INSTUTO FEDERAL DO ESPÍRITO SANTO

BRUNO DELL'ORTO MERGH

GABRIELA PONTES BREDER

TRABALHO DE PROGRAMAÇÃO ORIENTADA A OBJETOS 2

SERRA

2015

Sumário

1. Minimundo e descrição da utilização do padrão MVC

A padaria do senhor Paulo esta tecnologicamente atrasada. O sistema de controle de conta é a moda antiga: papel e caneta. Ele negocia com os fornecedores prazo para pagar suas próprias compras no início do mês seguinte. Acontece que a cada fim do mês é sempre o mesmo problema: o senhor Paulo tem dificuldades para calcular quanto tem a receber e quanto tem a pagar somente olhando as anotações em papel. Cansado desta situação, ele deseja um sistema para obter um melhor gerenciamento.

O sistema deve fazer o cadastro de clientes, fornecedores, produtos, compras e vendas (pacote CDP – Componente de Domínio do Problema), e também gerar relatórios como contas a receber por cliente e a pagar por fornecedor, valor bruto e lucro das vendas por produto, além de retornar um relatório informando a situação atual do estoque (A classe dos relatórios ficará no CGT – Camada de Gerência de Tarefas).

Dos clientes deseja-se registrar código identificador, nome, endereço, telefone e a data de seu cadastro. Clientes podem ser de dois tipos: pessoas físicas ou jurídicas. De pessoas físicas deseja-se registrar o CPF, enquanto de pessoas jurídicas deseja-se registrar CNPJ.

Dos fornecedores deseja-se registrar código identificador, nome, endereço, telefone e CNPJ.

Os produtos são cadastrados no sistema com as seguintes informações: código, descrição, estoque mínimo, quantidade atual em estoque, valor de custo e percentual de lucro. O valor de venda de um produto é calculado com base no valor de custo e percentual de lucro:

<valor de venda> = <valor de custo> * (1.0 + <percentual de lucro>)

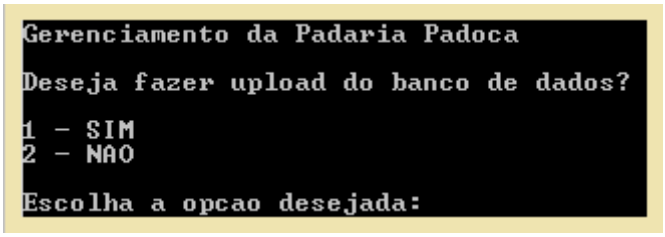
Das compras efetuadas pela padaria junto aos fornecedores são registrados o número da nota fiscal, o fornecedor, a data da compra, o produto comprado e a quantidade.

Das vendas efetuadas pela padaria aos clientes são registrados o cliente, a data da venda, o produto vendido e a quantidade vendida.

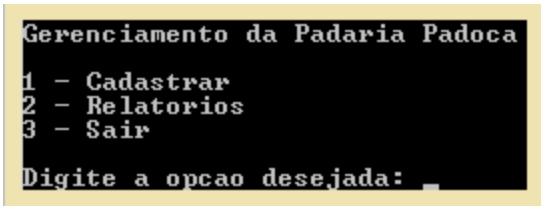
Após todos os dados registrados, o sistema deve gerar os seguintes relatórios:

- Total a pagar por fornecedor.
- Total a receber por cliente.
- Vendas e lucro por produto.
- Estado do estoque, incluindo alertas de estoque abaixo do mínimo.

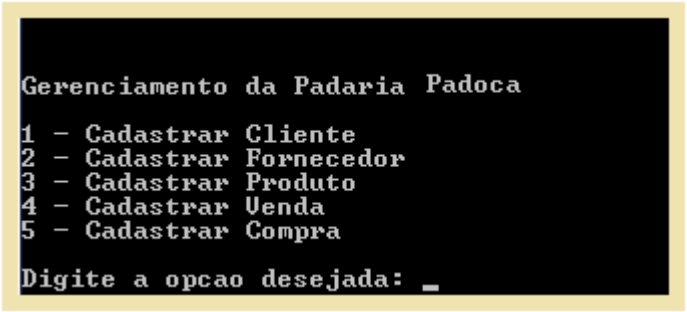
Foi combinado com o Sr. Paulo que os cadastros poderiam ser lidos de arquivos “.txt” logo quando abre o programa, e também serem registrados na hora, através de um menu iterativo de fácil entendimento. Veja:



