BreadTrack

SGBD para uma Padaria

Trabalho Integrado – Banco de Dados II, Engenharia de Software I e Programação II

Arthur Kochem, Arthur Gruber,
Bruno Konzen e Leonardo Costa

4ª Fase – Ciência da Computação

São Miguel do Oeste, novembro de 2023

RECAPTULAÇÃO (BD1)

POR QUE ESCOLHER A PADARIA?

- Facilidade de **acesso** à **informações** funcionais;
- **Como funciona o sistema no papel** de vendas, compras, produção, etc...

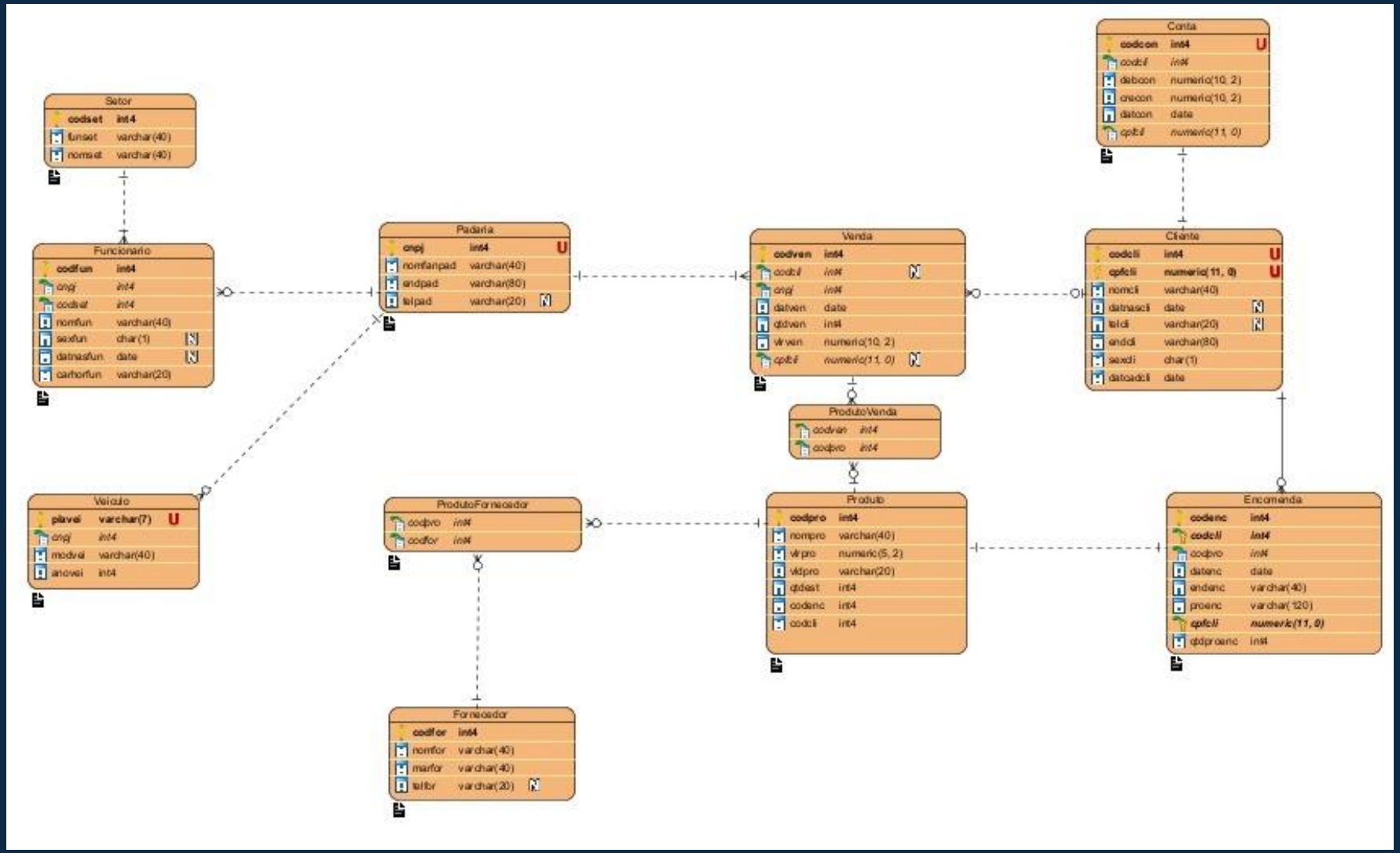
OBJETIVO

- **Desenvolver e implementar um sistema gerenciador de banco de dados para uma padaria**, utilizando o PostgreSQL como SGBD, a fim de otimizar as operações internas, melhorar o controle de estoque, registrar vendas, gerenciar informações de clientes e proporcionar uma gestão eficiente e integrada para a empresa.

