**FEATURE DRIVEN DEVELOPMENT**

**Ana Carolina Scherer, Débora Loro, Matheus Machado, Miquéias Queiroz** **da Silva, Natália Pires**

Colégio Técnico Industrial de Santa Maria– Universidade Federal de Santa Maria(UFSM)

Av. Roraima n° 1000 – 97.105-900 – Santa Maria – RS – Brazil

[schereranacarol@hotmail.com](mailto:schereranacarol@hotmail.com), [deborallopes2002@gmail.com](mailto:deborallopes2002@gmail.com), [matheusmazar@yahoo.com.br](mailto:matheusmazar@yahoo.com.br), [mikeiasqueioz670@gmail.com](mailto:mikeiasqueioz670@gmail.com), natalia.phth@gmail.com

**Resumo.** Este “trabalho/artigo” tem o objetivo de descrever os resultados das pesquisas sobre Feature Driven Development, uma metodologia ágil de desenvolvimento de software que permite desenvolver um sistema de forma rápida e introduzir novas features facilmente, entre muitos outras características. Abordaremos seu surgimento, seus conceitos, membros, etapas, fases, etc. Escrito de forma simples e clara, com o auxílio de exemplos autorais, este é proveniente de pesquisas realizadas pelos alunos, Ana Carolina Scherer, Débora Loro, Matheus Machado, Miquéias Queiroz da Silva e Natália Pires, na matéria de Análise e Projeto de Sistemas Web sobre Feacture Driven Development.

# 1. Feature Driven Development

No ano de 1997, devido à experiência de análise e modelagem orientada por objetos de Peter Coad e o gerenciamento de software de Jeff de Luca, em um grande projeto para o UOB (United Overseas Bank) em Singapura, foi criado o Feature Driven Development, ou em português, Desenvolvimento Guiado por Funcionalidade, também chamado de FDD, é uma metodologia ágil de desenvolvimento e gerenciamento de software que teve sua descrição originalmente publicada em 1999 no livro “Java Modeling in Color with UML” de Peter Coad, porém uma versão mais completa da metodologia foi publicada posteriormente no livro “A Practical Guide to Feature-Driven Development” de Stephen R. Palmer e John M. Felsing. Está metodologia ágil tem como lema: resultados frequentes, tangíveis e funcionais.

O Feature Driven Development é um dos seis métodos ágeis originais, ou seja, que foram criadas antes do manifesto ágil, que procura o desenvolvimento através de funcionalidades. Uma das funcionalidade, como descreve Coad, “é uma função com valor para o cliente que pode ser desenvolvida em duas ou menos semanas”. As funcionalidades são divididas em pequenos blocos, entregues dentro de um período de 2 semanas, que trazem como benefícios a facilidade de compreensão da relação entre funcionalidade, uma descrição simplificada pelo usuário e uma melhor revisão de falhas. Uma funcionalidade pode ser definida através do seguinte modelo:

<Ação> <Resultado> <Objeto>

Exemplo: Calcular o valor de uma compra, onde calcular seria a ação, o valor o resultado, e a compra o objeto.

A integração com o cliente é contínua, ou seja, demonstra-se constantemente as funcionalidades ao cliente fazendo assim com que se obtenha uma maior progresso no projeto.

Outro aspecto importante no FDD é a ênfase na cooperação entre os membros de equipe, o modelo só estará pronto quando todos os membros da equipe entrarem em consenso. Em sua organização, o FDD tem seis papéis principais, dos quais todos estão diretamente envolvidos na realização dos processos da metodologia. São eles:

* **Gerente de projeto:** líder administrativo do projeto e financeiro do projeto, onde possui a palavra final no que diz respeito ao que deve ser feito;
* **Gerente de Desenvolvimento:** responsável por resolver quaisquer divergências que venham ocorrer dentro da equipe. Também é responsável por avaliar se o código realizado pelo time de desenvolvimento está nos padrões do projeto;
* **Arquiteto Chefe:** responsável pela modelagem do projeto;
* **Programador Chefe:** líder de pequenos times de análise, modelagem e desenvolvimento de novas funcionalidades;
* **Dono de Classe:** responsável pela implementação das funcionalidades definidas pelo gerente de projeto;
* **Especialistas do domínio:** responsável por adquirir e transmitir informações a respeito do funcionamento dos requisitos do sistema, podem ser usuários, clientes, patrocinadores ou analistas de negócio.