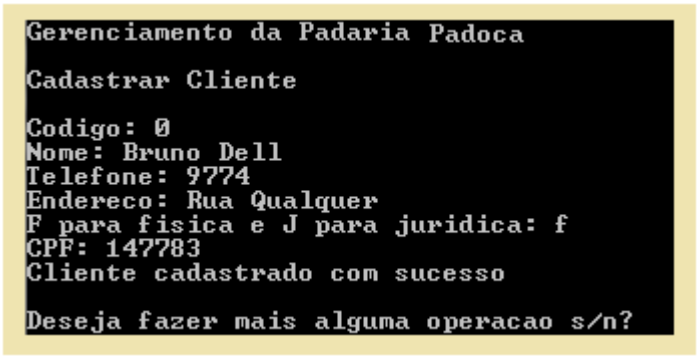
Após a escolha da opção desejada, o programa seguirá para a próxima tela.



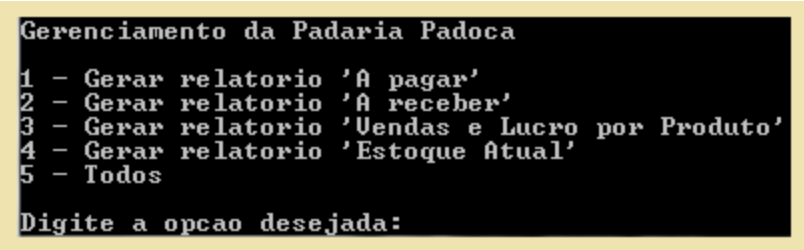
Ao escolher a opção digitando o número “1” os funcionários teriam acesso a seguinte tela do menu:



Os funcionários têm as seguintes opções acima para escolher e fazer os respectivos cadastros, por exemplo escolhendo novamente a opção “1”:



Após o cadastro, existe a opção de fazer mais alguma operação que te retorna ao menu principal. Ao escolher agora a opção de número ‘2’, o Sr. Paulo terá na tela as opções de gerar os relatórios, como na imagem seguinte:



A interface ficará no CIH – Componente de Interação Humana, ou Camada de Apresentação. Já o processamento dos comandos fica no CCI – Componente de Controle de Interação.

Após a escolha de gerar os relatórios os funcionários terão um arquivo no formato ".txt" com os resultados.

Os dados cadastrais dos Clientes, Fornecedores, Produtos, Vendas e Compras terão que seguir o modelo informado abaixo, em um arquivo no formato ".txt", para que o upload do banco seja feita da maneira correta no início do programa. Essa parte de salvar fica no CGD – Componente de gerência de dados.

Os arquivos devem seguir o seguinte formato:

Arquivo de cadastro de clientes:
<código>; <nome>;<telefone>;<endereço>;<tipo(f para físico e j para jurídico)>;<CPF ou CNPJ>;

Arquivo de cadastro de fornecedores:
<nome>;<endereço>;<telefone>;<código>;<CNPJ>;

Arquivo de cadastro de produtos:
<código>;<descrição>;<estoque mínimo>;<quantidade atual>;<custo>;<percentual de luco>;