REQUISITOS

- **Gerenciamento de produtos:** Cadastro de produtos com informações como nome, descrição. Controle de estoque, incluindo o registro de entrada e saída de produtos.
- **Sistema de pedidos:** Registro e gerenciamento de pedidos de clientes. Opção de criar pedidos personalizados.
- **Gestão de clientes e funcionarios:** Cadastro de clientes com informações como nome, endereço, telefone e histórico de compras.
- **Agendamento de encomendas: :** Possibilidade de agendar a produção de produtos específicos para datas futuras.
- **Integração com delivery:** Integração com aplicativos de delivery para receber pedidos e gerenciá-los.

INICIO DO PROJETO (BD1)



OBJETIVO

- **BreadTrack >** Gestão de entregas, pedidos, vendas e compras de uma padaria.

- **Implementação do projeto de banco**, com documentação em Java, utilizando o **JavaDOC** e também utilizando conceitos abordados em **Engenharia de Software**.

INTRODUÇÃO (FERRAMENTAS)

- **Trello >** Organização do projeto. Ferramenta visual que possibilita ao time o gerenciamento de qualquer tipo de projeto, fluxo de trabalho ou monitoramento de tarefas.

- **Eclipse e Visual Studio Code >** Nossos ambientes de desenvolvimento.

- **Visual Paradigm >** Programa de modelagem do banco. Fornece Scripts de tabelas, etc...

- **dBeaver >** Administração do banco de dados da padaria.

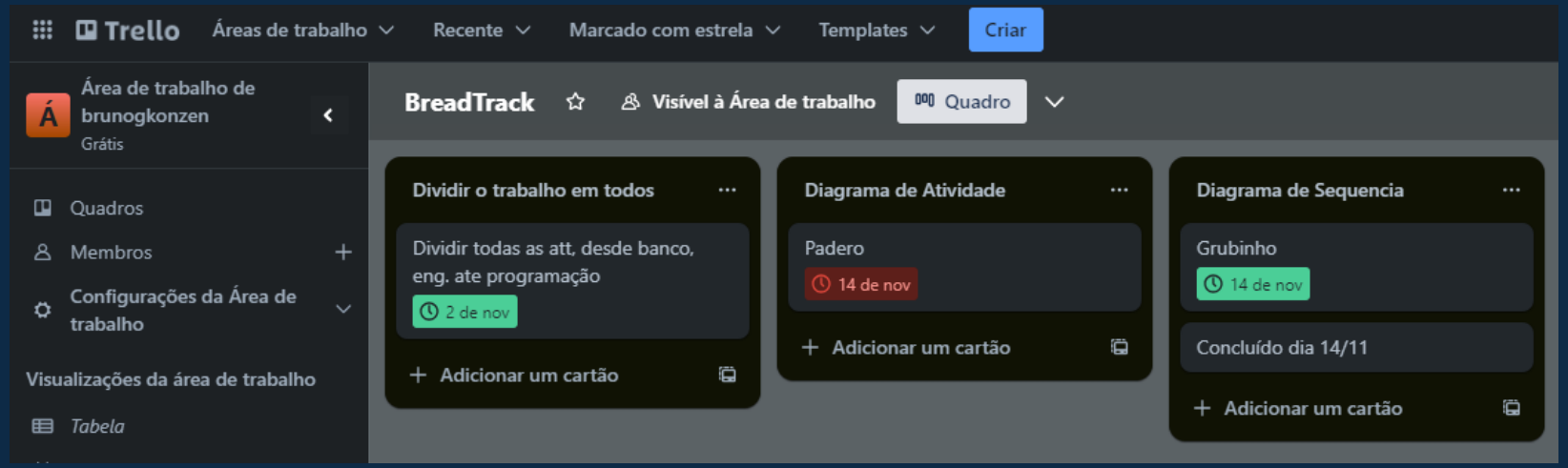
INTRODUÇÃO (FERRAMENTAS)

- **GitHub >** Versionamento do projeto.

