Compass UOL

Plano de Teste

SERVEREST

Relatório de planejamento de teste, apresentado ao Evangelista Matheus Domingos Locatelli, como parte das exigências do challenge.

Recife
2022

Sumário

Plano de testes da API SERVEREST

1. INTRODUCÃO

1.1. Visão geral

O objetivo deste documento é ser um norte para os testes que serão feitos na API pública SERVEREST. Os testes serão executados a fim de garantir a qualidade do objeto em questão partindo da hipótese inicial do seu pleno funcionamento. Ou seja, iniciaremos nossos testes com base na premissa de total funcionalidade da API visando:

- Compatibilidade da API;
- Ausência de restrições em suas funções;
- Pleno funcionamento;

Este programa vai resultar validações significativas nos atuais processos da API e sua incorporação. As funcionalidades serão entregues em sprints.

A sprint1 vai incorporar as seguintes características:

1. Substituição de funções não operacionais;
2. Otimização da API;
3. Sistema operando com suas funcionalidades básicas;
4. Características novas/revisadas pela equipe.

[Inclusões detalhadas estão listadas mais adiante neste documento]

1.2. Propósito deste documento

Este documento deve servir como o **Rascunho** da Abordagem (estratégia) de Teste para o **Projeto de Desenvolvimento da API SERVEREST**.

A preparação para este teste consiste de três estágios principais:

- A **Abordagem (estratégia) de Teste** define o escopo do teste do sistema, a estratégia geral a ser adotada, as atividades a serem completadas, os recursos gerais necessários e os métodos e processos a serem usados para testar a versão. Ela também detalha as atividades, dependências e esforços requeridos para conduzir o teste de sistema.
- O **Planejamento de Teste** detalha as atividades, dependências e esforços requeridos para conduzir o teste de sistema.
- As **Condições/Casos de Teste** documentam os testes a serem aplicados, os dados a serem processados, a cobertura automatizada de teste e os resultados esperado.

1.3. Revisão formal

Há vários pontos formais de revisão antes e durante o teste da API. Este é um elemento vital para alcançar um produto de qualidade.

- **Pontos formais de revisão ocorrem ao final da elaboração de:**

1. Documentação de Desenho (mapa-mental)

2. Abordagem de teste
3. Planos de Teste Unitário (Desenvolvimento)
4. Condições e Resultados do Teste Unitário
5. Condições do Teste de Sistema
6. Progresso do teste de sistema
7. Revisão após o teste de sistema

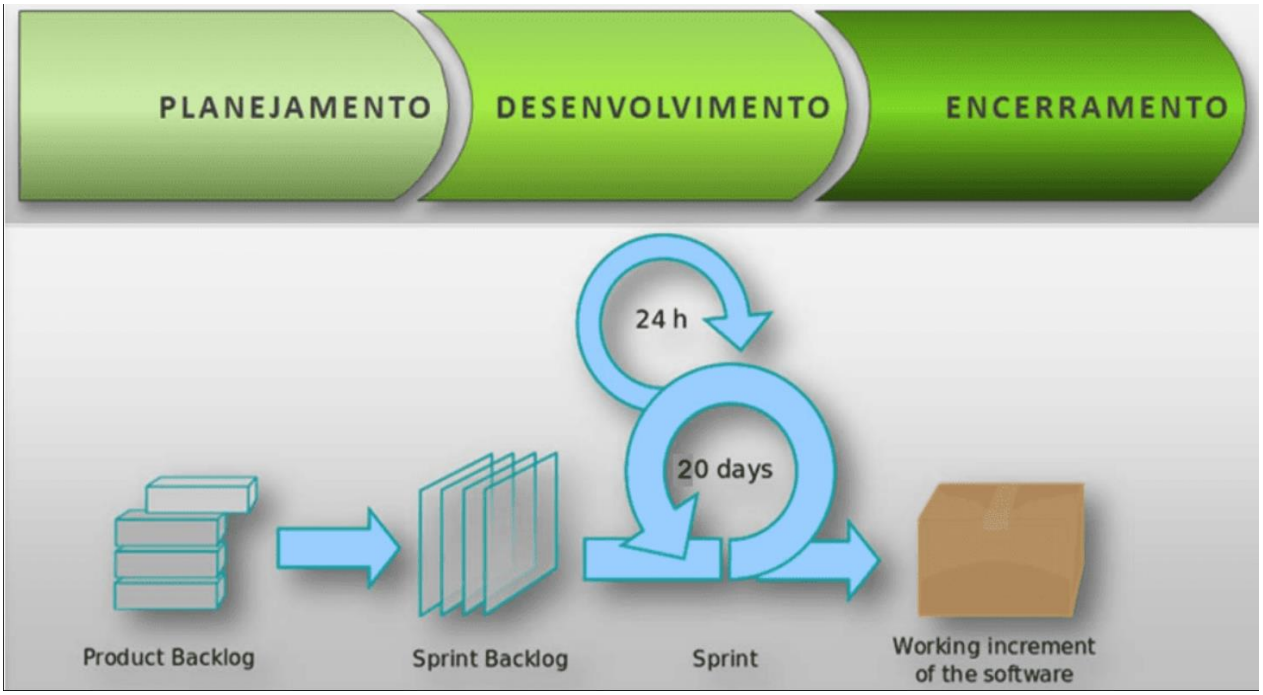
1.4. Objetivos do Teste de Sistema

Em um nível elevado, o teste de sistema tem o objetivo de provar que:

- A funcionalidade, entregue pela equipe de desenvolvimento, está de acordo com as especificaçõesdo negócio e da documentação de requisitos.
- O software é de alta qualidade; a API vai gerenciar às funções desejadas e alcançar os padrões requisitados pela companhia.
- A API entregue faz a interface correta com os sistemas existentes.