Outros papéis podem ser incluídos conforme a complexidade do projeto como papéis de apoio (gerente do domínio, gerente de versão, especialista (guru) de linguagem, coordenador de construção, “ferramenteiro” (toolsmith) e administrador de sistema) ou papéis adicionais (testador, desenvolvedores e escritor técnico). Ressalta-se ainda que uma pessoa pode assumir várias funções em um projeto.

**2. Processo Feature Driven Development**

O processo Feature Driven Development é dividido em cinco etapas sequenciais, das quais, as três primeiras pertencem à fase de “Concepção e Planejamento”, que atua com o objetivo de proporcionar um entendimento inicial e abrangente do escopo, possibilitando um planejamento de entregas durante as iterações e, além disso, é onde ocorre a triagem de requisitos. As duas últimas etapas pertencem à fase de “Construção”, que é a fase iterativa e incremental, onde será inserida a prática de modelagem de interação antes do projeto de cada funcionalidade (para que se tenha uma visão mais específica do comportamento de cada requisito).

Por sua simplicidade e objetividade este processo é excelente para um projetos de manutenção por exemplo.

**2.1. Etapas do processo**

As etapas do processo são divididos em suas duas fases, “Concepção e Planejamento” e “Construção”.

**2.1.1.Etapas da fase de “Concepção e Planejamento”:**

* **Etapa de desenvolver um modelo abrangente (Análise orientada por objetos):**

Inicia-se com o detalhamento do domínio do negócio com divisão de áreas que serão modeladas e envolve o desenvolvimento de requisitos, análise orientado por objetos e modelagem lógica de dados. Essa etapa representa a atividade inicial na metodologia e tem como objetivo desenvolver um modelo abrangente que resulte na confecção de diagramas de classe, de sequência e na modelagem de objetos. Nesse processo é necessário uma equipe de modelagem (formada por programadores, líderes e especialistas de domínios) que dará início ao entendimento do conjunto de características que descreverão o resultado final do produto que será desenvolvido (escopo), e requer a orientação de um modelador de objetos experiente, que irá ser o arquiteto líder. O resultado é um modelo de objetos e ou de dados, que guiarão a equipe durante os ciclos de construção. Caso seja necessário serão criados diversos grupos para as diferentes áreas do projeto, onde cada grupo faz sua proposta de modelo que será comparada e revisada por todos grupos. Dos modelos feitos alguns serão escolhidos e utilizados por todos envolvidos durante o desenvolvimento do projeto para servir de orientação para o trabalho, e para esclarecer questões que possam afetar de alguma forma o modelo e o resultado final do trabalho será feita também uma autoavaliação por meio da participação de especialistas do domínio do negócio e também por consultores externos ao projeto.

* **Etapa de construção da lista de funcionalidades (Decomposição funcional):**

Os documentos e o modelo feitos na fase anterior possibilitam o início desta onde o entendimento do escopo é ampliado através da descoberta das funcionalidades de cada uma das áreas que são agrupadas em pacotes de trabalho e são implementados durante as iterações. A qualidade de realização dessa etapa define a precisão na qual o progresso do processo será rastreado e o grau de manutenibilidade do código.

* **Etapa de planejamento por funcionalidade (Planejamento incremental):**

É onde a equipe poderá planejar de que forma, como e em qual iteração serão implementados e entregues os requisitos. Nessa etapa se determina a sequência de desenvolvimento (levando em consideração as dependências entre as funcionalidades e classes envolvidas) e é também quando os programadores recebem os cronogramas e as atividades definidas.

**2.1.2. Etapas da fase de “Construção”:**

* **Etapa de design por funcionalidade (Design orientado a objetos):**

Já dentro de uma iteração de construção aqui são detalhados os requisitos para cada funcionalidade ser modelada de acordo com as necessidades arquitetais do produto sendo em prototipação de telas, esboçar, especificar e construir os esqueletos de cada classe e os diagramas de classes, de sequência ou qualquer outra ferramenta de modelagem que a equipe precise para ampliar o aprendizado da funcionalidade em questão.