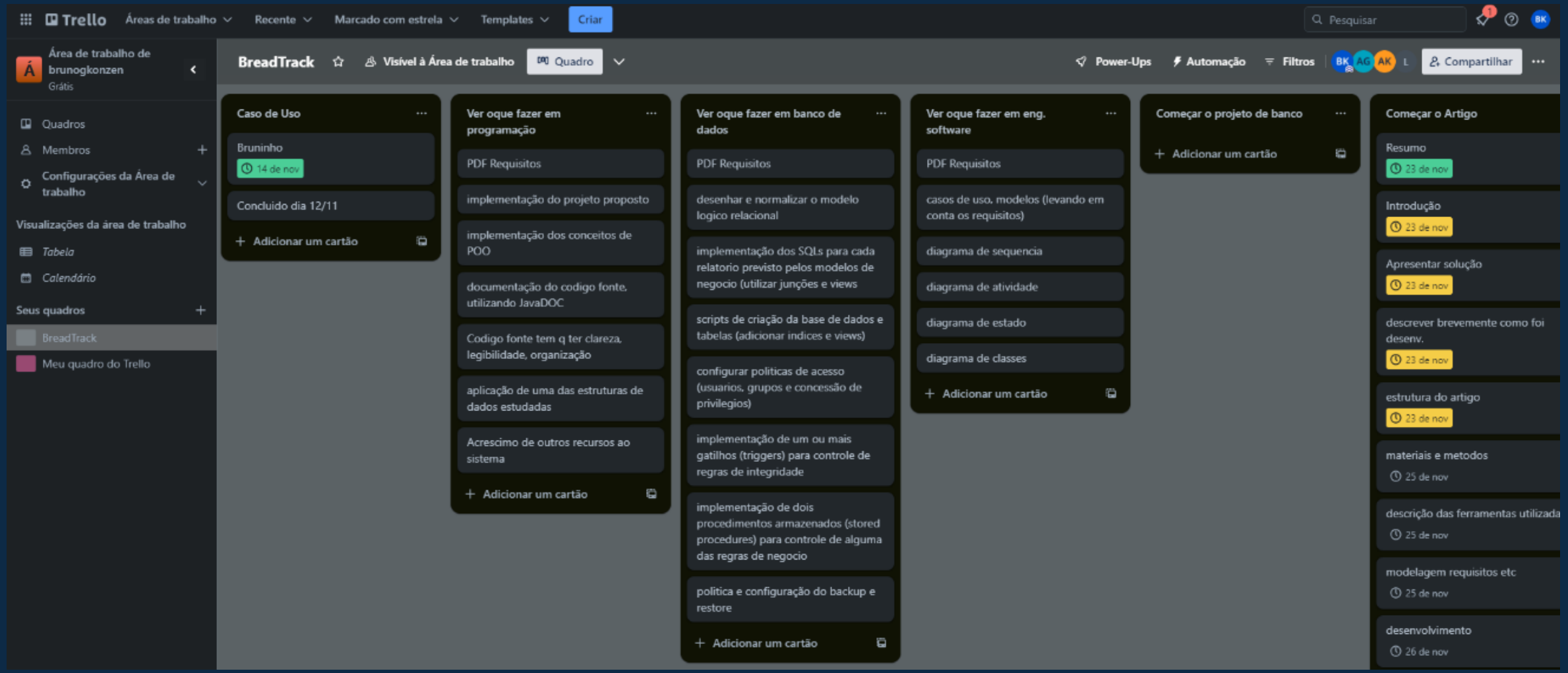
- Visamos colocar em **prática** todo **conhecimento obtido** nos **componentes curriculares**, para obter um **projeto sólido e estável**.

DESENVOLVIMENTO

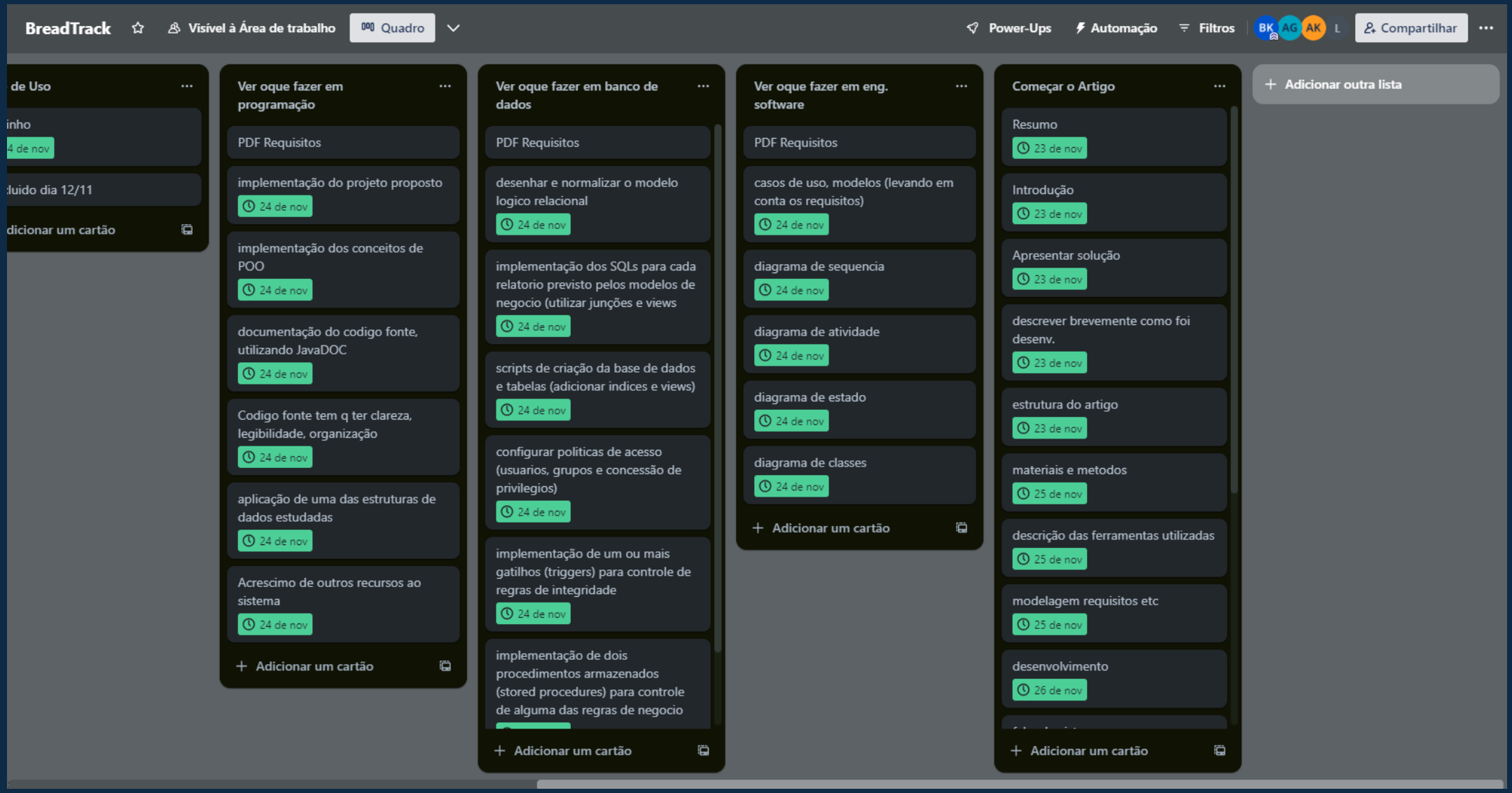
- Depois de apresentar brevemente os requisitos, decidimos separar todas as nossas tarefas. Primeiramente revisando o que cada componente queria que fosse feito. Após isso realizamos a inserção das tarefas no Trello no modelo Kanban.



