****

**UNIVERSIDADE PAULISTA UNIP**

**ENGENHARIA DE SOFTWARE ÁGIL**

SOFIA DE CASTRO ONGARO - 2528018

**ENGENHARIA DE SOFTWARE ÁGIL**

SANTANA DE PARNAÍBA

2025

SOFIA DE CASTRO ONGAR0 - 2528018

**ENGENHARIA DE SOFTWARE ÁGIL**

Roteiro sobre o pensamento lógico computacional, e conceitos fundamentais da programação, como a lógica, algoritmos, pseudocódigos e fluxogramas. Para que o programador possua códigos organizados e bem estruturados. TENHO QUE ATUALIZAR AQUI AINDA

Orientador: **XXXXX**

SANTANA DE PARNAÍBA

**SUMÁRIO**

**1 RELATÓRIO 1**…..…………………………………….……………………….....……4

**2 ROTEIRO 2**………………………...…………………………..…………………….......6

**3 ROTEIRO 3**…………………………………………………..…………….........……….8

**4 ROTEIRO 4**……..…………………………………………..…………….........……….11

**RELATÓRIO 1**

O Software possui características marcadas pela sua evolução no ciclo de desenvolvimento, sendo ele um conjunto de instruções e dados que permitem com que os hardwares executem tarefas específicas. É imprescindível conhecer suas características básicas como ser evolutivo, intangível, sujeito a falhas e constantes atualizações até os dias atuais. A evolução do software no ciclo de desenvolvimento, vai desde o planejamento à manutenção, ele passa pelos processos de concepção, codificação, diversos testes, implementação e contínua atualização como dito anteriormente.

Já o Hardware, possui ciclos físicos, atualizações graduais, custo elevado e sofre desgaste por ser algo tangível. A principal diferença entre o Hardware e o Software é que o software é mais flexível, e consequentemente evolui mais rápido que o hardware.



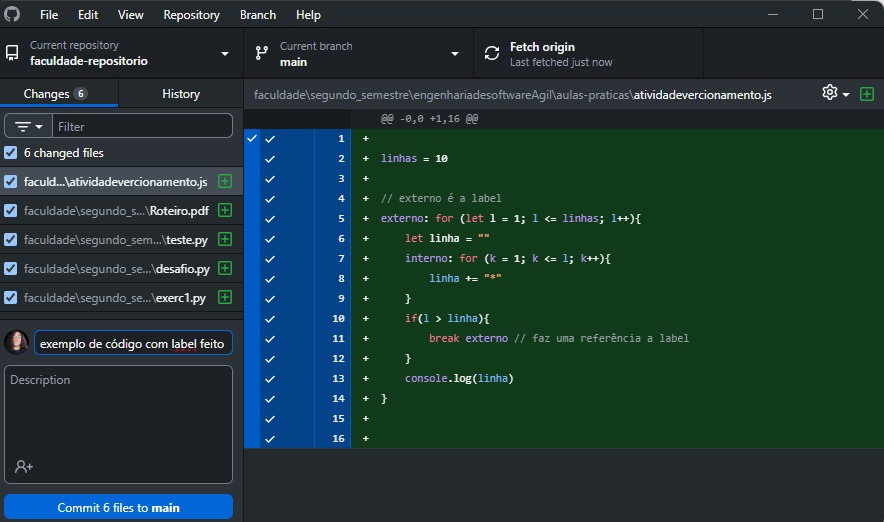
(Créditos: EDteam)

**Método de Controle de Versões com Labels:** Tem como serviço organizar e controlar o desenvolvimento do software, sendo capaz de: Rastrear mudanças, facilitar colaboração, reverter erros, comunicar atualizações. Labels, são como etiquetas que identificam o estado do repositório, marcando pontos importantes.

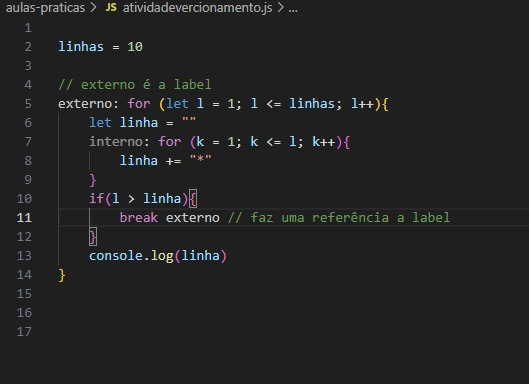
**Estrutura**: é necessário o uso de um formato padronizado: V[MAJOR] . [MINOR] . [PATCH]. O “Major” significa mudanças grandes ou que podem quebrar a contabilidade; já o “Minor” representa novas funcionalidades ao programa que sejam compatíveis com as versões antigas; “Patch” são todas as correções ou melhorias.

Por exemplo: V1.0.0 - Para versões iniciais, V1.1.0 - Para possíveis funcionalidades no projeto, V1.1.1 - Correção de Bugs, V2.0.0- Mudanças significativas.