[Objetivos detalhados estão listados mais adiante neste documento]

Envolvimento da Garantia de Qualidade de Software:



O modelo Scrum acima mostra o processo de teste ideal, onde a preparação de teste começa assim que a definição de requisitos é produzida. O planejamento de teste de sistema começou em um estágio inicial, e por esta razão o teste de sistema vai se beneficiar de iniciativas de qualidade através do ciclo de vida do projeto.

As responsabilidades entre o departamento de Qualidade de Software (QA) e o departamentodo de Desenvolvimento são as seguintes:

- Testes Unitários são de responsabilidade dos desenvolvedores;
- Testes de sistemas são de responsabilidade da Qualidade de Software;
- Testes de aceitação de usuário são de responsabilidade de todas as equipes, mas se possível

ser validado através da equipe de representação do usuário (PO);
- Testes de conformidade tecnológica são de responsabilidade de toda a equipe.

2. ESCOPO E OBJETIVOS

2.1. Escopo da abordagem de teste – Funções do sistema

INCLUSÕES (ESCOPO)

O conteúdo desta versão é:

Funções a serem entregues na sprint 1(v2.25.4)

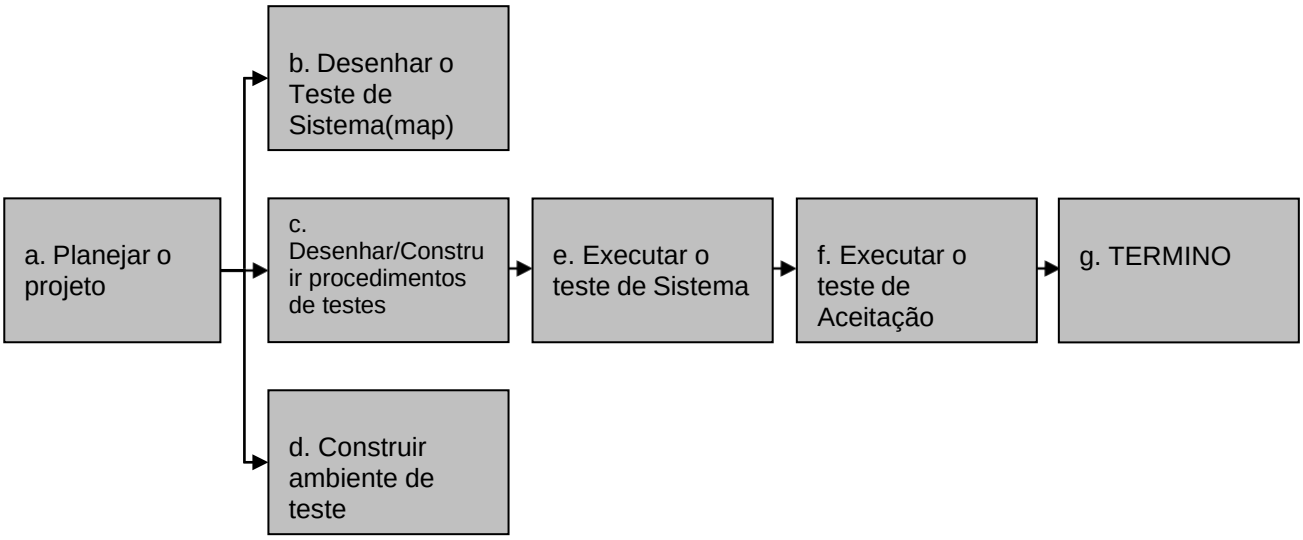
- Validação dos endpoints presentes no swagger (v2.25.3):
  - /login
  - /usuarios
  - /usuarios/{_id}
  - /produtos
  - /produtos/{_id}
  - /carrinhos
  - /carrinhos/{_id}
  - /carrinhos/concluir-compra
  - /carrinhos/cancelar-compra
- Validações de possíveis falhas, assim como suas correções.

Observação:
**Será feita à adição de informações de status na documentação do swagger.*

Funções a serem entregues na sprint 2 (v2.25.5)

- Reparo e manutenção de possíveis bugs reportados por clientes;

2.2. Processo de teste



[O diagrama acima explica a abordagem do processo de teste que será seguida]

a. **ORGANIZAR O PROJETO -** envolve a criação de um plano de teste de sistema, abordagem de cronograma & teste, e requisitar/delegar recursos.
b. **DESENHAR/CONSTRUIR TESTE DE SISTEMA -** envolve identificar ciclos de teste, casos de teste, critérios de entrada e de saída, resultados esperados, analise da

documentação(swagger), etc. Em geral, as condições de teste e os resultados esperados serão identificados pela equipe de teste em conjunto com o analista de negócio do projeto ou com o especialista do negócio. A equipe de teste vai então identificar os casos de teste e os dados requeridos. As condições de teste são derivadas do desenho do negócio e dos documentos de requisitos de transação.

c. **DESENHAR/CONSTRUIR PROCEDIMENTOS DE TESTE -** inclui preparar procedimentos como sistemas de gerenciamento de erro e relatório de status, e preparar as tabelas de dados para a ferramenta automatizada de teste.
d. **CONSTRUIR O AMBIENTE DE TESTE -** inclui requisitar/construir hardware e software e preparar dados, assim como moldar ambientes virtuais.
e. **EXECUTAR TESTE DE INTEGRAÇÃO -** veja seção 5 – Ciclos e fases de teste.
f. **EXECUTAR TESTE DE ACEITAÇÃO -** veja seção 5 – Ciclos e fases de teste.
g. **TÉRMINO -** o término acontece quando todos os critérios pré-definidos foram alcançados.

