

Universidade São Judas Tadeu Butantã

Gestão e qualidade de software - CCP1AN-BUE1-6507138

Guilherme de Camargo Leite Eubank Pereira - 822142574 -
822142574@ulife.com.br

Guilherme Farias Menoci - 822135941 - 822135941@ulife.com.br

João Henrique Bezerra dos Santos - RA: 821141558 - 821141558@ulife.com.br

Felipe dos Santos Gamboa - RA: 821141558 - 821137220@ulife.com.br

Sumário

1. Planejamento de testes de software.....	4
1.1. Cronograma de Atividades:.....	4
1.2. Alocação de Recursos.....	4
1.3. Marcos do Projeto.....	4
2. Documentos de desenvolvimento de software.....	5
2.1. Plano de projeto:.....	5
2.1.1. Planejamento do projeto.....	5
Objetivo Geral.....	5
Fases do projeto.....	5
2.1.2. Escopo:.....	7
2.1.3. Recursos:.....	8
2.1.4. Estimativas de projeto.....	9
2.2. Documento de requisitos.....	12
Requisitos Funcionais.....	12
Requisitos Não Funcionais.....	13
Requisitos de Interface:.....	13
Casos de Uso:.....	14
2.3.1 Plano de testes.....	17
2.3.1.1. Introdução.....	17
2.3.1.2. Escopo.....	17
2.3.1.3. Objetivos.....	17
2.3.1.4. Requisitos a serem testados.....	18
Funcionais:.....	18
Não Funcionais:.....	18
2.3.1.5. Estratégias, tipos de testes e ferramentas a serem utilizadas.....	18
2.3.1.6. Recursos a serem empregados.....	19
2.3.1.7. Cronograma das atividades.....	20
2.3.1.8. Definição dos marcos do projeto.....	21
2.3.2. Casos de testes.....	22
3. Gestão de configuração de software;.....	23
3.1. Identificação de Configuração.....	23
3.2. Controle de Configuração.....	23
3.3. Integração Contínua (CI).....	24
3.4. Entrega Contínua (CD).....	24
3.5. Monitoramento e Log.....	24
3.6. Auditoria e Relatórios.....	25
3.7. Conclusão.....	25
4. Repositório de gestão de configuração de software.....	26
4.1. Introdução.....	26
4.2. Políticas de Controle de Versão.....	26
4.2.1. Estrutura de Branches.....	26
4.2.2. Política de Merge.....	26

4.3. Integração Contínua (CI).....	27
4.3.1. Configuração.....	27
4.4. Entrega Contínua (CD).....	27
4.4.1. Configuração.....	27
4.4.2. Automação.....	27
4.5. Monitoramento.....	28
4.5.1. Ferramenta.....	28
4.5.2. Configuração.....	28
4.5.3. Local de Configuração.....	28
4.6. Auditoria e Relatórios.....	28
4.6.1. Logs de Auditoria.....	28
4.6.2. Revisões de Código.....	28
4.6.3. Relatórios Automatizados.....	28
4.7. Benefícios do Repositório.....	29

1. Planejamento de testes de software

1.1. Cronograma de Atividades:

1	Planejamento dos casos de teste
2	Configuração do ambiente de testes
3-4	Testes de unidade
5	Testes de integração
6	Testes de sistema
7	Testes de aceitação pelo usuário
8	Análise e correções finais

1.2. Alocação de Recursos

- **Testador de Software:** 2 pessoas para planejamento e execução dos testes.
- **Desenvolvedor:** 2 pessoas para suporte técnico durante os testes.
- **Ferramentas:** Ambiente de testes com banco de dados e emulador de front-end.

1.3. Marcos do Projeto

1. **Planejamento Completo:** Conclusão da criação dos casos de teste na Semana 2.
2. **Testes de Unidade Concluídos:** Final da Semana 4.
3. **Testes de Integração e Sistema Aprovados:** Semana 6.
4. **Testes de Aceitação e Revisão Final:** Semana 8.

2. Documentos de desenvolvimento de software

2.1. Plano de projeto:

2.1.1. Planejamento do projeto

Objetivo Geral

Desenvolver um sistema que permita o gerenciamento financeiro pessoal, facilitando o registro, categorização e análise de despesas e receitas, com geração de relatórios detalhados para apoiar o usuário na tomada de decisões financeiras.