DESENVOLVIMENTO

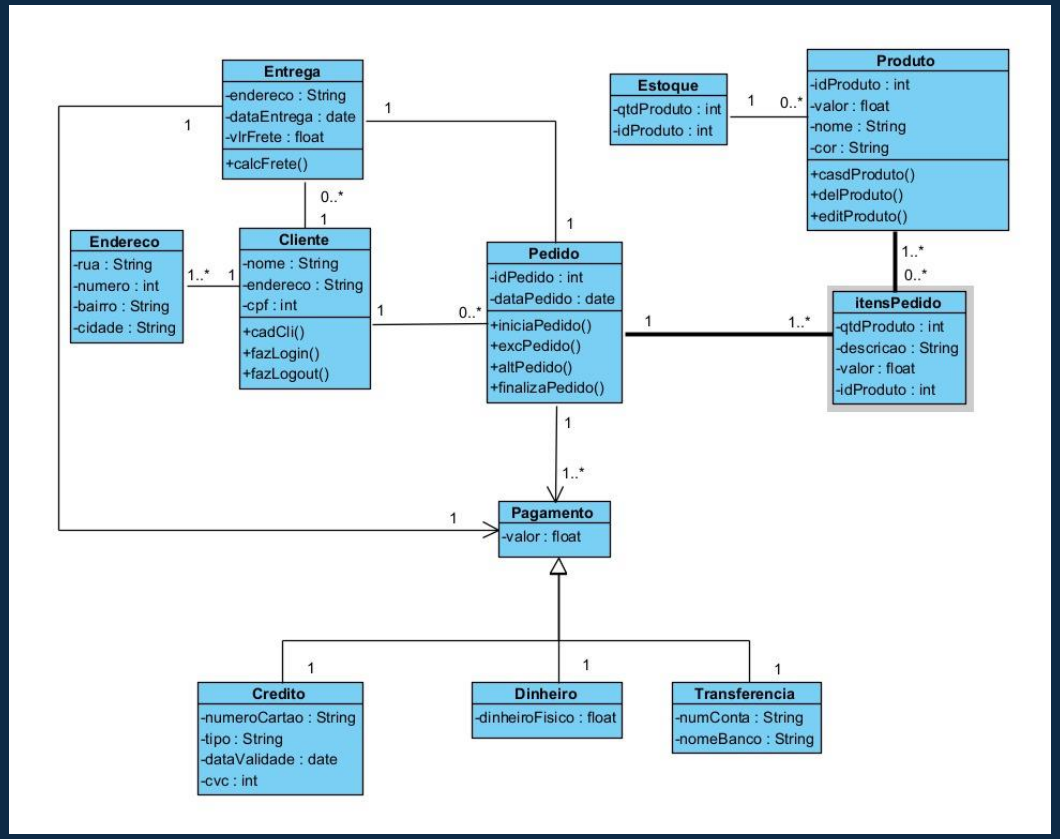


DESENVOLVIMENTO

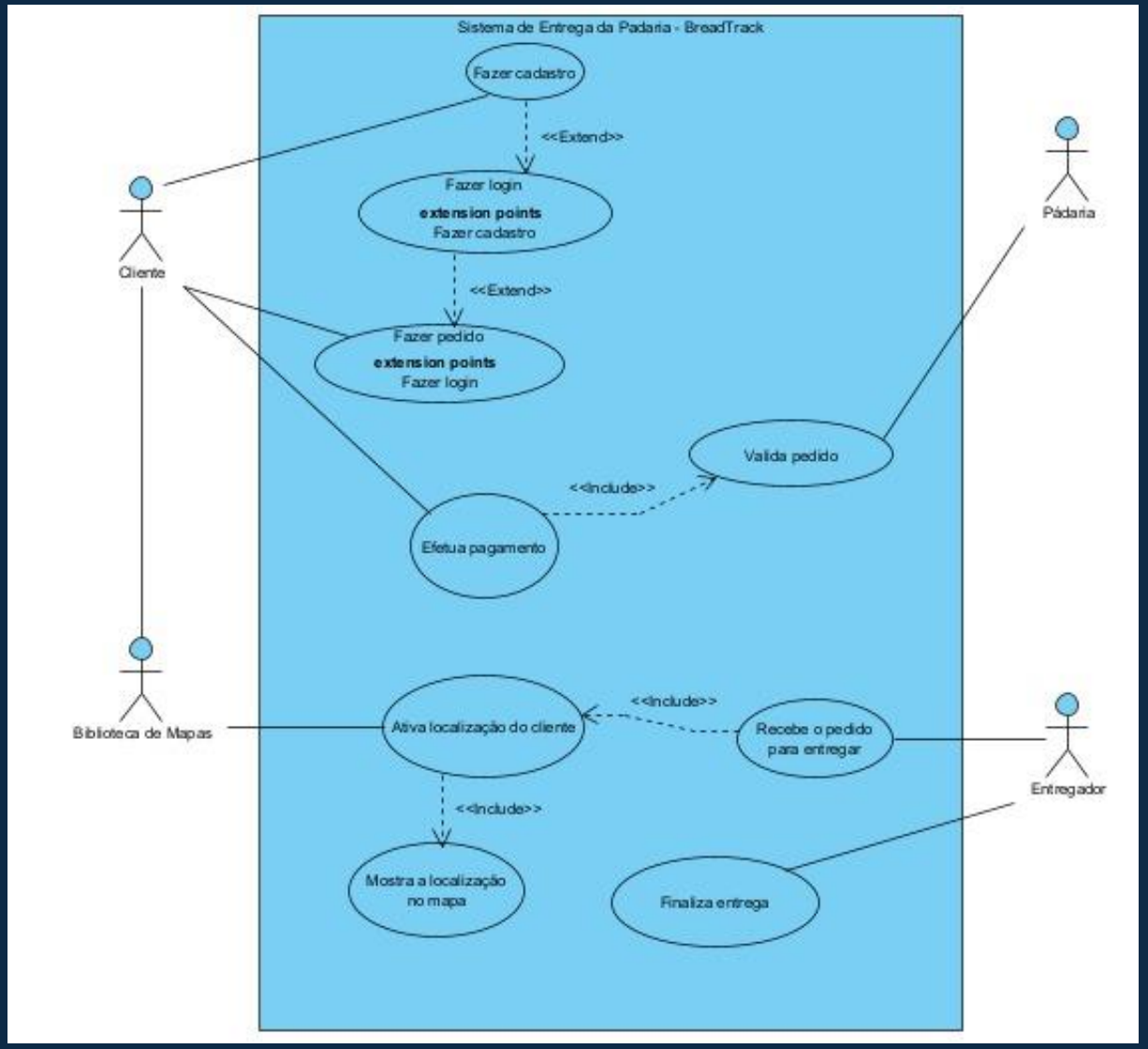


DIAGRAMAS

- Levando em conta os mesmos requisitos já apresentados, começamos modelando nossos diagramas.



DIAGRAMAS


Sistema de Entrega da Padaria - BreadTrack
Fazer cadastro
<<Extend>>
Fazer login
extension points
Fazer cadastro
<<Extend>>
Fazer pedido
extension points
Fazer login
Valida pedido
<<Include>>
Efetua pagamento
Cliente
Pádaria
Biblioteca de Mapas
Ativa localização do cliente
<<Include>>
Recebe o pedido para entregar
Entregador
<<Include>>
Mostra a localização no mapa
Finaliza entrega

BANCO DE DADOS

- Com a modelagem concluída e tarefas já estabelecidas, foi dado inicio 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.

```
create user Arthur with login createdb password '1';
create user Bruno with login createdb password '2';
create user ArthurG with login createdb password '3';
create user Leo with login createdb password '4';

create group consulta;
create group operador;

grant select on estoque to consulta;
grant select, insert, delete, update on estoque, produto to operador;

grant operador to Leo;
grant consulta to Arthut;
grant consulta to Bruno;
grant consulta to ArthurG;
```

BANCO DE DADOS

- 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.

```
1    CREATE INDEX Cliente_codigo_IDX ON Cliente(codcli);
2    CREATE INDEX Cliente_cpf_IDX ON Cliente(cpfcli);
3    CREATE INDEX Encomenda_codigo_IDX ON Encomenda(codenc);
4    CREATE INDEX Fornecedor_codigo_IDX ON Fornecedor(codfor);
5    CREATE INDEX Funcionario_codigo_IDX ON Funcionario(codfun);
6    CREATE INDEX Produto_codigo_IDX ON Produto(codpro);
7    CREATE INDEX Veiculo_placa_IDX ON Veiculo(plavei);
8    CREATE INDEX Setor_codigo_IDX ON Setor(codset);
```