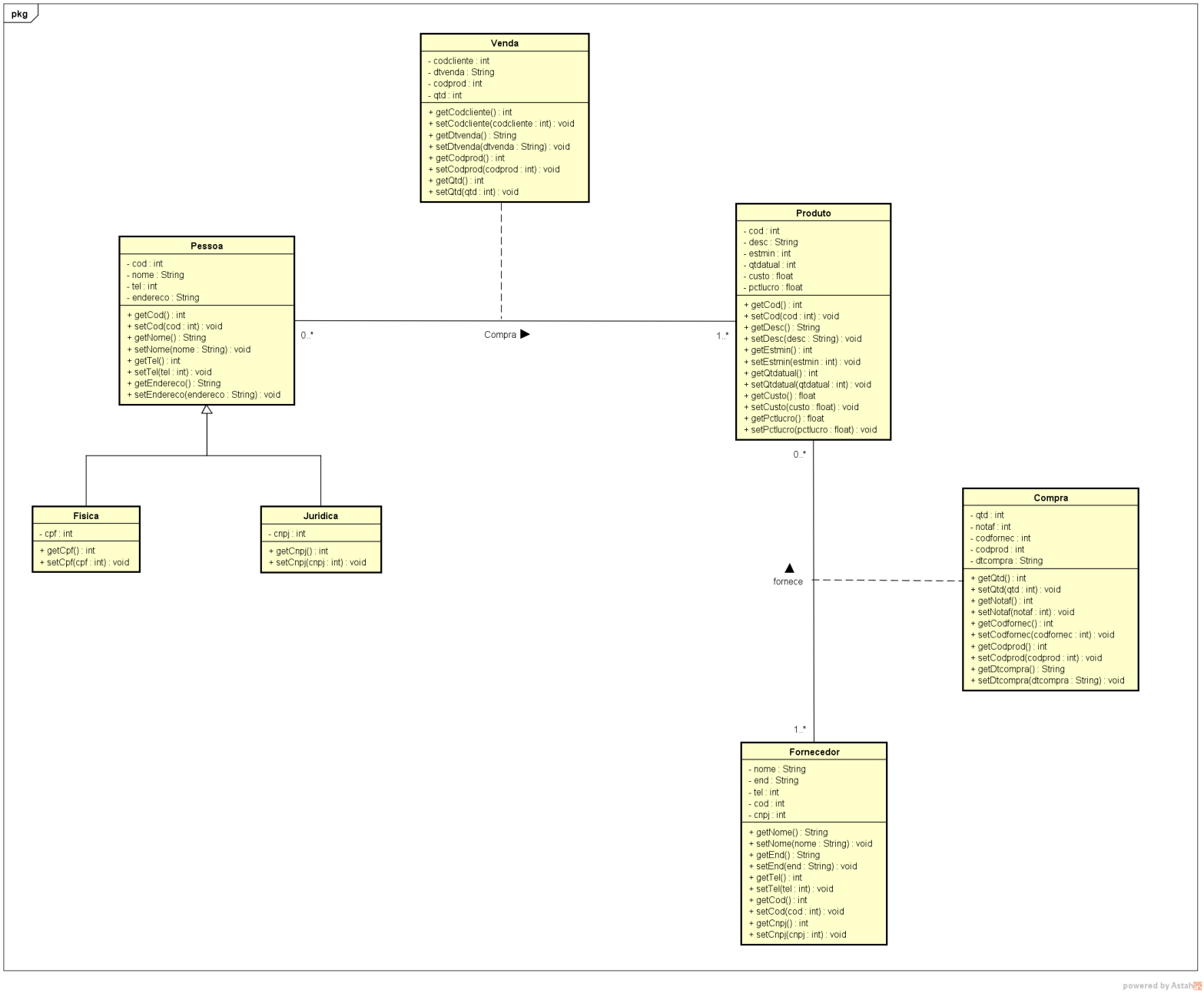
Arquivo de cadastro de vendas:
<código do cliente>;<data>;<código do produto>;<quantidade>;

Arquivo de cadastro de compras:
<quantidade>;<número da nota fiscal>;<código do fornecedor>;<código do produto>;<data da compra>;

Código do cliente, código do fornecedor, código do produto e quantidade devem ser números inteiros.

E por último, escolhendo a opção de número "3", o programa é encerrado.