* **Etapa de construir por funcionalidade (Programação e teste orientado a objetos):**

A segunda etapa da fase de Construção e última de todo o processo tem como objetivo a codificação, ou seja, o preenchimento de cada esqueleto de código, e é nessa etapa que o código referente a cada funcionalidade do produto é realmente desenvolvido pela equipe e também ocorrem os seus testes. Após ser devidamente testado o resultado será um incremento do produto integrado ao repositório principal do código.

**3. Como se dá a interação da equipe com o cliente?**

A interação de equipe e cliente se dá desde o início pelo processo Develop an Overall Model, que consiste em, o cliente apresentar uma espécie de “esboço do projeto”, em seguida, a equipe de desenvolvimento faz uma análise de tudo que foi dito pelo cliente e conversam com ele, sobre o que pode ser melhorado por exemplo, ou o que não foi muito bem especificado, coisas que foram esquecidas, e até mesmo sugestões ao cliente.

Feito isso, o cliente com o projeto final em mãos, apresenta-o aos desenvolvedores, dentre eles estão os especialistas da área, um modelador de componentes e um chefe design. Esse projeto é analisado e, caso o projeto esteja completamente claro, com todos os requisitos bem especificados dando a entender claramente o objetivo do projeto, é feito então uma espécie de estrutura inicial do sistema, onde é modelado o “esqueleto” do projeto.

Essa estrutura inicial então é mostrada ao cliente e, caso o mesmo receba o esperado, começam a ser adicionados detalhes ao sistema, as equipes então são divididas, tornando desnecessário a participação de pessoas de outras áreas em um setor específico do desenvolvimento do projeto. Por esse e outros motivos é essencial que todos os passos solicitados pelo cliente estejam perfeitamente claros e entendidos.

A partir deste momento, o cliente se torna mais afastado da equipe, porém, não totalmente, fazendo uma lista de coisas que são prioridades, sendo assim, definindo o que é mais importantes e primeiras a serem feitas.

Enquanto o desenvolvimento ocorre, são feitas reuniões semanalmente, onde se necessário, o cliente é chamado.

Para o processo de desenvolvimento ser finalizado, o programador chefe com a companhia do cliente, realiza os seguintes testes:

* Teste, implementação e revisão de cada método
* Testes individuais de cada método
* Teste do conjunto

Caso tudo ocorra conforme o previsto, o projeto é dado como pronto. É importante ressaltar que como o FDD foi criado para ser de mais rapidez, a relação entre o cliente e desenvolvedor não é muito próxima, para evitar atrasos de desenvolvimento.

**4. FDD e o uso de outras metodologias, ferramentas, etc.**

A metodologia FDD, obtém grande e satisfatório resultado quando se utiliza de metodologias e/ou ferramentas diversas, isso pelo fato do mesmo dar maior ênfase que outros métodos ágeis, o resultado disso, quando se é bem pensado, é o melhor aproveitamento de cada ferramenta e/ou metodologia que for designada a fazer parte do processo. Exemplos disso são:

*“ Um aspecto interessante do FDD em relação a outros métodos ágeis é que ele dá maior ênfase às diretrizes e técnicas de gerenciamento de projetos do que outros métodos ágeis disponíveis. (R.B. 18)”*

*“ É possível notar como o FDD pode atuar em conjunto com outras metodologias, principalmente com o Scrum, encaixando-se perfeitamente como metodologia de engenharia ágil de software com projeto ágil de software. ( R.B. 19)”*

* **FDD com Scrum:** a metodologia FDD pode atuar, obtendo grandes resultados, com outras metodologias, um exemplo disso é o Scrum. Quando os dois estão em conjunto, o Scrum assume o foco do gerenciamento do projeto e o FDD o processo de desenvolvimento.
* **FDD com TDD:** a FDD também faz testes de software e nessa etapa é possível utilizar a TDD, que é indicada para uma etapa de testes bem sucedidas.

**4.1. Ferramentas de aplicações**

Para utilizar esta metodologia em projeto, é necessário a utilização de ferramentas para que seja possível a organização de implementação do que deseja ser criado, essas ferramentas precisam ter requisitos específicos para a que a utilização do mesma seja facilitada.