RELATÓRIOS

```
--1) Relatório de clientes com o nome, sexo e idade em ordem crescente
de nome. Relacionar  somente clientes cadastrados antes de 2023;
```

	nomcli	sexcli	idade
1	Ana Oliveira	F	31
2	JoĆ£o Silva	M	33
3	Lucas Santos	M	0
4	Marcelo Freitas	M	19
5	Maria Souza	F	38
6	Matheus Machado	M	33

```sql
CREATE VIEW view_clientes AS
SELECT nomcli, sexcli, EXTRACT(YEAR FROM AGE(datnascli)) AS idade
FROM Cliente
WHERE datcadcli < '2023-01-01'
ORDER BY nomcli ASC;

select * from view_clientes vc;
```

RELATÓRIOS

--2) Relação de produtos(nome e descrição) vendidos nos meses pares de 2022. Ordene o relatório pelo nome do produto de forma ascendente;

	nompro	proenc
1	Bolo de Chocolate	Bolo de Chocolate
2	Docinhos	Docinhos Sortidos
3	PĆ£o FrancĆRs	PĆ£es franceses
4	Salgado Fritos	Salgados Fritos Sortidos

```
CREATE VIEW view_vendas_meses_pares AS
SELECT p.nompro, e.proenc
FROM Produto p
INNER JOIN Encomenda e ON p.codpro = e.codpro
WHERE EXTRACT(YEAR FROM e.datenc) = 2022
AND EXTRACT(MONTH FROM e.datenc) % 2 = 0
ORDER BY p.nompro ASC;

select * from view_vendas_meses_pares;
```

RELATÓRIOS

```
--3) Relação dos top 10 produtos vendidos em 2023;
```

	nompro	total_vendas
1	PĆ£o FrancĆRs	314
2	Enroladinho	190
3	Docinhos	183
4	PĆ£o de Milho	144
5	Salgado Fritos	90
6	PĆ£o de Mel	76
7	Bolo de Chocolate	74

```sql
CREATE VIEW view_top10 AS
SELECT p.nompro, SUM(v.qtdven) AS total_vendas
FROM Produto p
INNER JOIN ProdutoVenda pv ON p.codpro = pv.codpro
INNER JOIN Venda v ON pv.codven = v.codven
WHERE EXTRACT(YEAR FROM v.datven) = 2023
GROUP BY p.nompro
ORDER BY total_vendas DESC
LIMIT 10;
```

RELATÓRIOS

--4) Relação de meses, quantidade total de vendas, valor total de vendas por mês. Relacionar somente meses com quantidade de vendas acima de 100. Ordenar o relatório do mês com o maior valor(R$) em vendas para o mês com menos vendas.

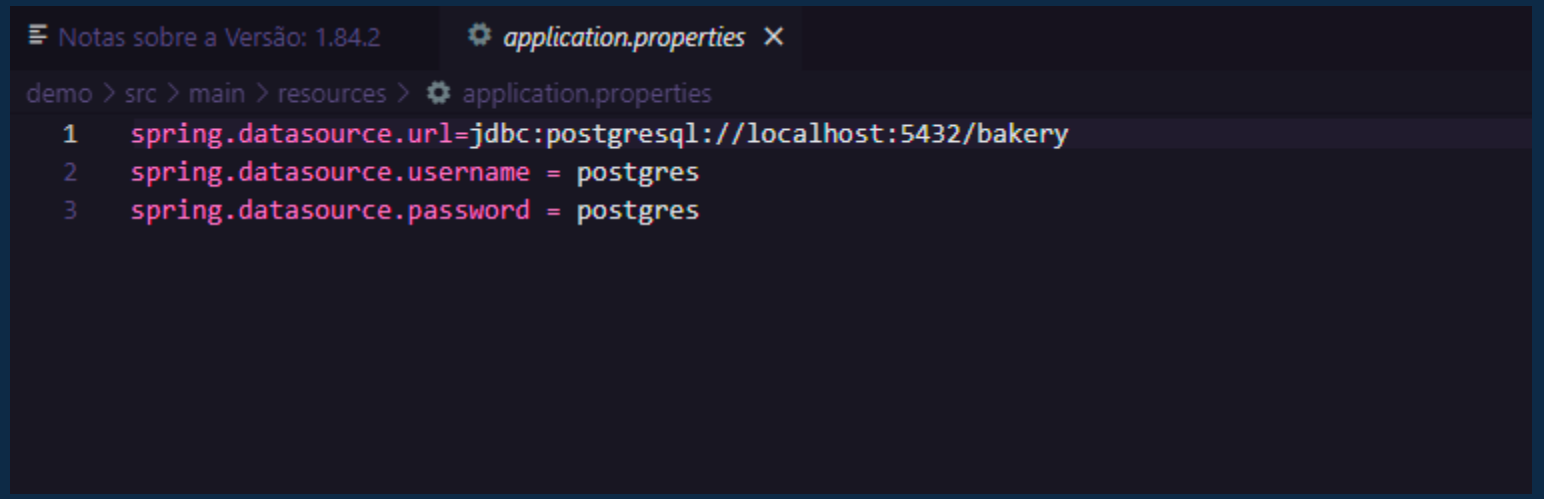
	mes	quantidade_vendas	valor_total_vendas
1	2	50	1.288
2	10	54	1.009,8
3	1	40	776,4
4	4	28	513,8
5	9	14	490
6	7	14	350
7	6	14	147
8	3	14	147
9	8	12	126

```sql
CREATE VIEW view_mes_maior_venda AS
SELECT EXTRACT(MONTH FROM v.datven) AS mes, COUNT(*) AS quantidade_vendas, SUM(v.vlrven) AS valor_total_vendas
FROM Venda v
where vlrven > 10
GROUP BY mes
ORDER BY valor_total_vendas DESC, quantidade_vendas ASC;

select * from view_mes_maior_venda
```