2.1. Escopo de teste

Abaixo estão os principais tipos de testes que serão feitos para esta versão. Todos os planos e condições de testes de sistema serão desenvolvidos a partir de especificações funcionais e do documento de requisitos.

2.2. Teste Funcional

O objetivo deste teste é garantir que cada elemento da API atenda os requisitos funcionais do negócio conforme descritos:

- No documento de requisitos
- Na especificação de desenho do negócio
- Nos padrões de “desenvolvimento atuais”
- Em outros documentos funcionais produzidos durante o andamento do projeto, como resoluções de questões, requisições de mudança e feedbacks.

Nesta etapa inclui **testes funcionais específicos** – estes são testes de nível baixo que têm o objetivo de testar os processos individuais e fluxos de dados.

2.3. Teste de Integração

Esse teste é uma atividade sistemática que tem como objetivo verificar a construção da estrutura do software que está sendo desenvolvida e a sua comunicação entre módulos.

2.4. Teste de Aceitação

Este teste, que é planejado e executado pelo(s) representante(s) do negócio (cliente), assegura que o sistema opera da maneira esperada, e que o material de suporte (como procedimentos e formulários) está correto e serve ao seu propósito. É um teste de alto nível, que assegura que não há problemas na funcionalidade.

2.5. Teste de Performance

Este teste assegura que o sistema fornece tempo de resposta dentro do padrão aceitável.

2.6. Teste de Regressão

Um teste de regressão deve ser feito depois da liberação de cada fase para assegurar que:

- Viabilidade da aplicação mais recente da API;
- Garante que não surgiram novos defeitos em componentes já analisados.

O teste de regressão será automatizado com o uso da ferramenta automatizada de teste. Lembrando que é muito importante ser feito nos casos que houve mais incidências de erros.