**Exemplo Prático:**



Versionamento - (Créditos: GitHub)



Código (Créditos: GitHub)

**RELATÓRIO 2**

Os Requisitos do software, são os responsáveis por descreverem as funções e o que os sistemas devem fazer e a forma que devem agir. Existem vários tipos de Requisitos sendo os principais: Requisito de usuário (RU) composto por uma linguagem simples, descrevendo o que o usuário espera do programa (ex.: senha incorreta, tente novamente.); Requisitos Funcionais (RF) para ações específicas como, verificação em duas etapas via e-mail; Requisitos de Sistemas (RS), São especificações técnicas que engloba todos os requisitos, a fim de armazenar dados em um banco relacional; requisitos não funcionais (RNF), medem a qualidade do sistema, como o desempenho, otimização, segurança e usabilidade.

Seus processos e atividades são vastos, como o levantamento de requisitos, que são brainstormings (chuva de ideias), basicamente uma reunião que são levantadas as pautas do cliente, e as possíveis ações da empresa, definindo o escopo do projeto. Também atuam na modelagem ágil, que se baseia no uso de mapas mentais para organizar as ideias. Na documentação, criando relatórios, infográficos e tabelas. E por fim, na validação de Stakeholders, apresentado todos os requisitos buscando a aprovação antes do desenvolvimento do projeto.

**Método de Reuniões:** Para uma reunião bem-sucedida, e com respostas definidas do cliente. O trabalho se inicia antes mesmo da reunião, deve-se fazer um estudo do caso do cliente, averiguando sua situação no mercado, seu mídia kit, branding, e diversos fatores para enfim, realmente conhecer o cliente. Após isso, montar perguntas diretas e coesas ao cliente, com linguagem simples para que ele o entenda com clareza, anotando as respostas e há demonstrando entusiasmo com possíveis ideias.

**Diagramas de Casos de Uso (ERP):**



(Créditos: autoria própria)

Mapa mental – Requisitos do ERP



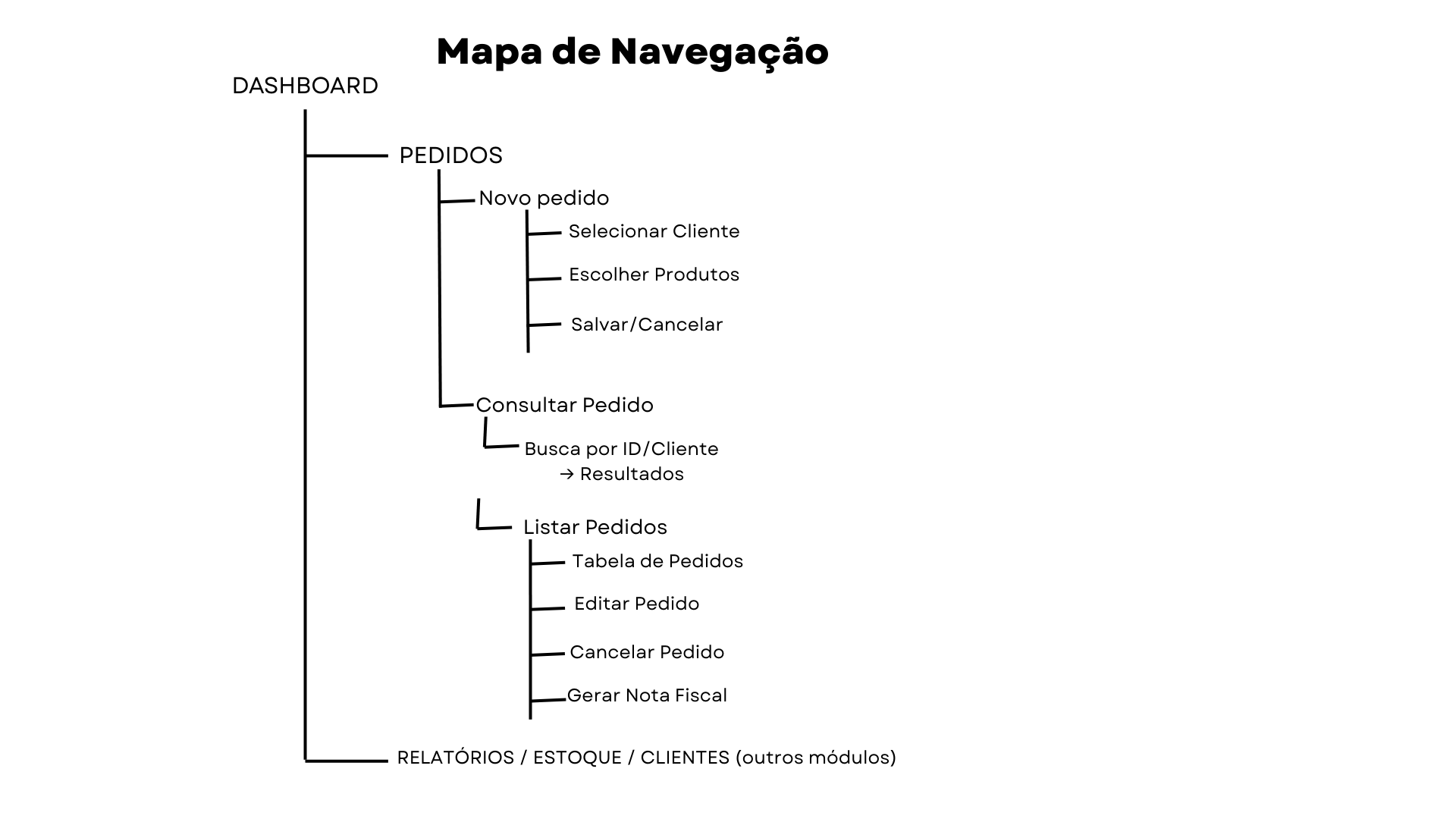
(Créditos: autoria própria)