Fases do projeto

Fase 1: Planejamento e Definição de Requisitos (1 Semana)

- Descrição: Coleta de requisitos, definição de funcionalidades e especificações detalhadas.
- Entregáveis:
 - Documento de requisitos.
 - Esboço inicial da arquitetura do sistema.

Fase 2: Design e Protótipos de Interface (2 Semanas)

- Descrição: Criação do design de interface e protótipos para a experiência do usuário, com base nos requisitos coletados.
- Tarefas:
 - Desenho de telas de login, tela de registro de transações, e visualização de relatórios.
 - Protótipo interativo (usando ferramentas como Figma).
- Entregáveis:
 - Protótipo de interface aprovado.
 - Layout para desenvolvimento.

Fase 3: Desenvolvimento do Back-End (3 Semanas)

- Descrição: Implementação das funcionalidades principais no servidor, como o registro de transações e geração de relatórios.

- Tarefas:
 - Configuração do banco de dados para armazenar transações, categorias e usuários.
 - Implementação da API para registrar transações, com endpoints para consultas e relatórios.
 - Funções de cálculo para total gasto no mês e saldo positivo.
- Entregáveis:
 - Back-end funcional com endpoints testados.
 - Banco de dados configurado e integrado ao sistema

Fase 4: Desenvolvimento do Front-End (3 Semanas)

- Descrição: Construção da interface do usuário e integração com o back-end.
- Tarefas:
 - Implementação das telas de login, registro de transações, e geração de relatórios.
 - Integração com a API para envio e recuperação de dados.
 - Validação e formatação dos dados inseridos pelo usuário.
- Entregáveis:
 - Interface de usuário funcional.
 - Integração completa entre front-end e back-end.

Fase 5: Testes e Validação (2 Semanas)

- Descrição: Testes de funcionalidades e validação da interface, incluindo testes de integração e usabilidade.
- Tarefas:
 - Testes de unidade e integração para o back-end e front-end.
 - Testes de usabilidade com usuários para garantir a intuitividade.
 - Correções de bugs e ajustes finais.
- Entregáveis:
 - Relatório de testes com resultados e correções aplicadas.
 - Aprovação final da versão para lançamento.

Fase 6: Lançamento e Treinamento (1 Semana)

- Descrição: Lançamento da versão final do sistema e treinamento básico para os usuários.
- Tarefas:
 - Preparação do ambiente de produção e implantação do sistema.
 - Documentação do usuário.
 - Sessões de treinamento ou vídeos.
- Entregáveis:
 - Sistema disponível em produção.
 - Documentação e tutorial de uso.

Fase 7: Monitoramento e Suporte Pós-Lançamento (Contínuo)

- Descrição: Monitoramento do sistema e suporte inicial aos usuários.
- Tarefas:
 - Monitoramento de erros e coleta de feedback dos usuários.
 - Planejamento de melhorias e ajustes.
- Entregáveis:
 - Relatório de feedback e lista de futuras melhorias.

2.1.2. Escopo:

Este projeto visa desenvolver um sistema financeiro pessoal que permita aos usuários gerenciar suas finanças de forma eficaz e intuitiva. A principal funcionalidade do sistema é o registro e categorização de despesas e receitas. Os usuários poderão cadastrar diferentes tipos de despesas e lançamentos, selecionando categorias específicas para cada entrada.

Além do registro de transações, o sistema fornecerá uma funcionalidade de geração de relatórios detalhados, ou extratos, para o período escolhido. Esses relatórios incluirão informações essenciais como:

- Tipo de Despesa/Lançamento: Categoria e descrição da transação.
- Valor: Quantia associada a cada despesa ou receita.
- Total Gasto no Mês: Soma das despesas registradas no período.
- Saldo Positivo: Diferença entre o total das receitas e o total das despesas, indicando o saldo final do mês.

O objetivo é oferecer aos usuários uma visão clara e precisa de suas finanças, facilitando o acompanhamento de gastos e a análise do saldo. Com isso, o sistema pretende ajudar na gestão orçamentária e na tomada de decisões financeiras mais informadas.