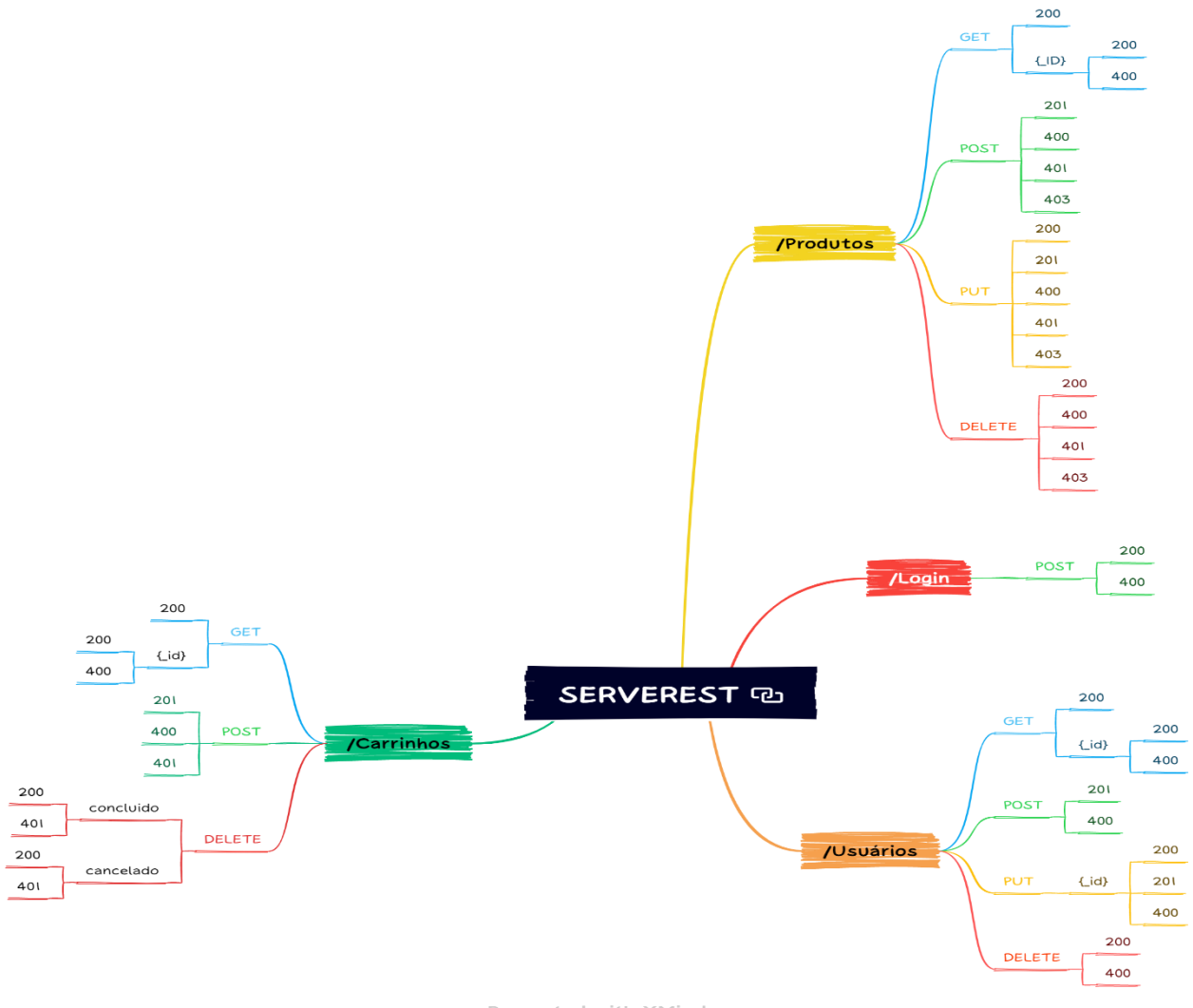
2.7. Teste de carga

O teste de carga é uma tentativa customizada de sobrecarregar o sistema, ele também já foi chamado de teste de quebra (nomenclatura menos utilizada recentemente). Ele engloba um teste muito importante que é o teste de **multi-usabilidade** que deve provar que é possível que um número aceitável de usuários trabalhe no sistema ao mesmo tempo.

Observação:

Todos os testes serão realizados em ambiente local.

3. MAPA MENTAL



4. VALIDAÇÕES A SEREM TESTADAS

Observações:
**Importante todas as validações serão testadas por usuário logado e por usuário não logado, assim como por usuários administradores;*
*****Candidatos a automação estão em negrito abaixo.***

Todos os testes serão executados em ambiente local.

4.1. Login

- **Realizar login como usuário admin**
- **Realizar login como usuário comum**
- Realizar login com email e senha vazia
- Realizar login com email em formato invalido
- Realizar login com login e/ou senha errada
- Realizar login com request diferente do padrao
- Realizar login sem os campos obrigatórios

4.2. Usuários

- **Listar usuários**
- **Cadastrar usuário – Admin**
- **Cadastrar usuário – Comum**
- **Buscar usuário por ID**
- **Excluir usuário (com efeito)**
- Excluir usuário (sem efeito)
- **Editar usuário (alterar)**
- **Editar usuário (cadastro)**
- Cadastrar usuário - Usuário já existente
- Cadastrar usuário - sem email
- Cadastrar usuário - email invalido
- Cadastrar usuário - sem body
- Buscar usuário por ID - Com ID errado
- Editar usuário - email já cadastrado
- Excluir usuário - com carrinho cadastrado

4.3. Produtos

- **Listar produtos cadastrados**
- **Cadastrar produto**
- **Buscar produto por ID**
- **Excluir produto - Excluído Sucesso**
- **Excluir produto - Nenhum Excluído**
- **Editar produto**
- Editar produto - Cadastrar produto
- Cadastrar produto - Já cadastrado
- Cadastrar produto - Sem autorização para cadastrar
- Cadastrar produto - Usuário cadastrando produto
- Cadastrar produto - Quantidade negativa
- Cadastrar produto - preço sendo menor que zero
- Buscar produto - Não encontrado
- Excluir produto - No carrinho
- Excluir produto - Sem Token autorizando
- Excluir produto - Exclusivo para administradores
- Editar produto - Produto com mesmo nome
- Editar produto - Sem token
- Editar produto - Rota de Administradores

4.4. Carrinhos

- **Listar carrinhos cadastrados**
- **Cadastrar carrinho**
- **Buscar carrinho por ID**
- **Excluir carrinho (concluir-compra)**
- Excluir carrinho (concluir-compra) - Carrinho não encontrado
- **Excluir carrinho e retornar produtos para estoque**
- Excluir carrinho e retornar produtos para estoque - Não foi encontrado carrinho
- Cadastrar carrinho - Produtos duplicados
- Cadastrar carrinho - Mais de 1 carrinho
- Cadastrar carrinho - Erro Produto não encontrado
- Cadastrar carrinho - Erro Produto zerado
- Cadastrar carrinho - sem token
- Cadastrar carrinho - quantidade de produtos negativa
- Buscar carrinho por ID - Não encontrado
- Excluir carrinho (concluir-compra) - Sem token
- Excluir carrinho e retornar produtos para estoque - Sem Token

4.5. Fluxos

- **Fluxo de cadastro produto;**
  - Cadastrar usuário – admin;
  - Realizar login;
  - Cadastrar produto.
- **Fluxo de exclusão – Usuário comum desistindo da compra e tentado excluir sua própria conta, mas com carrinho ativo;**
  - Cadastrar usuário;
  - Realizar login;
  - Buscar produto;
  - Cadastrar carrinho;
  - Excluir usuário.
- **Fluxo de compra – Simulação de usuário comprando;**
  - Cadastrar usuário;
  - Realizar login;
  - Buscar produto;
  - Cadastrar carrinho;
  - Excluir carrinho (Compra concluída).
- **Fluxo de compra – Simulação de usuário desistindo da compra.**
  - Cadastrar usuário;
  - Realizar login;
  - Buscar produto;
  - Cadastrar carrinho;
  - Excluir carrinho (Cancelando compra).

