales, supplier orders, and employee records. The database modeling was carried out using a relational approach, defining entities, attributes, and relationships relevant to the bakery. With the modeled database in place, the study progressed by architecting the database with the foundation of software engineering, followed by agile models that facilitate a comprehensive understanding of the software. Finally, the implementation of the DBMS was carried out, creating tables, establishing primary and foreign keys, and populating the database with existing bakery data. Throughout the implementation, tests were conducted to verify the effectiveness of the DBMS. Tools such as Trello were used for project planning, along with PostgreSQL, Eclipse, Visual Paradigm, and DBeaver for modeling and structuring. Version control was managed through GitHub. It is crucial to emphasize the need for a robust software engineering approach, encompassing aspects such as version control, to meet the constantly evolving demands of a renowned event agency.*

*Keywords: Developers. Engineering. Software. Programming. Data. Information. Bakery. DBMS. Implementation.*

## REFERÊNCIAS

ALVEZ, Roberson J. F. **Apostila de Banco de Dados**. São Miguel do Oeste: Unoesc, 2023. Material didático em PDF.

PETRY, Franciele C. **ANÁLISE E PROJETO OO COM UML: INTRODUÇÃO E CASOS DE USO**. 2023. 34 slides. Apresentação de slides.

BARBOSA, Otilia D. **Spring Boot API**. São Miguel do Oeste: Unoesc, 2023. Material didático em PDF.

IBM. **API (JPA)**. Disponível em: [https://www.ibm.com/docs/pt-br/was/8.5.5?topic=SSEQTP\\_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/cejb\\_persistence.htm](https://www.ibm.com/docs/pt-br/was/8.5.5?topic=SSEQTP_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/cejb_persistence.htm). Acesso em: 14 nov, 2023.

IBM. **O que é Java Spring Boot?**. Disponível em: <https://www.ibm.com/br-pt/topics/java-spring-boot>. Acesso em: 07 nov, 2023.

PostgreSQL. **Documentation**. Disponível em: <https://www.postgresql.org/docs/current/sql-createview.html>. Acesso em: 09 nov, 2023.

PETRY, Franciele C. Modelagem **UML: DIAGRAMA DE CLASSES**. 2023. 26 slides. Apresentação de slides.

BARBOSA, Otilia D. **Anotações, JavaDoc e JUnit**. São Miguel do Oeste: Unoesc, 2023. Material didático em PDF.

VENTURA, Plinio. **Entendendo definitivamente o que é um Caso de Uso**. Disponível em: <https://www.ateomomento.com.br/o-que-e-caso-de-uso/>. Acesso em: 18 nov, 2023.

LISBOA, Paulo. **Como Criar um Banco de Dados Em Java**. Disponível em: [https://awari.com.br/como-criar-um-banco-de-dados-em-java/?utm\\_source=blog&utm\\_campaign=projeto+blog&utm\\_medium=Como%20Criar%20um%20Banco%20de%20Dados%20Em%20Java](https://awari.com.br/como-criar-um-banco-de-dados-em-java/?utm_source=blog&utm_campaign=projeto+blog&utm_medium=Como%20Criar%20um%20Banco%20de%20Dados%20Em%20Java). Acesso em: 17 nov, 2023.

Autor Desconhecido: **Guia de tipos de diagramas UML: aprenda sobre todos os tipos de diagramas UML com exemplos**. Disponível em: <https://creately.com/blog/pt/diagrama/guia-de-tipos-de-diagramas-uml-aprenda-sobre-todos-os-tipos-de-diagramas-uml-com-exemplos/>. Acesso em: 17 nov, 2023.

SELEM, Cristiane. **Engenharia de Software em Destaque: Diagramas Comportamentais da UML**. Disponível em: <https://www.estrategiaconcursos.com.br/blog/engenharia-software-diagramas-comportamentais-uml/>. Acesso em: 23 nov, 2023.

CATUNDA, Heitor. **TRIGGER EM SQL: APLICAÇÕES, EXEMPLOS E BOAS PRÁTICAS**. Disponível em: <https://www.hashtagtreinamentos.com/trigger-em-sql>. Acesso em: 24 nov, 2023.

Autor Desconhecido. **Gerenciamento de padaria: 6 dicas para melhorar o seu empreendimento.** Disponível em: <https://gdoor.com.br/gerenciamento-de-padaria/>. Acesso em: 13 nov, 2023.

Autor Desconhecido: **10 dicas de gestão para ter uma padaria de sucesso.** Disponível em: <https://massamadreblog.com.br/know-how/info-tecnicas/6-dicas-de-gestao-para-ter-uma-padaria-de-sucesso/>. Acesso em: 13 nov, 2023.