**Tabela de Requisitos:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Código** | **Tipo** | **Descrição** | **Prioridade** | **Status** |
| RU01 | Usuário | usuário deve acessar o sistema com login e senha | Alta | Em análise |
| RU02 | Funcional | sistema deve registrar vendas e atualizar o estoque automaticamente | Alta | Validado |
| RU03 | Não Funcional | O sistema deve processar consultas em até 2 segundos | Média | Validado |
| RU04 | Sistema | O sistema deve armazenar dados em banco de dados relacional (MySQL) | Alta | Em Análise |

**RELATÓRIO 3**

O processo do software, é um conjunto de diversas atividades que são usadas para manter o software com sua funcionalidade, o processo possui uma estrutura simples objetivada em organizar como trabalho será feito pelos desenvolvedores, podendo planejar o desenvolvimento da reunião, testes e manutenções, a fim de garantir maior qualidade em seus serviços, previsibilidade e maior produtividade. Dentro dos processos, existem alguns modelos deles, sendo eles: modelo Cascata (Waterfall), que possui fases sequenciais (análise, design, implementação, testes, manutenção); modelo Incremental, que a cada versão é incrementada uma atualização; modelo Espiral, que tem foco em analisar os riscos de forma interativa; modelos Ágeis, que possui poucos ciclos (sprint) e adaptação rápida; modelo de prototipação, que valida rrequisitos antes do produto final gerando uma otimização.



(Créditos: autoria própria)

**Diagrama:**

[Início]

↓

(Planejar objetivos e requisitos)

↓

(Analisar riscos e alternativas)

↓

[Decisão: riscos aceitáveis?]

├── Não → (Revisar requisitos/estratégia) → volta para "Planejar"

└── Sim → segue

↓

(Desenvolver protótipo / incremento)

↓

(Testar e validar protótipo)

↓

(Avaliar com cliente)

↓

[Decisão: requisitos atendidos?]

├── Não → volta para "Planejar"

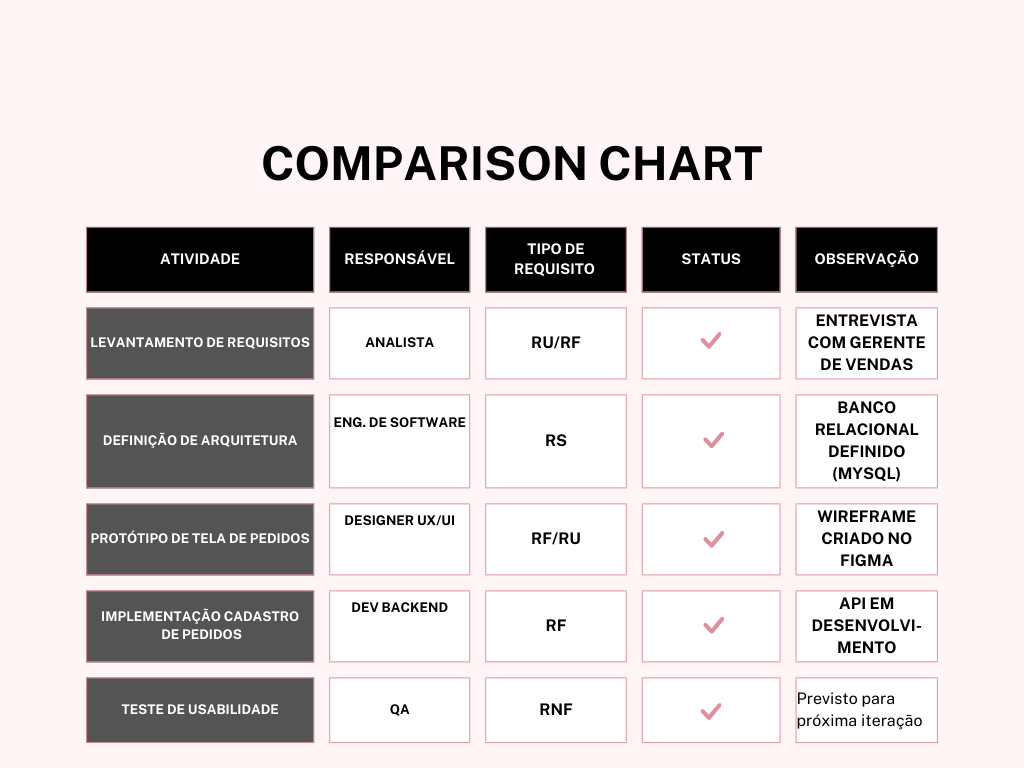
└── Sim → (Entregar versão final)

↓

[Fim]

**Exemplo prático relatório de equipe:**

Para o processo incremental da equipe é necessário que haja uma divisão das tarefas a partir de etapas, assim como os tipos de requisitos presentes nos textos supracitados, a tabela abaixo representa a distribuição de tarefas e suas obrigações.