2.1.3. Recursos:

Recursos Humanos:

- **Testador de Software:** 2 pessoas para planejamento e execução dos testes.
- **Desenvolvedor :** 4 Fullstack Sênior
- **Gerente de Projeto:** 1
- **Analista de Requisitos:** 2

Materiais e Equipamentos:

- **Notebooks : 9**

Recursos Tecnológicos:

1. IDEs:
 - Eclipse
2. Plataformas de controle de Versão/Esteira:
 - Github
 - Github actions
3. Ferramentas de DevOps:
 - Dynatrace
 - Docker
 - Kubernetes
 - GCP

Recursos Financeiros:

1. Salários:

QA	R\$ 3.000,00
Desenvolvedor Fullstack	R\$ 15.000,00
Gerente de Projeto	R\$ 13.000,00
Analista de Requisitos	R\$ 5.500,00

Ferramentas :

GCP	R\$ 20.000,00
Dynatrace	R\$ 3.000,00

2.1.4. Estimativas de projeto

Fase 1: Planejamento e Definição de Requisitos:

- Complexidade: Baixa
- Descrição: Coleta de requisitos, definição de funcionalidades e especificações detalhadas.
- Horas estimadas: 2 dias

Fase 2: Design e Protótipos de Interface:

- Complexidade: Média
- Descrição: Criação de layout e protótipos interativos para a interface do usuário.
- Horas estimadas: 2 semanas

Fase 3: Desenvolvimento do Back-End:

- Complexidade: Alta
- Descrição: Implementação de funcionalidades principais, como registro de transações e geração de relatórios.
- Horas estimadas: 3 semanas

Fase 4: Desenvolvimento do Front-End

- Complexidade: Alta
- Descrição: Construção da interface do usuário e integração com o back-end.
- Horas Estimadas: 3 semanas

Fase 5: Testes e Validação

- Complexidade: Média
- Descrição: Testes de funcionalidades, validação da interface e testes de integração.
- Horas Estimadas: 2 semanas

Fase 6: Lançamento e Treinamento

- Complexidade: Média
- Descrição: Lançamento do sistema em produção e criação de documentação do usuário.
- Horas Estimadas: 1 semana

Fase 7: Monitoramento e Suporte Pós-Lançamento

- Complexidade: Baixa a Média (contínuo)
- Descrição: Monitoramento do sistema e suporte aos usuários.
- Horas Estimadas: **

2.2. Documento de requisitos

Requisitos Funcionais

Código	Descrição	Prioridade
RF01	O sistema deve permitir o registro de despesas e receitas com descrição e valor.	Alta
RF02	O sistema deve permitir a seleção de categorias específicas para cada transação.	Alta
RF03	O sistema deve armazenar e organizar as transações no banco de dados.	Alta
RF04	O sistema deve calcular automaticamente o saldo (receitas - despesas)	Alta
RF05	O sistema deve gerar relatórios financeiros para períodos escolhidos pelo usuário.	Alta
RF06	O sistema deve exibir o total de despesas registradas no período escolhido.	Alta
RF07	O sistema deve exibir o saldo final com base nas receitas e despesas cadastradas.	Alta
RF08	O sistema deve permitir ao usuário editar ou excluir transações registradas.	Média
RF09	O sistema deve suportar exportação de relatórios para formatos como XLS.	Baixa
RF10	O sistema deve oferecer uma interface responsiva, acessível em dispositivos móveis e PC.	Média

Requisitos Não Funcionais

Código	Descrição	Prioridade
RNF01	O sistema deve ter uma interface intuitiva e de fácil navegação.	Alta
RNF02	O sistema deve garantir o armazenamento seguro das informações financeiras dos usuários.	Alta
RNF03	O sistema deve responder a ações do usuário em até 4 segundos.	Alta
RNF04	O sistema deve ser compatível com os navegadores modernos (Chrome, Firefox, Safari, Edge).	Média
RNF05	O sistema deve ser escalável para suportar até 1000 usuários simultâneos.	Baixa
RNF06	O sistema deve ser implementado com tecnologias que facilitem a manutenção e evolução do projeto.	Alta
RNF07	O sistema deve seguir padrões de segurança, incluindo criptografia de dados sensíveis.	Alta

Requisitos de Interface:

O sistema deve ter as seguintes telas:

1. **Tela de Login:** Para autenticação do usuário.
2. **Dashboard:** Visão geral de receitas, despesas, saldo atual e atalho para cadastrar transações.
3. **Tela de Registro de Transações:** Para adicionar despesas e receitas.
4. **Tela de Relatórios:** Exibição de relatórios detalhados.