Esta ferramenta tem o objetivo de organizar tudo, desde a discriminação de todas as “features” a ser implementadas como também das suas respectivas componentes, encontrando-se estas devidamente documentadas, dando uma visão geral da arquitetura do projeto, a partir disso, o uso desta ferramenta permitirá o controle de todos os setores/equipes, do tempo, além de evitar problemas como a repetição de alguma “features”, a partir dessa ferramenta será determinado como o projeto será construído, e assim o cliente terá uma ideia de como tudo será, assim podendo ter certeza de que era isso que ele queria, ou não.

*“ A utilização destas ferramentas permite maximizar as vantagens do FDD, essencialmente, no sentido de evitar custos extra, como também de tempo gasto em criação de componentes desnecessárias. (R.B. 17)”*

Sendo principalmente utilizadas nos primeiras três fases do desenvolvimento com FDD, possibilita o planejamento com mais eficácia. Algumas ferramentas que podem ser usadas, sendo algumas mais complicadas que outras, são Enterprise Architect, OptimalJ, MagicDraw, Poseidon, Simply Objects, Together, FDD Manager e Rose por exemplo.

**5. Utilização do FDD na prática**

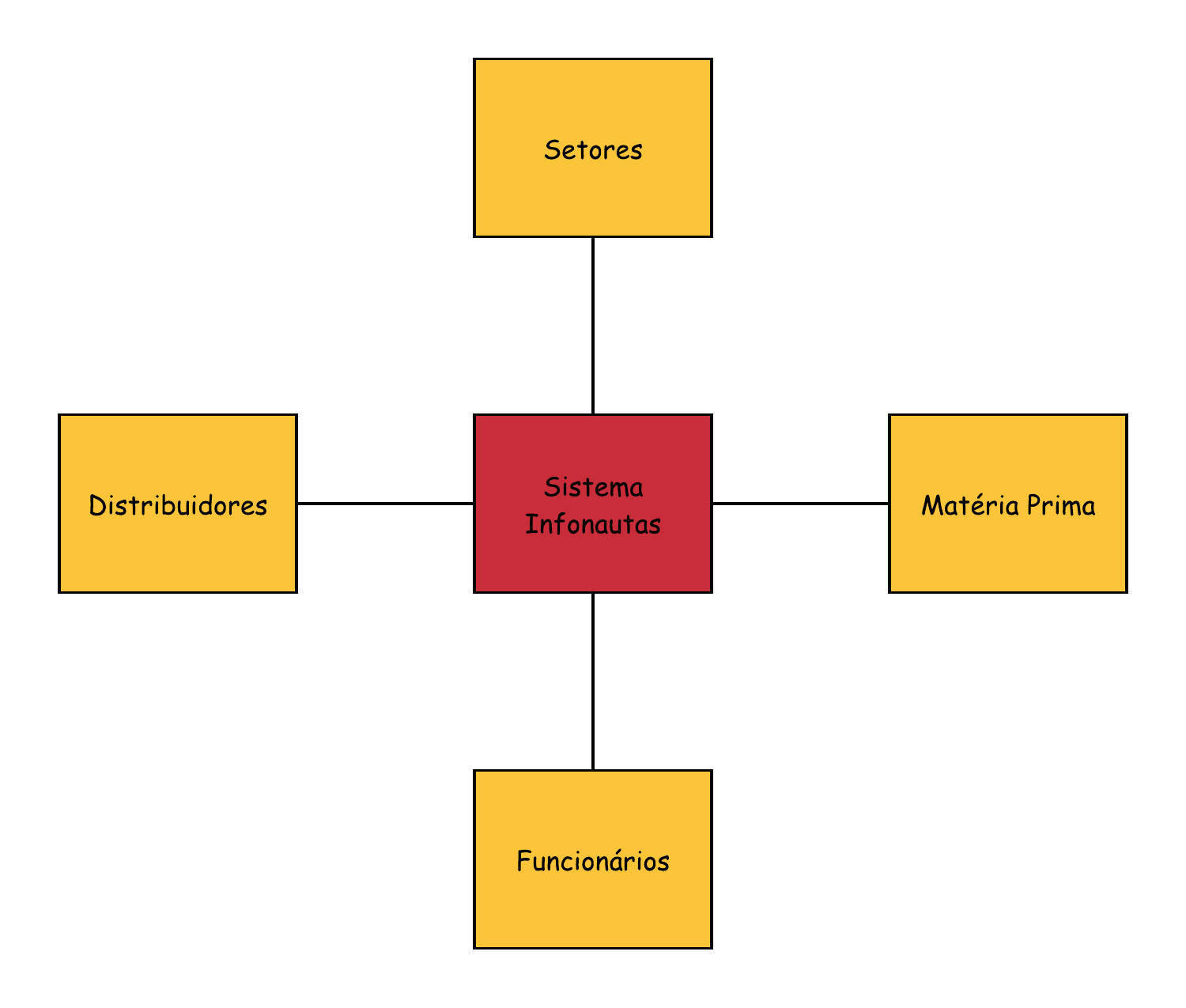
Suponhamos que uma franquia de pizzarias, denominada *“Infonautas”*, tem interesse em um sistema prático e quer que o mesmo seja entregue em um curto prazo de tempo para acelerar todo o seu processo produtivo. Atualmente, toda a sua base de dados está em planilhas e arquivos sem nenhuma ligação real uns com os outros.