Será entregue na primeira sprint a validação de todos os endpoints da suíte de testes, priorizados de acordo com o fluxo de prioridade. Na sprint seguinte será tratado os bugs relatados por usuários para conclusão do plano. E assim ao final teremos todos os 9 endpoints da suíte de testes já validados no Postman e devidamente comprovados no report Newman.

5. FASES E CICLOS DE TESTE

Haverá dois estágios principais de teste para novos aplicativos durante o teste de sistema:

- Teste de sistema
- Teste de aceitação

5.1. Ciclos de teste de sistema

O principal impulso da abordagem (estratégia) é testar intensivamente, assim levantando uma grande parte dos erros/bugs neste período. Com a maioria dos erros consertados, ações- padrão ou ações frequentemente usadas serão testadas para provar elementos individuais e o processamento total do sistema na liberação v2.25.5.

Quando todos os erros (que potencialmente impactam o processamento geral) forem consertados, um conjunto adicional de casos de teste são processados na liberação v.2.25.6 para assegurar que o sistema funciona de maneira integrada. É esperado que a liberação v2.25.6 seja a prova final do sistema enquanto API única.

Casos de teste por versão liberada:

Teste por fase	
	Aceitação
V2.25.4	Funcional
	Regressão (em relação a v2.25.3)
	Aceitação 2
	Funcional 2
V2.25.5	Performance
	Teste de carga
	Regressão (em relação a v2.25.4)
	Regressão 2 (em relação a v2.25.5)
	Integração 1
	Técnico 1
V2.25.6	Regressão 1 (em relação a v2.25.3)
	Regressão 2 (em relação a v2.25.4)
	Regressão 3 (em relação a v2.25.5)
	Teste de instalação
Contingência	*Apenas teste de conserto por bug*

Teste automatizado: Ferramentas de teste automatizado serão usadas em ambiente de teste para teste funcional e teste de regressão. O foco principal do teste automatizado é o teste de regressão da funcionalidade previamente entregue – isto é, quando a versão 2.25.4 do software é entregue, a maioria dos testes de regressão da funcionalidade entregue na versão 2.25.3 será automática. É esperado que o benefício total do teste automatizado ocorra somente quando os testes tenham sido executados três ou mais vezes.

5.2. Entrega da software

Durante o teste de sistema, a liberação de novas versões do software deve ser coordenada entre o líder da equipe de desenvolvimento e o analista de teste de sistema. Entretanto, a menos que sirvam para corrigir um erro muito grave, novas versões só devem ser liberadas quando objetivos concordados sejam alcançados (por exemplo, a próxima versão contém correções para um número X ou mais de bugs).

Cronograma de liberação:

Funcionalidade a ser entregue*	*V2.25.4 17 de junho*	*V2.25.5 1 de julho*	*V2.25.6 15 de julho*	*V2.25.7 29 de julho*
1. Cadastrar produto				
2. Listar produtos cadastrados		*Nenhuma*	*Nenhuma*	*Apenas*
3. Buscar produto por ID				
4. Excluir produto		*funcionalidade*	*funcionalidade*	*versão de*
5. Editar produto				
6. Cadastrar usuário		*nova a ser*	*nova a ser*	*contingência*
7. Listar usuários				
8. Buscar usuário por ID		*entregue*	*entregue*	*para*
9. Excluir usuário				
10. Editar usuário		*nesta*	*nesta*	*conserto*
11. Realizar login				
12. Cadastrar carrinho		*versão*	*versão*	*de bug*
13. Listar carrinhos cadastrados				
14. Buscar carrinho por ID				
15. Excluir carrinho				
16. Outros**				

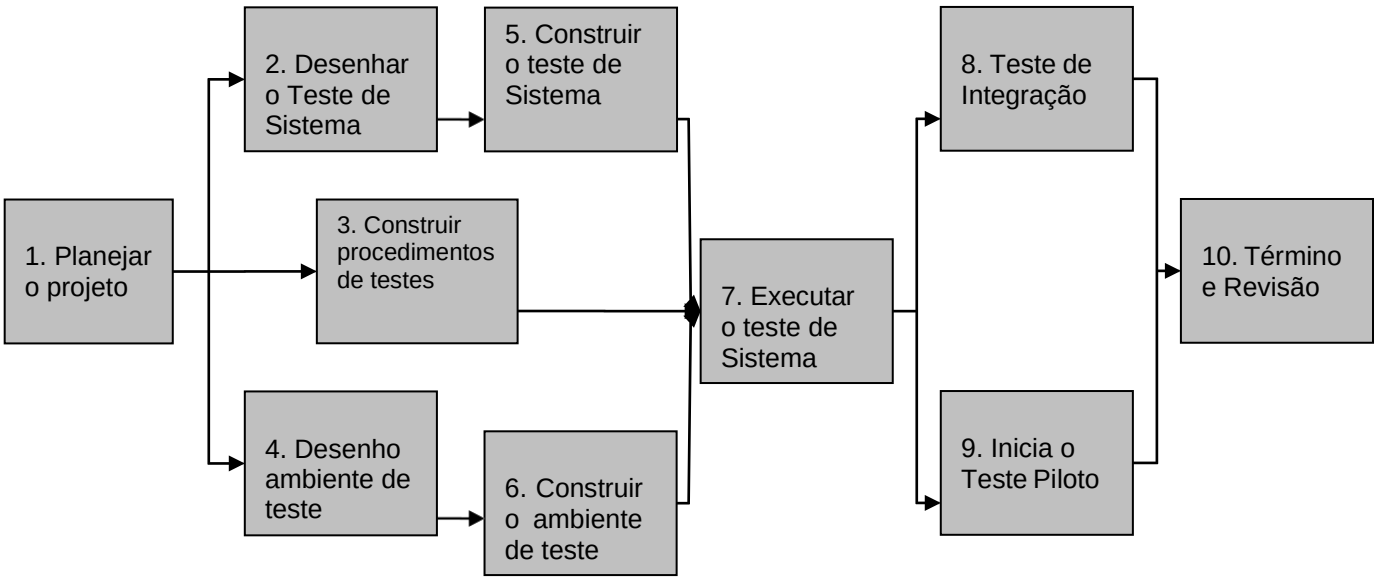
** (por especificação funcional, por prioridade)*
*** (função(s) adicional(s) solicitada(s) pelo cliente, ou pela equipe a fim de melhor no projeto)*