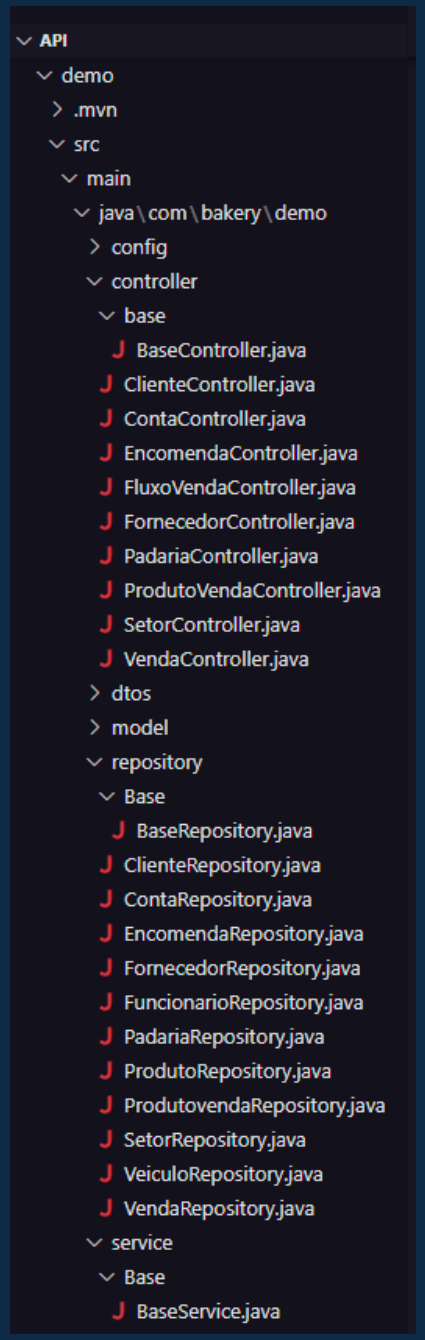
PROGRAMAÇÃO

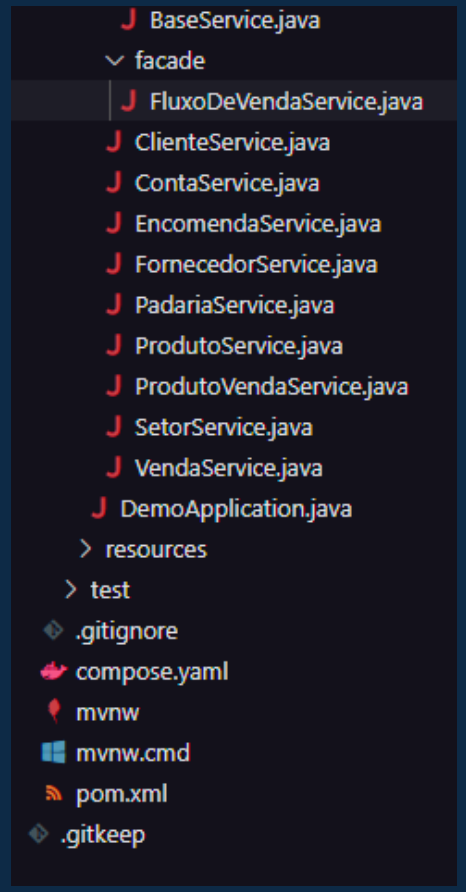
- Após modelar e normalizar todo o banco, seguimos para sua implementação em Java.



- Arquivo .properties para fazer a conexão com o banco de dados.