Após uma conversa, entre o gerente de projetos e os stakeholders, foram levantados diversos requisitos, que basicamente são:

* Sistema de cadastro dos distribuidores, dos usuários individuais e dos setores da empresa;
* Um controle de estoque, para controle de quantidade de matéria prima e compra de novos recursos;
* Registro dos pedidos bem como de todos os envolvidos naquele pedido, desde a distribuição até a entrega dos lanches;
* Todo o sistema deve ser baseado em Linux e o banco de dados deve ser em MySQL, já que o novo CEO da empresa é um apoiador declarado do software livre.

**5.1 Fase 1: Modelagem de Objetos do Domínio**

A partir dos requisitos levantados, o gerente de projetos modela os objetos do domínio, para se ter uma visão mais ampla do sistema. A figura abaixo exemplifica um possível modelo para o nosso software da *“Infonautas”*:

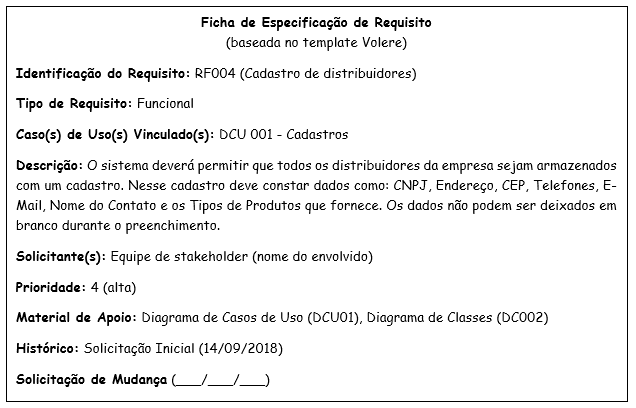


**Figura 1. Uma figura para a ênfase da Fase 1**

**5.2 Fase 2: Regras de Negócio**

Após a criação do modelo de objetos do domínio, o envolvido no projeto com mais experiência na área de atuação da *“Infonautas”* (o arquiteto-chefe) deve criar um relatório com as regras do negócio, que deve detalhar como tudo funciona na prática dentro da *“Infonautas”*.

O resultado disso tudo, como podemos ver na imagem abaixo, é um documento feito em reunião do arquiteto-chefe e dos stakeholders, que de preferência ser assinado por todos os envolvidos. Temos agora em mãos o documento de requisitos, que deve estar à disposição de consulta de todos os envolvidos no projeto. Ele pode ser feito utilizando um template como o template Volere, utilizado na figura. No geral, esse documento serve para melhor descrever cada exigência do projeto.