2. Diagrama de padrões de projeto



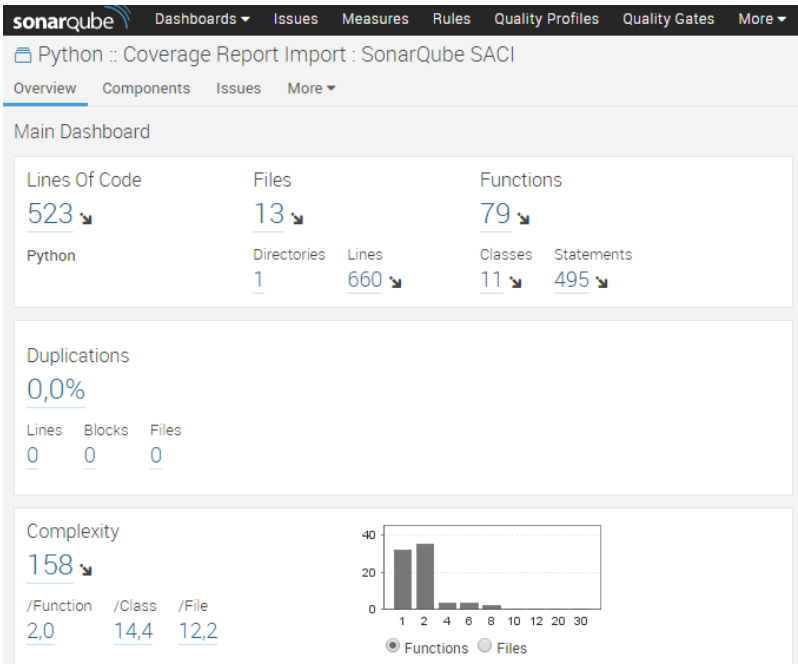
3. Plataforma de desenvolvimento

O programa foi desenvolvido na linguagem *python* versão *3x*.

4. Testes

```
class TestPessoa(unittest.TestCase):
    def setUp(self):
        self.pessoa = Pessoa("0", "Bruno", 27997743714, "Rua qualquer")
        self.produto = Produto(0,"miojo",3,5,3.2,40)
        self.fornecedor = Fornecedor("Casas Bahia","Rua 1", 999, 0, 122)
        self.compra = Compra(1,111,0,0,"12/01/2015")
        self.venda = Venda(0,"13/01/2015",0,1)

    def testAtributosPessoa(self):
        self.assertEqual(self.pessoa.getCod(), "0")
        self.assertEqual(self.pessoa.getNome(), "Bruno")
        self.assertEqual(self.pessoa.getTel(), 27997743714)
        self.assertEqual(self.pessoa.getEnd(), "Rua qualquer")

    def testAtributosProduto(self):
        self.assertEqual(self.produto.getCod(), 0)
        self.assertEqual(self.produto.getDesc(), "miojo")
        self.assertEqual(self.produto.getEstmin(), 3)
        self.assertEqual(self.produto.getQtdatual(), 5)
        self.assertEqual(self.produto.getCusto(), 3.2)
        self.assertEqual(self.produto.getPctlucro(), 40)

    def testAtributosFornecedor(self):
         self.assertEqual(self.fornecedor.getNome(), "Casas Bahia")
         self.assertEqual(self.fornecedor.getEnd(), "Rua 1")
         self.assertEqual(self.fornecedor.getTel(), 999)
         self.assertEqual(self.fornecedor.getCod(), 0)
         self.assertEqual(self.fornecedor.getCnpj(), 122)

    def testAtributosCompra(self):
        self.assertEqual(self.compra.getQtd(), 1)
        self.assertEqual(self.compra.getNotaf(), 111)
        self.assertEqual(self.compra.getCodfornec(), 0)
        self.assertEqual(self.compra.getCodprod(), 0)
        self.assertEqual(self.compra.getDtcompra(), "12/01/2015")

    def testAtributosVenda(self):
        self.assertEqual(self.venda.getCliente(), 0)
        self.assertEqual(self.venda.getDt(), "13/01/2015")
        self.assertEqual(self.venda.getProd(), 0)
        self.assertEqual(self.venda.getQtd(), 1)
```