PROGRAMAÇÃO (API)





PROGRAMAÇÃO (pom.xml)

pom.xml

demo > pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
```

PROGRAMAÇÃO (FLUXO DE VENDAS)

```
46
47      @Transactional
48      public VendaDTO comprarProdutos(FluxoVendaPayloadDTO controllerPayload) {
49          Cliente cliente = cs.findOne(controllerPayload.getClienteId());
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.getNompro() + ". 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
66          Venda newVenda = new Venda();
67          newVenda.setCliente(cliente);
68          newVenda.setDatven(LocalDate.now());
69          newVenda.setQtdven(produtoList.size());
70          newVenda.setVlrven(BigDecimal.valueOf(produtoList.stream().collect(Collectors.summingDouble(pd -> pd.getVlrpro().doubleValue())).doubleValue()));
71          newVenda.setQtdven(controllerPayload.getProdutos().stream().collect(Collectors.summingInt(pd -> pd.getQtdProduto())));
72          newVenda.setPadaria(padariaS.findOne(controllerPayload.getCnpjPadaria()));
73
74          Venda venda = vs.save(newVenda);
75
76
77          for (Produto produto : produtoList) {
78              Produtovenda pv = new Produtovenda();
79              ProdutovendaKey pvk = new ProdutovendaKey();
80
81              pvk.setCodpro(produto.getCodpro());
82              pvk.setCodven(venda.getCodven());
83
84              pv.setId(pvk);
85              pv.setProduto(produto);
86              pv.setVenda(venda);
87
88
89              pvs.save(pv);
90          }
91
92          return vs.findWithProducts(venda.getCodven());
93      }
94  }
95
```

PROGRAMAÇÃO (SERVICE)

BaseService.java

demo > src > main > java > com > bakery > demo > service > Base > BaseService.java

```java
package com.bakery.demo.service.Base;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Primary;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.http.HttpStatusCode;
import org.springframework.stereotype.Component;
import org.springframework.web.client.HttpClientErrorException.NotFound;
import com.bakery.demo.repository.Base.BaseRepository;
import jakarta.transaction.Transactional;

@Component
@Primary
public abstract class BaseService<T> {
    @Autowired
    private BaseRepository<T> repository;

    public Page<T> findAll(Pageable pageable) {
        return repository.findAll(pageable);
    }

    public T findOne(Integer id) {
        return repository.findById(id).orElseThrow(
                () -> NotFound.create(HttpStatusCode.valueOf(404), "Entidade nĆ£o encontrada!", null, null, null));
    }

    @Transactional
    public T save(T entity) {
        return repository.save(entity);
    }

    @Transactional
    public void delete(Integer id) {
        repository.deleteById(id);
    }
}
```

PROGRAMAÇÃO (REPOSITORY)

BaseRepository.java

demo > src > main > java > com > bakery > demo > repository > Base > BaseRepository.java

```java
package com.bakery.demo.repository.Base;
import org.springframework.data.repository.NoRepositoryBean;
import org.springframework.data.jpa.repository.JpaRepository;

@NoRepositoryBean
public interface BaseRepository<T> extends JpaRepository<T, Integer> {
}

```

PROGRAMAÇÃO (CONTROLLER)

BaseController.java

demo > src > main > java > com > bakery > demo > controller > base > BaseController.java

```java
package com.bakery.demo.controller.base;

import org.springframework.data.domain.Pageable;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.bakery.demo.service.Base.BaseService;

@RestController
public abstract class BaseController<T> {
    private final BaseService<T> service;

    public BaseController(BaseService<T> service) {
        this.service = service;
    }

    @GetMapping("")
    public ResponseEntity<Iterable<T>> getPage(Pageable pageable){
        return ResponseEntity.ok(service.findAll(pageable));
    }

    @GetMapping("/{id}")
    public ResponseEntity<T> getOne(@PathVariable Integer id){
        return ResponseEntity.ok(service.findOne(id));
    }

    @PutMapping("")
    public ResponseEntity<T> update(@RequestBody T updated){
        return ResponseEntity.ok(service.save(updated));
    }

    @PostMapping("")
    public ResponseEntity<T> create(@RequestBody T created){
        return ResponseEntity.ok(service.save(created));
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<String> delete(@PathVariable Integer id){
        service.delete(id);
        return ResponseEntity.ok("Ok");
    }
```

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.

PONTOS POSITIVOS E NEGATIVOS

Autonomia pra fazer o trabalho, escolha de tema livre, inserção de temas pertinentes sobre a área de gerenciamento de dados;

Utilização prática dos conteúdos abordados nas matérias;

O projeto pode apresentar falhas e/ou erros, por mais que o tempo foi suficiente, não conseguimos deixar 100% concluído, faltando o desenvolvimento front-end.

Agradecemos a oportunidade de trabalhar com o Banco de Dados, Engenharia de Software e Programação, com toda experiencia ganha. E mais uma vez destacamos nossos erros e deixamos claro que estamos preparados para seguir em frente com o projeto e cada vez mais buscando sua perfeição.

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. Acesso em: 14 nov, 2023.

REFERÊNCIAS

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.

ALGUMA DÚVIDA?

E-mail: arthurcg21@gmail.com

Telefone: (49) 9 9157-1654

E-mail: arthurkochem12@gmail.com

Telefone: (49) 9 8501-7031

E-mail: brunogkonzen@hotmail.com

Telefone: (49) 9 8400-4883

E-mail: laccosta1242@gmail.com

Telefone: (49) 9 9172-0437

ANEXOS (LINK GITHUB)

ANEXOS (ARTIGO)