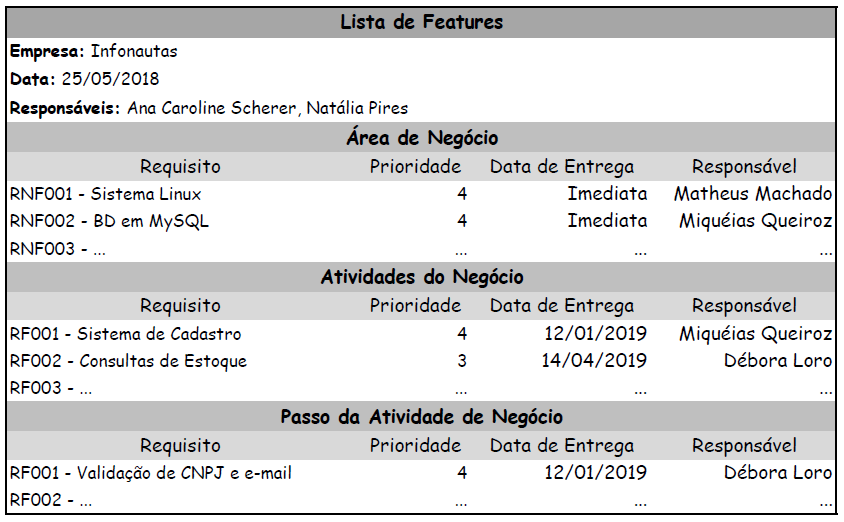


**Figura 2. Uma figura para a ênfase da Fase 2**

**5.3 Fase 3: Lista de Features**

Com o documento de requisitos pronto, a equipe de modelagem pode começar a produzir a lista de funcionalidades, que é uma listagem que contém informações detalhadas sobre tudo que deve ser desenvolvido e deve ser abstraída do documento de requisitos. Nela deve ser informado o tempo de desenvolvimento de cada funcionalidade e seus responsáveis, bem como novos recursos que podem agregar benefícios ao cliente, além de ser ordenada por prioridade. As funcionalidades devem ser categorizadas em três tipos: Áreas de Negócio, Atividade de Negócio e Passos de Atividade de Negócio.

Um exemplo de Lista de funcionalidades é a representada abaixo:

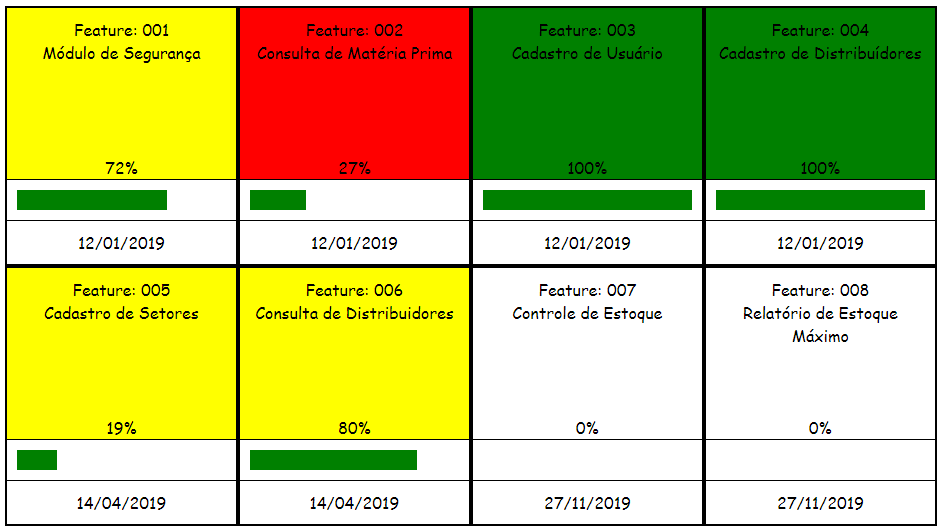


**Figura 3. Uma figura para a ênfase a Fase 3**

**5.4 Fase 4: Refinar a Lista de Features**

Com base na lista de funcionalidades, o programador-chefe cria o modelo de objetos, que divide os requisitos em pequenos pacotes a serem entregues à *“Infonautas”*. Cada pacote é como um conjunto de features que o cliente pode utilizar enquanto as outras ficam prontas. O programador-chefe ainda escolhe a linguagem de programação, as ferramentas e o tipo de banco de dados que será utilizado, embora no nosso exemplo, o próprio cliente tenha escolhido um deles. Cabe a ele ainda registrar os proprietários pelos códigos de cada build e conferi-los em suas funcionalidades por meio de testes. É aconselhável ainda gerar um documento que contenha os pacotes, os proprietários, datas, compilações e resultados dos testes, bem como um controle de versão.

Após todo esse processo, é criado um relatório para medir o progresso do desenvolvimento de cada funcionalidade, devendo ser constantemente atualizado. Podem ser utilizadas diversas cores e texturas para fácil entendimento. Esse tipo de gráfico permite ao gerente de projeto saber em quais features deve se ter mais atenção, além de dar um ótimo feedback aos stakeholders sobre o andamento do projeto.