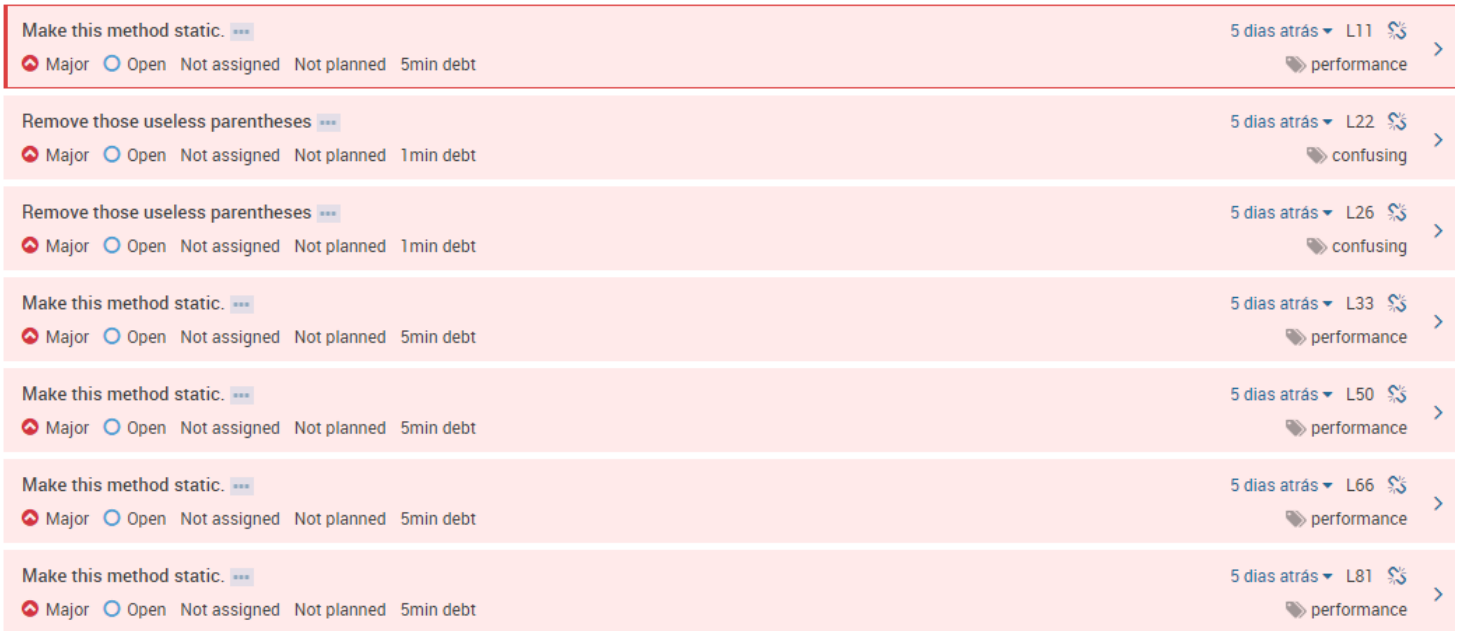
5. Avaliação do código



Com o SonarQube podemos identificar que nosso projeto possui 523 linhas de códigos, 13 arquivos e 79 funções. No canto superior direito, há um campo chamado Technical Debt Ratio (dívida técnica). É um cálculo baseado na metodologia SQUALE (Software Quality Assessment based on Lifecycle Expectations), que tem a ver com a organização de requisitos relacionados à qualidade do código. Dessa forma, o Sonar possui ferramentas que permitem controlar essa dívida, através de reusabilidade de código, portabilidade, segurança, eficiência, etc.

No canto esquerdo, percebe-se que não há duplicação de código. Em contrapartida, a complexidade está em 158, um nível elevado. A complexidade mede a quantidade de caminhos de execução independentes que o código possui. Se ele não contém estruturas de controle senão sequenciais, a complexidade é 1, já que há somente um caminho válido através do código.

À direita da figura, também vemos a taxa de erros, que está em 122, sendo 35 principais e 87 secundários.



Dos erros principais, observamos apenas a presença de parênteses inúteis e a mensagem para definir o método em questão como estático.

Já dos erros secundários, é necessário renomear os métodos e variáveis para corresponder às seguintes expressões regulares em questão. Analisando a primeira, por exemplo, ela diz que o método pode ser aceito desde que seja uma string que comece com letra minúscula, seguida de quantos números ou dígitos forem necessários. No nosso caso, seria apenas colocar a primeira letra minúscula para o erro deixar de existir.