Casos de Uso:

Caso 1: Registro de Transações

- **Ator Principal:** Usuário.
- **Descrição:** O usuário registra uma nova despesa ou receita, incluindo valor, descrição e categoria.
- **Fluxo Principal:**
 - O usuário seleciona a opção "Nova Transação".
 - Preenche os campos obrigatórios (valor, descrição, categoria).
 - Confirma o registro.
 - O sistema salva a transação no banco de dados.
- **Fluxo Alternativo:**
 - Se faltar algum campo obrigatório, o sistema exibe uma mensagem de erro.

Caso 2: Geração de Relatórios

- **Ator Principal:** Usuário.
- **Descrição:** O usuário solicita um relatório para um período específico.
- **Fluxo Principal:**
 - O usuário seleciona a opção "Gerar Relatório".
 - Escolhe o período desejado.
 - O sistema exibe um relatório com as transações registradas, total gasto, e saldo final.

Caso 3: Edição de Transações

- **Ator Principal:** Usuário.
- **Descrição:** O usuário edita uma transação já registrada para corrigir informações ou alterar valores.
- **Fluxo Principal:**
 - O usuário acessa a lista de transações.
 - Seleciona a transação desejada para edição.
 - Atualiza os campos necessários (valor, descrição, categoria).
 - Confirma a edição.

- O sistema salva as alterações no banco de dados.
- **Fluxo Alternativo:**
 - Se algum campo obrigatório estiver vazio, o sistema exibe uma mensagem de erro.

Caso 4: Exclusão de Transações

- **Ator Principal:** Usuário.
- **Descrição:** O usuário remove uma transação incorreta ou desnecessária do sistema.
- **Fluxo Principal:**
 - O usuário acessa a lista de transações.
 - Seleciona a transação desejada para exclusão.
 - Confirma a exclusão.
 - O sistema remove a transação do banco de dados.
- **Fluxo Alternativo:**
 - Se o usuário cancelar a exclusão, nenhuma ação é realizada.

Caso 5: Cadastro e Login de Usuários

- **Ator Principal:** Usuário.
- **Descrição:** Um novo usuário se cadastra no sistema ou faz login para acessar suas finanças.
- **Fluxo Principal:**
 - O usuário acessa a tela de login/cadastro.
 - Insere as informações necessárias:
 - Cadastro: Nome, número da conta, usuário, e-mail, senha.
 - Login: Usuário e senha.
 - Confirma a ação.
 - O sistema verifica os dados e autentica o usuário.
- **Fluxo Alternativo:**
 - Se as credenciais estiverem incorretas ou já existirem (no cadastro), o sistema exibe um erro.

Caso 6: Exportação de Relatórios

- **Ator Principal:** Usuário.
- **Descrição:** O usuário exporta um relatório em formato XLS.
- **Fluxo Principal:**
 - O usuário seleciona o período desejado para o relatório.
 - Confirma a exportação.
 - O sistema gera o arquivo e disponibiliza o download.
- **Fluxo Alternativo:**
 - Se o relatório for muito extenso, o sistema exibe uma mensagem para ajustar o período.

Caso 7: Logout

- **Ator Principal:** Usuário.
- **Descrição:** O usuário encerra sua sessão no sistema.
- **Fluxo Principal:**
 - O usuário clica no botão "Logout".
 - O sistema encerra a sessão e redireciona para a tela de login.

2.3.1 Plano de testes

2.3.1.1. Introdução

Este plano de testes descreve as atividades necessárias para validar e verificar a funcionalidade, segurança e desempenho do sistema financeiro pessoal. O objetivo é garantir que o sistema atenda aos requisitos funcionais e não funcionais especificados, oferecendo uma experiência confiável e intuitiva ao usuário.