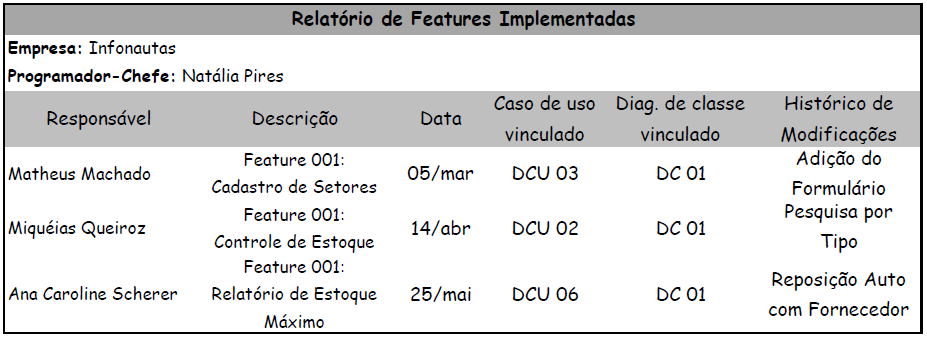
**Figura 4. Uma figura para a ênfase a Fase 4**

**5.5 Fase 5: Implementação**

Com a relação de features pronta, o programador-chefe pode agora lançar um registro em um relatório de features implementadas que deve conter informações sobre cada nova alteração feita em cada feature, como mostra a figura abaixo:

Nessa fase é crucial que cada feature finalizada seja incessantemente testada para garantir um controle de qualidade e se prevenir de eventuais erros que pode causar dor de cabeça mais à frente.

No nosso caso foi tudo apenas ilustrativo, mas é importante ressaltar que em um projeto de software real, as tabelas e relatórios podem e vão ser muitas vezes maiores, mas seguirão sempre um padrão semelhante ao apresentado.



**Figura 5. Uma figura para a ênfase a Fase 5**

**6. Referências Bibliográficas:**

1. SBROCCO, J. e De Macedo, Paulo (2012) “Metodologias Ágeis: Engenharia de Software Sob Medida”, Editora Érica Ltda., 1ª edição.
2. <https://visaoagil.wordpress.com/2010/02/25/fluxo-de-processos-da-fdd/>
3. http://periodicos.unesc.net/sulcomp/article/viewFile/245/250
4. https://www.devmedia.com.br/introducao-ao-fdd-feature-driven-development/27971
5. https://pt.slideshare.net/engenhariadesoftwareagil/fdd-5139226
6. https://blog.cedrotech.com/feature-driven-development-fdd/
7. <http://diego-pacheco.blogspot.com/2009/06/fdd-um-metodo-agil-e-eficiente.html>
8. https://sitecampus.com.br/feature-driven-development-artigo-1-de-2/
9. <https://agilementoring.wordpress.com/tag/fdd/>
10. <https://www.devmedia.com.br/modelos-de-processos-ageis-conceitos-e-principios/30059>
11. https://www.devmedia.com.br/introducao-ao-fdd-feature-driven-development/27971 http://www.leandromtr.com/metodologia-agil-fdd/
12. <http://homes.dcc.ufba.br/~mauricio052/Engenharia%20de%20Software%20I/FDD/FDDProcessos.pdf>
13. <https://htwojsystem.wordpress.com/2011/06/26/metodologia-agil-de-desenvolvimento-de-softwarefdd/>
14. <https://blog.cedrotech.com/feature-driven-development-fdd/>
15. <http://ariston-araujo.blogspot.com/2012/05/v-behaviorurldefaultvmlo.html>
16. <http://www.cpgls.pucgoias.edu.br/7mostra/Artigos/AGRARIAS%20EXATAS%20E%20DA%20TERRA/Levantamento%20de%20requisitos%20no%20desenvolvimento%20%C3%A1gil%20de%20software.pdf>
17. <https://paginas.fe.up.pt/~aaguiar/es/artigos%20finais/es_final_22.pdf>
18. <https://www.devmedia.com.br/modelos-de-processos-ageis-conceitos-e-principios/30059>
19. <https://www.devmedia.com.br/introducao-ao-fdd-feature-driven-development/27971>