Observações:
Espera-se que 100% das funcionalidades tenham sido completamente testadas e todas as funcionalidades devem estar presentes na liberação de sprint 1.

5.3. Revisão formal

Há vários pontos formais de revisão antes e depois do teste de sistema, incluindo a revisão deste documento. Este é um elemento vital para obter um produto de qualidade.

5.3.1. Pontos de revisão formal



1. Documentação de desenho – especificação de requisitos e especificações funcionais
2. Plano de teste de sistema
3. Planos de testes unitários & condições de teste
4. Resultados de testes unitários
5. Condições de teste de sistema
6. Progressos/resultados de teste de sistema
7. Revisão após teste de sistema
8. Resultados de teste de integração
9. Revisão de implantação-piloto
10. Revisão do projeto

O diagrama acima mostra a abordagem (estratégia) de teste. As caixas de 1 a 6 mostram os principais estágios de revisão antes da execução do teste. As caixas de 7 a 10 mostram as fases planejadas para antes e depois da execução do teste.

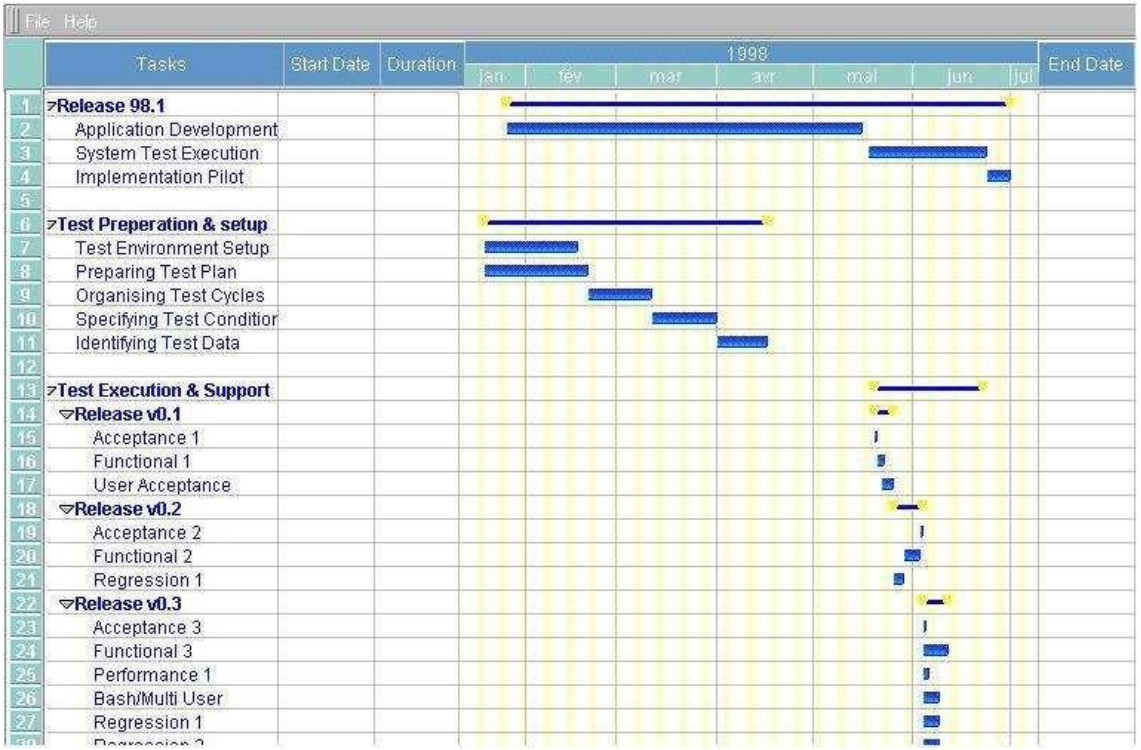
Esse diagrama se concentra no aspecto de teste do papel da Garantia de Qualidade do Software, mas há também um papel em andamento, para assegurar a qualidade dos entregáveis principais através do ciclode vida do projeto. O papel da Garantia de Qualidade do Software será assegurar que todas as inspeções de qualidade ocorram para todos os entregáveis concordados, e que ações de follow-up e iniciativas sejam buscadas.