2.3.1.2. Escopo

Os testes abrangem todas as funcionalidades do sistema, incluindo:

- Cadastro, edição e exclusão de transações (despesas e receitas).
- Geração de relatórios financeiros.
- Cadastro e autenticação de usuários.
- Recuperação de senha.
- Validação de segurança (criptografia e acesso).
- Testes de desempenho e responsividade da interface.

2.3.1.3. Objetivos

- Garantir que o sistema funcione conforme os requisitos funcionais e não funcionais.
- Identificar e corrigir erros nas funcionalidades principais antes do lançamento.
- Validar a usabilidade, segurança e desempenho do sistema.
- Verificar a integração entre o front-end e o back-end.

2.3.1.4. Requisitos a serem testados

Funcionais:

- Cadastro, edição e exclusão de transações.
- Geração de relatórios detalhados.
- Cálculo correto de totais e saldo.
- Login, cadastro e recuperação de senha.

Não Funcionais:

- Tempo de resposta do sistema (máximo de 4 segundos por ação).
- Compatibilidade com navegadores modernos (Chrome, Firefox, Safari, Edge).
- Segurança no armazenamento e transmissão de dados (criptografia).

2.3.1.5. Estratégias, tipos de testes e ferramentas a serem utilizadas

Estratégias:

- Adotar a abordagem de testes baseados em requisitos.
- Realizar testes automatizados para casos críticos e repetitivos.
- Executar testes manuais para validação de interface e usabilidade.

Tipos de Testes:

- Testes de unidade: Garantir o funcionamento correto de componentes individuais (ex.: APIs e funções de cálculo).
- Testes de integração: Verificar a comunicação entre o front-end e o back-end.
- Testes de sistema: Validar o sistema completo com todos os componentes.
- Testes de aceitação: Validar o sistema com base na perspectiva do usuário.
- Testes de desempenho: Avaliar o tempo de resposta e carga do sistema.
- Testes de segurança: Validar criptografia e proteção contra acessos não autorizados.

Ferramentas:

- Testes automatizados: Selenium, Cypress.
- Testes de desempenho: Apache JMeter, Postman.
- Gerenciamento de testes: Jira, TestRail.
- Segurança: JWT

2.3.1.6. Recursos a serem empregados

Equipe:

- 1 Analista de Testes para criar os casos e roteiros de testes.
- 1 Desenvolvedor para suporte técnico durante os testes.
- 1 Gerente de Projeto para acompanhar e gerenciar os resultados.

Hardware:

- Computadores para testes em diferentes navegadores e dispositivos.

Software:

- Ferramentas de automação, gerenciadores de casos de teste e navegadores.

2.3.1.7. Cronograma das atividades

Atividade	Responsável	Duração	Início	Término
Planejamento dos casos de teste	Analista de Testes	5 dias	05/08/2024	09/08/2024
Configuração do ambiente de testes	Desenvolvedor/Dev Ops	3 dias	12/08/2024	14/08/2024
Testes de unidade	Desenvolvedor	10 dias	15/08/2024	28/08/2024
Testes de integração	Analista de Testes	7 dias	29/08/2024	06/09/2024
Testes de sistema	Analista de Testes	7 dias	09/09/2024	17/09/2024
Testes de aceitação pelo usuário	Equipe e Usuários	5 dias	18/09/2024	24/09/2024
Análise dos resultados	Analista de Testes	2 dias	25/09/2024	26/09/2024
Correções e ajustes finais	Desenvolvedor	5 dias	27/09/2024	03/10/2024

Resumo do Cronograma

- 1. **Data de Início:** 05 de agosto de 2024.
- 2. **Data de Término:** 03 de outubro de 2024.
- 3. **Período Total:** Aproximadamente 2 meses.

Indicadores de Sucesso

- 1. **Planejamento Concluído:** 100% dos casos de teste documentados.
- 2. **Testes de Unidade:** Cobertura mínima de 85% do código.

- 3. **Testes de Integração e Sistema:** Passagem de pelo menos 95% dos casos de teste críticos.
- 4. **Testes de Aceitação:** Aprovação de pelo menos 90% dos critérios definidos pelos usuários.