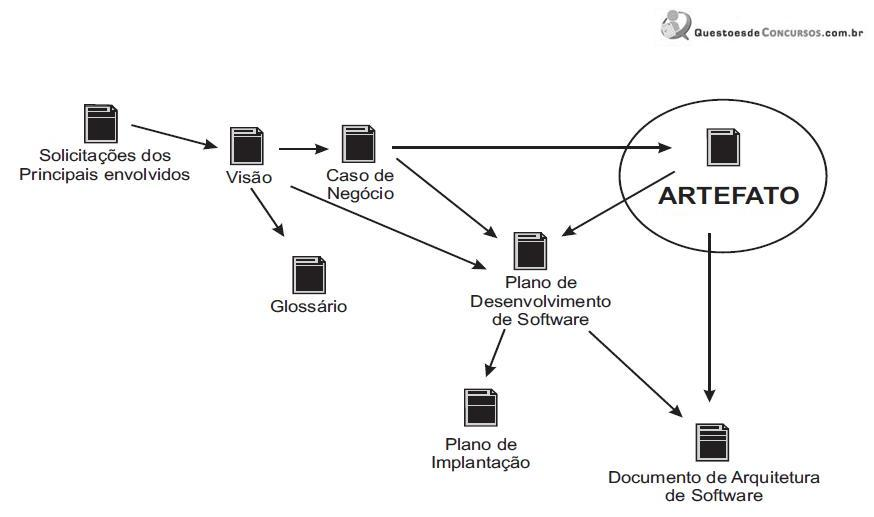


(Créditos: autoria própria)

**RELATÓRIO 4**

Os artefatos de sistema de software, são documentos produtos ou modelos, que são gerados a partir de um ciclo de desenvolvimento. Ou seja, são produtos intermediários ou finais gerados durante o desenvolvimento do software. Por exemplo, podem ser utilizados em levantamento de requisitos, colaborando com especificações e história de usuário; Manutenções com os logs e documentações atualizadas e entre outras utilidades.

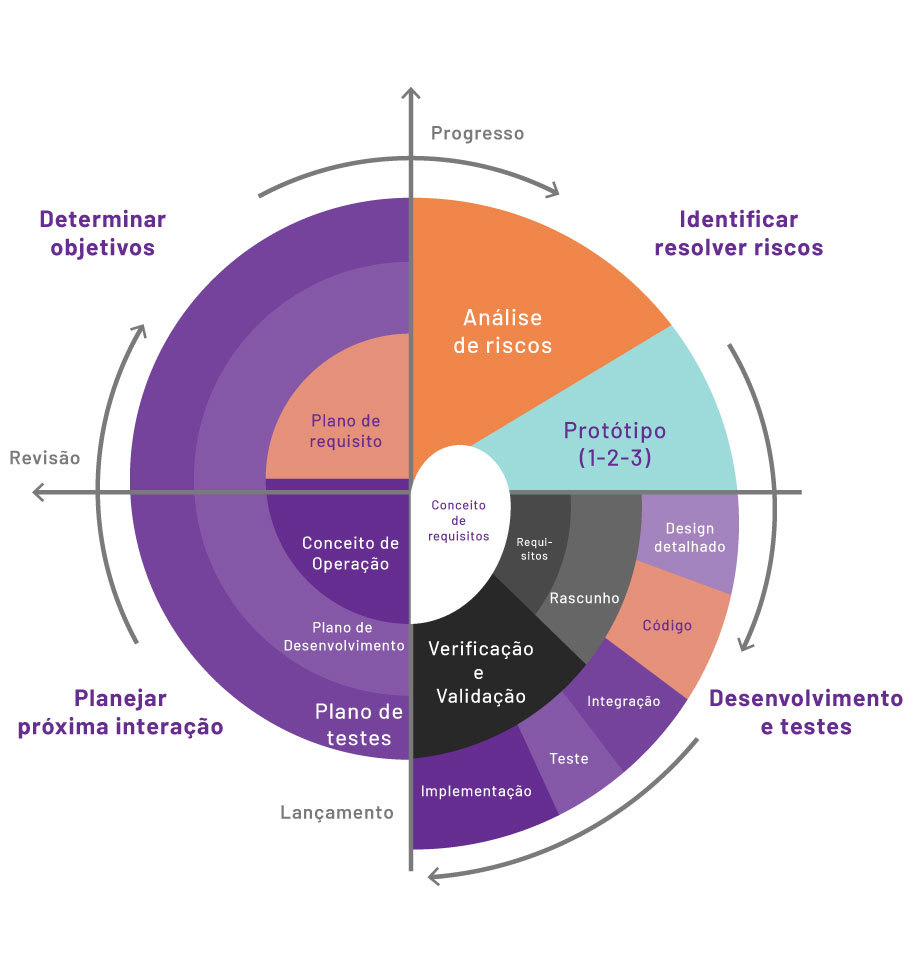
Durante o desenvolvimento de software, existem várias etapas, que geram diferentes tipos de materiais, como documentos que descrevem a visão do projeto, planos de desenvolvimento, arquiteturas e detalhes de implantação. A imagem abaixo, mostra como esses materiais são produzidos ao longo do processo, desde as primeiras solicitações até a documentação técnica final.