5.3.2. Monitoramento de progressos/resultados

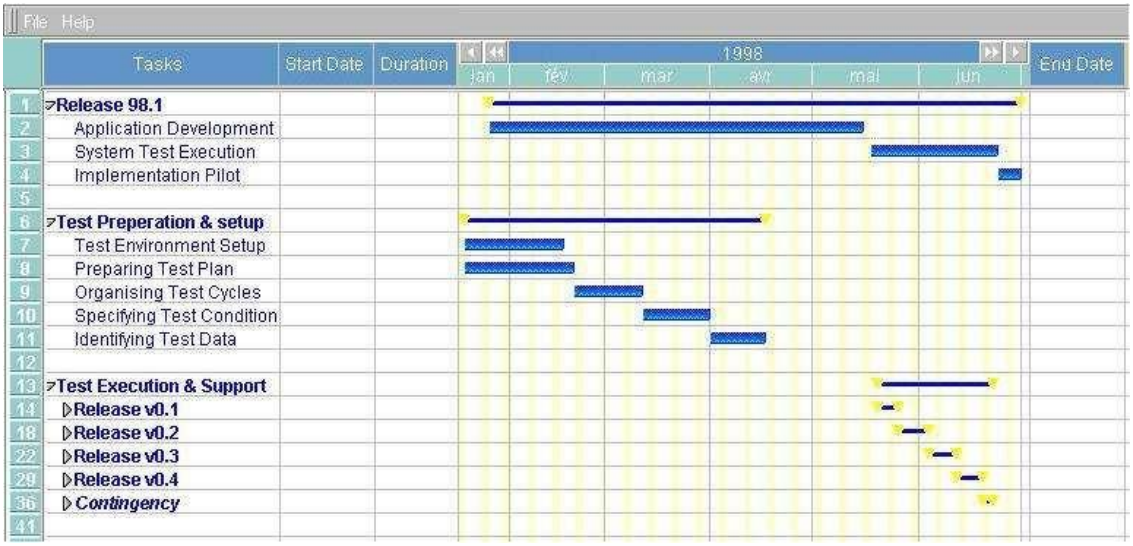
- Resultados do teste de aceitação 1
- Resultados de teste – liberação v2.25.4
- Resultados de teste – liberação v2.25.5
- Resultados de teste – liberação v2.25.6
- Resultados do teste de performance 1
- Resultados dos testes de regressão 1 e 2
- Resultados de teste – liberação v2.25.7
- Resultados do teste técnico

6. CRONOGRAMA DO TESTE DE SISTEMA

Estas são imagens de alguns cronogramas de projeto de alto nível. Estes cronogramas servem somente como exemplos, e provavelmente não correspondem exatamente com o resto do plano de teste. A melhor imagem é a última.




File Help
Tasks
Start Date
Duration
1998
mai
jun
End Date
Test Execution & Support
Release v0.1
Acceptance 1
Functional 1
User Acceptance
Release v0.2
Acceptance 2
Functional 2
Regression 1
Release v0.3
Acceptance 3
Functional 3
Performance 1
Bash/Multi User
Regression 1
Regression 2
Release v0.4
Integration 1
Technical 1
Regression 1
Regression 2
Regression 3
Installation Test
Contingency
Per Bug Fix Test Only


File Help
Tasks
Start Date
Duration
1998
jan
fév
mar
avr
mai
jun
End Date
Release 98.1
Application Development
System Test Execution
Implementation Pilot
Test Preperation & setup
Test Environment Setup
Preparing Test Plan
Organising Test Cycles
Specifying Test Condition
Identifying Test Data
Test Execution & Support
Release v0.1
Release v0.2
Release v0.3
Release v0.4
Contingency

7. RECURSOS

7.1. Humanos

Tipo de recurso	Título	Nº.	Data requisitada	Quem	Status
Gerenciamento de projeto/Funcional	Analista de negócio	1	-	Matheus Domingos Locatelli	Designado
Teste	Controlador de teste	1	-	A ser designado	Designado
	Analista de Teste	1		José Milton de Oliveira Neto	Designado
Testador		1		A ser designado	Designado
Negócio	Especialista do negócio/representante do negócio (PO)	1	1 de maio	Filipe Conde Pereira	Desginado

7.2. Hardware

Um sistema controlado, em separado, será necessário para a fase inicial de teste, montado como um ambiente-padrão do negócio. Para manter a integridade do ambiente de teste, sua rede não deve ser acessível a ninguém de fora do projeto. As impressoras também devem ser exclusivamente para uso da rede de teste.

7.3. Software

7.3.1. Testar Postman

Ferramenta utilizada para testar comunicações com APIs, ele possui ambiente para documentação, testes e todo o tipo de requisições em geral.

7.3.2. Testar ambiente de software

O teste de sistema vai rodar nas seguintes versões de software:

- Windows 10
- Windows 11
- Sistema Operacional Linux
- Mac OS

7.3.3. ClickUP

Ferramenta de produtividade para gerência das etapas do projeto.

Seu uso ajudara a ter uma produção rápida e crescente onde serão gerências as demandas de cada etapa do processo de desenvolvimento do software, assim como a arquitetura de organização das equipes.

7.3.4. Whatsapp

Ferramenta que será usada para comunicação rápida entre as equipes, mas será de maneira informal para com relação ao projeto.