2.3.1.8. Definição dos marcos do projeto

Marco	Descrição	Data de Conclusão
Planejamento Concluído	Finalização do planejamento dos casos de teste, com todos os cenários documentados.	09/08/2024
Ambiente de Testes Configurado	Configuração completa do ambiente de testes, incluindo banco de dados e ferramentas.	14/08/2024
Conclusão dos Testes de Unidade	Finalização dos testes de unidade, garantindo que os componentes individuais funcionem.	28/08/2024
Conclusão dos Testes de Integração	Verificação de comunicação entre os componentes do sistema concluída com sucesso.	06/09/2024
Conclusão dos Testes de Sistema	Validação do sistema completo, incluindo funcionalidade e requisitos não funcionais.	17/09/2024
Conclusão dos Testes de Aceitação	Testes realizados pelos usuários finais, aprovando o sistema para produção.	24/09/2024
Correções e Ajustes Finais Concluídos	Finalização de todas as correções identificadas durante os testes e preparação para release.	03/10/2024

2.3.2. Casos de testes

Caso de teste	Descrição	Requisito
CT01: Registro de Transações	Validar o registro de receitas e despesas com descrição e valor.	RF01
CT02: Seleção de Categorias	Verificar a seleção correta de categorias nas transações.	RF02
CT03: Cálculo de saldo	Validar o cálculo automático de saldo (receitas - despesas).	RF04
CT04: Geração de Relatórios	Validar a geração de relatórios detalhados para o período selecionado.	RF05
CT05: Exportação de Relatórios	Testar a exportação de relatórios no formato XLS.	RF09
CT06: Interface Responsiva	Validar a compatibilidade da interface em dispositivos móveis e desktop.	RF10, RNF04
CT07: Segurança de Dados	Validar criptografia e proteção de dados armazenados.	RNF02. RNF07
CT08: Tempo de Resposta	Medir o tempo de resposta das ações (máximo de 4 segundos).	RNF03
CT09: Compatibilidade	Testar o sistema nos principais navegadores modernos (Chrome, Firefox, etc.).	RNF04

3. Gestão de configuração de software;

3.1. Identificação de Configuração

Itens a serem gerenciados

- **Código-fonte:**
 - Backend: Java com Spring.
 - Frontend: React.
- **Banco de Dados:**
 - Scripts de migração do PostgreSQL (Flyway).
- **Infraestrutura:**
 - Arquivos Dockerfile e configurações de Kubernetes (YAML).
- **Documentação:**
 - Manual do usuário, especificação técnica e guias de instalação.
- **Configurações de monitoramento:**
 - Configurações de Dynatrace para métricas e logs.

3.2. Controle de Configuração

Controle de Versão

- **Ferramenta:** Git/GitHub.
- **Estrutura do Repositório Git:**

Fluxo Git

- **Branch principal:** `main` (versão estável).
- **Branches de desenvolvimento:**
 - `feature/<nome-da-feature>`: Novas funcionalidades.
 - `bugfix/<nome-do-bug>`: Correção de bugs.
 - `hotfix/<nome-do-hotfix>`: Correção urgente de problemas na produção.
- **Política de Merge:**
 - Pull Requests (PRs) devem passar por revisão de código antes de serem integrados.
 - Integração somente após passar nos testes automatizados.

3.3. Integração Contínua (CI)

Ferramenta: GitHub Actions

- Configuração de **workflow** para CI em [.github/workflows/ci.yml](#).

Etapas:

1. Clonar o repositório.
2. Construir e testar o backend.
3. Construir e testar o frontend.
4. Validar scripts de banco de dados.

3.4. Entrega Contínua (CD)

Ferramentas: Docker e Kubernetes

- **Docker:**
 - Criação de imagens para o backend e frontend.
- **Pipeline de CD:**
 - Construir imagens Docker para backend e frontend.
 - Enviar imagens para o Docker Hub.
 - Implantar usando Kubernetes.

3.5. Monitoramento e Log

Ferramenta: Dynatrace

- **Configurar Dynatrace para:**
 - Monitoramento de métricas (uso de CPU, memória, número de requisições, etc.).
 - Coleta de logs (integrado com Docker/Kubernetes).
- **Configuração inicial:**
 - Adicionar o **Dynatrace OneAgent** às instâncias no Kubernetes.
 - Configurar alertas para desempenho ou falhas críticas.