Fluxo de criação de artefatos (créditos: Qconcurso)

São existentes diferenças entre os elementos de qualidade, para do processo, métodos e ferramentas. Sendo elas a qualidade pois os atributos que fazem com que o software tenha um bom funcionamento, é a sua usabilidade, desempenho e segurança; o seu processo de trabalho; as ferramentas que são utilizadas (ex. Git Hub); e seus métodos como brainstoriming e prototipação.

**Estrutura do processo de Software:** A sua estrutura pode ser exemplificada como na imagem abaixo, no modelo Big Bang que consiste em um método simples, porém, eficaz. A primeira etapa é especificação, que define o objetivo do sistema a ser criado, a segunda etapa é o projeto de implementação que utiliza os artefatos citados anteriormente e transforma requisitos e códigos, logo após os testes, e manutenções caso necessário. Também existem as atividades de apoio sendo elas: A gestão do projeto, Gestão de configurações, Documentação, Garantia de qualidade e gerenciamento de risco. Com isso, a equipe terá bons resultados, e maior satisfaçao de seus clientes.



(créditos: UbiMinds)

*Joint Application Development,* mais conhecido com JAD, tem como objetivo principal levantar requisitos de maneira prática e rápida, reduzindo falhas na comunicação, por isso, é muito usada durante reuniões e entrevistas entre a equipe técnica de stakeholders.

**Matriz de Responsabilidade (RACI):** Muito utilizada como uma forma visual na gestão de projetos para classificar a responsabilidade de cada integrante.

**Exemplo:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Atividade** | Analista de Sistemas | Desenvolvedor | Usuário-chave | Facilitador JAD | Gerente de Projeto |
| Levantamento de requisitos | Responsável | Consultado | Responsável | Aprovado | Informado |
| Diagramas | Responsável | Consultado | Consultado | Aprovado | Informado |
| Requisitos Funcionais | Responsável | Consultado | Consultado | Aprovado | Informado |
| Prototipação | Consultado | Consultado | Consultado | Aprovado | Informado |
| Desenvolvimento da funcionalidade | Consultado | Responsável | Informado | Informado | Aprovado |
| Teste | Consultado | Responsável | Responsável | Aprovado | Informado |
| Validação | Consultado | Aprovado | Responsável | Aprovado | Aprovado |

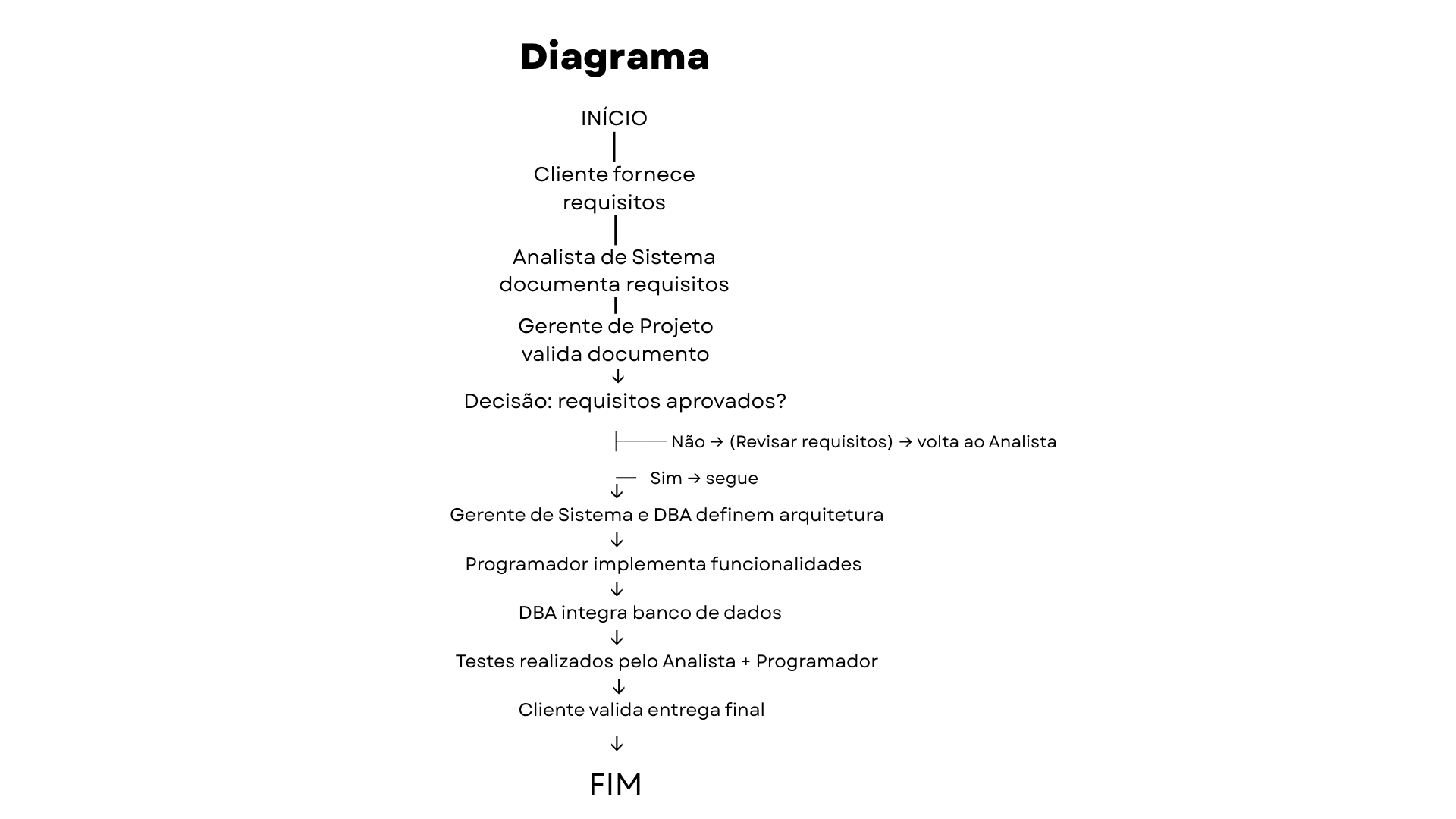
**Legenda:** Responsável, é quem foi determinado a executar a tarefa; Aprovado, é o dono final da tarefa, quem ficou responsável por aprovar; Consultado dá suporte a etapa final; Informado, sobre o andamento da atividade a ser feita.