7.3.5. Pacote officer

Ferramenta usada para construção da documentação necessária para a entrega do software.

7.3.6. Microsoft Teams

Ferramenta que será usada para comunicação constante entre as equipes e terá o carate formal. Uma parte das documentações oficiais serão transmitidas através dele.

7.3.7. XMind

Ferramenta que será usada para construção do mapa mental a fim de estrutura uma melhorar visualização das atividades de teste.

7.3.8. Microsoft Project

O Microsoft Project é uma ferramenta simples, porém avançada, para gerenciar o trabalho de projetos rápidos a iniciativas mais complexas.

7.3.9. Open Broadcaster Software

Ferramenta de gravação para documentação em vídeo.

7.3.10. Apache JMeter

Ferramenta de teste de performance.

8. PAPÉIS E RESPONSABILIDADES

8.1. Equipe de teste

Analistas de teste – José Milton de Oliveira Neto;

- Assegurar que a sprint 1 seja concluída dentro do cronograma de entregas, orçamento, qualidade e criar condições de teste detalhadas e de alto nível;
- Produzir os resultados esperados;
- Reportar progressos em reuniões regulares de Status;
- Coordenar revisão e término das condições de teste;
- Gerenciar ciclos individuais de teste e resolver questões/problemas dos testadores;
- Assegurar que os resultados/problemas do teste de sistema sejam reportados imediatamente, e que o acompanhamento seja feito;
- Assegurar que os critérios de entrada sejam alcançados antes que o teste de sistema inicie;
- Assegurar que os critérios de saída sejam alcançados antes do término do teste.

9. RELATÓRIO DE SITUAÇÃO

9.1. Relatório de Situação

A preparação de teste e o progresso de teste devem ser formalmente reportados em uma reunião semanal de status. Quem deve comparecer a esta reunião:

- Toda a equipe.

Um relatório de situação (status) deve ser preparado pelo analista de teste para facilitar esta reunião. Este relatório deve conter as seguintes informações:

1. Status atual X planejamento (Adiantado/Atrasado/No cronograma)
2. Progresso de tarefas planejadas da semana anterior
3. Tarefas planejadas para a próxima semana, incluindo tarefas remanescentes da semana anterior
4. Estatística de erros do sistema de medição de erros
5. Questões/Riscos

Vale salientar que as reuniões semanais são diferentes das reuniões diárias(daily). A primeira possui o carater de recapitulação das ocorrerias diárias a fim de checar o desenvolvimento semanal e dar um feedback ao cliente. Enquanto a segunda possui um carater de report diário para o andamento do processo de desenvolvimento do projeto.

10. QUESTÕES, RISCOS E PREMISSAS

10.1. Questões/riscos

- Mudanças e inclusões no meio do andamento do projeto, exceto: (1) onde houver a permissão e a concordância expressas do analista de negócio e do analista deteste; (2) onde as mudanças/inclusões não requeiram significativo esforço da equipe de teste e não afetem o cronograma de teste. Este é um assunto sério em potencial, pois qualquer grande mudança no desenho vai requerer tempo adicional para replanear o teste e para criar condições de teste suplementares.

- O desenho do software deve ser final, e a documentação deve ser completa, informativa e finalizada por todas as partes antes que o teste de sistema comece.

- Uma fraqueza da abordagem (estratégia) da “entrega por fases” é que o alto grau de interdependência no código significa que a menor mudança pode ter sérios efeitos em áreas do aplicativo que aparentemente não mudaram. O pressuposto da equipe de teste é que funcionalidades previamente entregues e testadas somente requeiram testes de regressão para verificar que elas ainda funcionam, ou seja, os testes não terão a intenção de descobrir novos erros. Por isso recomendamos que haja um mínimo de dois dias de testes de regressão DEPOIS que a correção final tenha sido testada novamente. Isto, porém, impõe uma restrição de tempo na finalização do período de testes, o que requer a concordância do líder do projeto.

- Teste automatizado será presente na maior parte dos testes de regressão por meio de ferramentas de teste automatizado. Entretanto, por causa da carga de trabalho requerida para implantar completamente e eliminar os bugs da ferramenta de teste, é provável que o retorno apenas seja maximizado depois da terceira vez que o teste rodar para cada liberação. Os outros principais usos da ferramenta de teste são: (1) carregar o teste; (2) permitir teste por múltiplos usuários; (3) dar entrada de dados que se repetem; (4) mapear o tempo de

resposta; (5) criar um ambiente sem interferências.

Responsável: Analista de teste

10.2. Premissas

- O software será entregue no prazo;
- O software terá a qualidade requisitada;
- O software não será impactado por mudanças de conformidade feitas na sua estrutura externa. Por exemplo, qualquer mudança externa terá que ser compatível com este aplicativo;
- Todos os bugs que possam comprometer o sistema receberão atenção imediata da equipede desenvolvimento;
- Todos os bugs encontrados em uma versão do software serão consertados e testados individualmente pela equipe de testes antes que uma nova versão seja liberada;
- A funcionalidade é entregue no prazo;
- Os recursos requeridos estarão disponíveis;
- Todos os acordos de serviço serão cumpridos;
- A ferramenta de teste automatizado vai funcionar e interagir corretamente com o software;
- Toda a documentação será atualizada e entregue à equipe de teste de sistema;
- Especificações funcionais e técnicas serão aprovadas pelo negócio.