3.6. Auditoria e Relatórios

- Revisões de código semanais para garantir conformidade com padrões de qualidade.
- Logs de auditoria em cada pipeline (CI/CD) para rastrear mudanças.
- Relatórios gerados automaticamente em caso de falhas na integração contínua ou monitoramento.

3.7. Conclusão

A implementação deste plano de SCM garante organização, rastreabilidade e automação do ciclo de vida do sistema de gestão financeira, promovendo maior qualidade e estabilidade.

4. Repositório de gestão de configuração de software

4.1. Introdução

Este documento define o repositório de gestão de configuração de software para o sistema financeiro pessoal, detalhando sua estrutura, conteúdo e políticas de uso. O repositório será a base para armazenamento, versionamento e controle dos itens de configuração, garantindo organização, rastreabilidade e suporte ao desenvolvimento e implantação.

4.2. Políticas de Controle de Versão

4.2.1. Estrutura de Branches

- **Branch Principal (**main**):**
 - Contém a versão estável do sistema.
- **Branches de Desenvolvimento:**
 - **develop**: Integração contínua de funcionalidades em desenvolvimento.
 - **feature/<nome-da-feature>**: Implementação de novas funcionalidades.
 - **bugfix/<nome-do-bug>**: Correção de bugs identificados.
 - **hotfix/<nome-do-hotfix>**: Correções emergenciais aplicadas à produção.

4.2.2. Política de Merge

- Todo código deve passar por um Pull Request (PR).
- Revisão de código obrigatória antes da integração ao **main**.
- Testes automatizados (unitários, integração e sistema) devem ser executados e aprovados no CI antes do merge.

4.3. Integração Contínua (CI)

4.3.1. Configuração

- **Ferramenta**: GitHub Actions.
- **Local de Configuração**: **.github/workflows/ci.yml**.

- **Etapas:**

1. Clonar o repositório.
2. Construir e testar o back-end:
 - a. Compilar o código Java com Spring Boot.
 - b. Executar testes de unidade e integração.
3. Construir e testar o front-end:
 - a. Compilar o código React.
 - b. Executar testes de UI automatizados.
4. Validar scripts de banco de dados:
 - a. Executar scripts de migração (Flyway).

4.4. Entrega Contínua (CD)

4.4.1. Configuração

- **Ferramentas:** Docker, Kubernetes.
- **Etapas do Pipeline:**
 1. Criar imagens Docker para o back-end e front-end.
 2. Fazer upload das imagens para o Docker Hub.
 3. Implantar o sistema no Kubernetes:
 - Aplicar configurações de `deployment.yaml`, `service.yaml`.

4.4.2. Automação

Os scripts de automação para entrega contínua estão armazenados no diretório `ci_cd/cd_pipeline/`.

4.5. Monitoramento

4.5.1. Ferramenta

- **Dynatrace:** Configurado para monitorar métricas e logs do sistema.

4.5.2. Configuração

- Adicionar o Dynatrace nas instâncias Kubernetes.

- Configurar alertas para:
 - Uso excessivo de CPU ou memória.
 - Erros críticos nas aplicações.
 - Lentidão nas respostas do sistema.

4.5.3. Local de Configuração

- Diretório: [monitoring/dynatrace/](#).

4.6. Auditoria e Relatórios

4.6.1. Logs de Auditoria

- Gerados automaticamente pelos pipelines de CI/CD.
- Incluem histórico de builds, testes e deploys.

4.6.2. Revisões de Código

- Realizadas semanalmente para garantir conformidade com os padrões estabelecidos.

4.6.3. Relatórios Automatizados

- Gerados em caso de falhas ou problemas críticos no monitoramento.

4.7. Benefícios do Repositório

1. **Organização:** Estrutura clara para diferentes itens de configuração.
2. **Colaboração:** Suporte ao trabalho em equipe com controle de versão eficiente.
3. **Rastreabilidade:** Histórico completo de mudanças e revisões.
4. **Automação:** Integração e entrega contínuas configuradas para minimizar erros manuais.
5. **Escalabilidade:** Preparado para gerenciar ambientes complexos e distribuídos.