**RELATÓRIO 5**

Como visto anteriormente no Relatório 4, JAD reúne stakeholder e a equipe técnica em reuniões colaborativas para definir responsabilidades. Como por exemplo a Matriz abaixo com a função de cada integrante da equipe, é importante salientar que para o exemplo estão presentes os cargos: Clientes (quem valida), gerente do projeto (planeja prazos), gerente de sistemas (integração do sotware), Analista de sistema (requisitos e especificações), programador (código fonte), e o administrador do banco de dados (projeta e otimiza o banco de dados).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Atividades** | **Cliente** | |  | | --- | | **Gerente de Projeto** |  |  | | --- | |  | | |  | | --- | | **Gerente de Sistema** | | **Analista de Sistema** | **Programador** | **DBA** |
| |  | | --- | | Documento de Requisitos do Sistema |  |  | | --- | |  | | |  | | --- | | Responsável |  |  | | --- | |  | | Consultado | Informado | Responsável | Consultado | Informado |
| |  | | --- | | Documento de Arquitetura de TI |  |  | | --- | |  | | Responsável | Informado | Consultado | Informado | Responsável | Informado |
| |  | | --- | | Modelagem de Casos de Uso |  |  | | --- | |  | | Aprovado | Consultado | Informado | Aprovado | Responsável | Consultado |
| |  | | --- | | Protótipo de Interface |  |  | | --- | |  | | Aprovado | Consultado | Consultado | Informado | Informado | Consultado |
| Implementação da Funcionalidade | Consultado | Informado | Informado | Consultado | Responsável | Consultado |
| |  | | --- | | Plano de Testes e Validação |  |  | | --- | |  | | Responsável | Informado | Consultado | Consultado | Aprovado | Aprovado |
| |  | | --- | | Implantação do Sistema |  |  | | --- | |  | | Responsável | Informado | Informado | Consultado | Aprovado | Aprovado |

|  |  |
| --- | --- |
| **Participante** | **Pontuação de Esforço (%)** |
| Cliente | 15% (valida requisitos e protótipos) |
| Gerente de Projeto | 20% (supervisão, aprovações) |
| Gerente de Sistema | 15% (garantia de integração e infraestrutura) |
| Analista de Sistema | 20% (especificações técnicas e casos de uso) |
| Programador | 20% (implementação e testes) |
| DBA | 10% (modelagem e manutenção do banco) |



(créditos: Autoria própria)

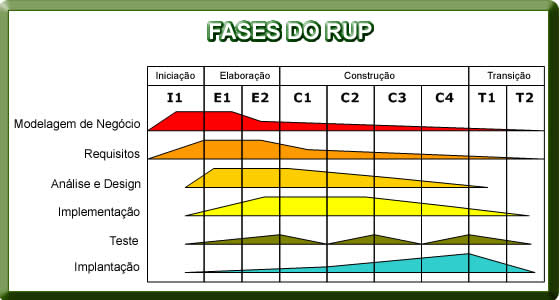
**RELATÓRIO 6**

O processo unificado (UP) é um método incremental para o desenvolvimento de um software, ele divide os projetos em ciclos menores, entregando pequenas versões até a versão final, consequentemente, reduz os riscos. O UP evoluiu para o RPU, que é o Processo Unificado Racional, que é uma implementação comercial.

O PERT (Program Evaluation and Review Technique), é uma ferramenta de gestão de negócios que utiliza diagramas de rede para visualizar melhor o projeto a ser feito pela equipe, comumente são utilizados símbolos de ciclos ou até mesmo nós para representação de evento a acontecer, e setas para atividades. O PERT, também utiliza tempos estimados, que nada mais são que, otimista (O), provável (M) e pessimista (P). **Fórmula do tempo esperado (TE) q**ue ajuda a identificar a duração de cada uma das atividades:

**Rede de cartões:** As redes PERT e CPM, são metodologias de gestão de projetos, é uma representação em maneira de grafos (nós e setas como dito anteriormente), elas exemplificam as dependências entre as atividades e o caminho crítico, são utilizadas principalmente quando a uma preocupação maior com tempo e prazo de entrega de um determinado projeto.

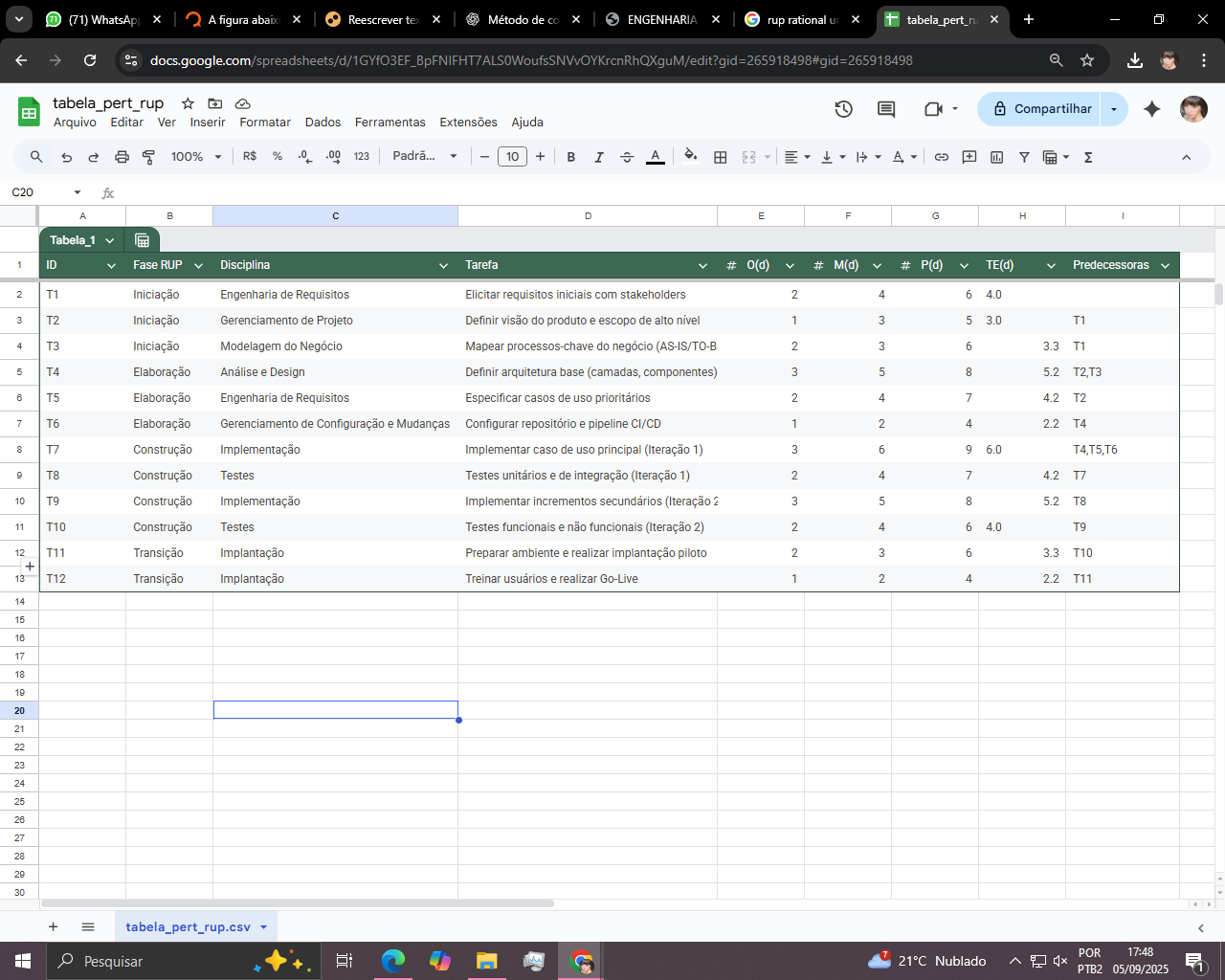
O RUP, é basicamente a implementação do processo unificado do IBM Rational (conjunto de ferramentas e soluções de ciclo de vida), utilizando de uma estrutura em diciplinas do que fazer e fases do período de que deve ser feito. O RUP, é composto por quatro fases, sendo elas: 1- Iniciação, que é a visão de como será o projeto; 2- Elaboração, em que são feitos protótipos do que há de ser feito; 3- construção, desenvolvimento do projeto; 4- Transição, que são validações e testes finais do projeto.



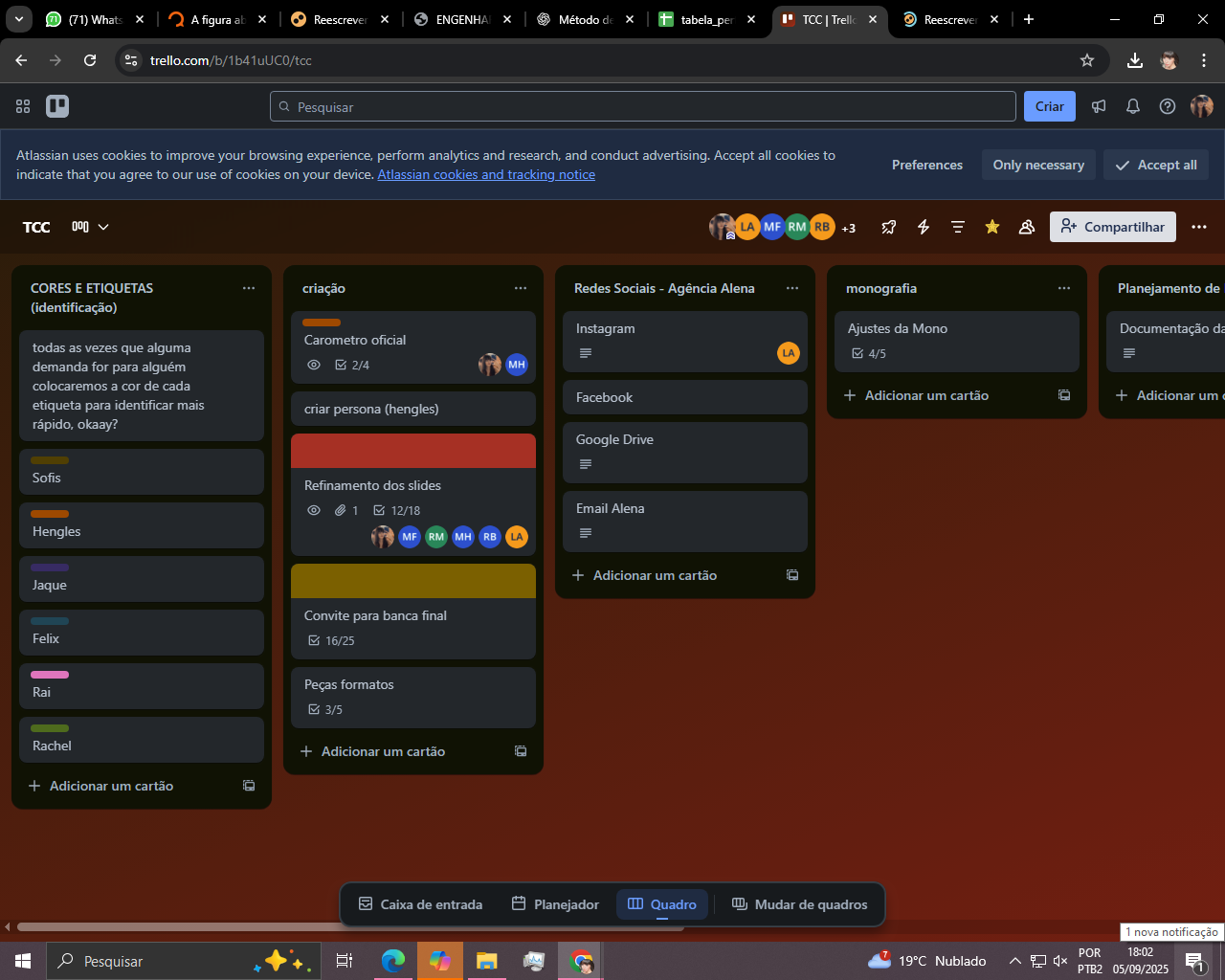
(créditos: InfoEscola)

Como visto na imagem acima, RUP engloba diversas matérias como: Modelagem de Negócios, Requisitos, Análise e Design, Implementação, teste e implantação. Também pode abordar matérias como Gestão de Configuração e Mudança e Gerenciamento de Projetos. Cada atividade no RUP leva um tempo, e ele pode ser estimado a partir do uso do PERT, mas antes disso deve-se listar quais são estas atividades e quais são dependentes uma da outra, para uma possível otimização do tempo.

**Exemplo prático de Matriz de tarefas:**

(créditos: autoria própria- ferramenta: Google Planilhas)

Exemplo de cronograma associando as atividades do RUP com o método PERT, feito pelo trello. As atividades foram divididas em colunas que representam as diferentes etapas do processo, alinhadas às fases do RUP. Cada tarefa foi atribuída a um cartão com um prazo definido, estimado usando a técnica PERT.



(créditos: Autoria própria)