Plano de Testes - Sistema Chronos

Versão: 1.0
Criado: 25/11/2022
Última atualização: 05/12/2022
Status: Primeira Release

Tabela de Conteúdos

1. Introdução

1.1. Objetivos do plano de testes

O presente documento tem como objetivo lançar os pilares aos quais o ciclo de vida de teste do software (STLC) Chronos irá se sustentar, bem como descrever processos, métricas, ferramentas a serem utilizadas, ambientes e cenários que serão considerados neste STLC.
Vale salientar que não será contemplado por este documento os testes unitários

1.2. Visão geral do projeto

O Chronos é um sistema de gerenciamento de cronograma para o programa de estágio 'Vem Ser DBC', que consiste em uma página web criada em react-js conectada a um banco de dados por meio de uma API REST feita em Java. A aplicação conta com: 3 perfis de uso, CRUD de usuários, CRUD de dados do cronograma e geração de calendário.

2.Requisitos

2.1. Requisitos funcionais

- Página inicial
  - A página inicial do sistema, caso o usuário não esteja logado, é a de login (item 5).
  - Se estiver logado tem que aparecer mensagem de boas-vindas, Página Principal (item 2,3,4).
- Página Principal (Acessando como administrador)
  - Deve ter um cabeçalho com os dados do usuário e foto.
    - Nome do Usuário
    - Imagem do Usuário
  - Logado como administrador a página principal deve ter um cadastro de colaborador (item 6) e verificar cronograma (item).

- Página Principal (Acessando como gestor de pessoas)
  - Deve ter um cabeçalho com os dados do usuário e foto.
    - Nome do Usuário
    - Imagem do Usuário
  - Logado como gestor de pessoas a página principal deve ter um cadastro de edição (item), cadastro de etapa (item), cadastro de processo (item), gerar calendário(item) e verificar cronograma (item).

- Página Principal (Acessando como instrutor)
  - Deve ter um cabeçalho com os dados do usuário e foto.
    - Nome do Usuário
    - Imagem do Usuário
  - Logado como instrutor, a página deverá ter um verificar cronograma (item).

- Login
  - Deve ser implementada uma página de login com campos usuário e senha (ambos obrigatórios)
  - A página de login deve ser acessada sempre que um usuário não estiver com uma sessão / token válido e tentar acessar alguma página
  - Ao realizar login com sucesso, o usuário deve ser redirecionado para a página inicial.

- Página de cadastro de colaborador
  - Página onde o administrador poderá cadastrar um colaborador.
    - Nome completo
    - Upload de foto de perfil (deve ser salvo no servidor) tem que ser opcional.
    - Deverá poder selecionar qual o tipo de cadastro o usuário irá receber (Gestão de pessoas, Instrutor ou Administrador).
    - O login deverá ser único para cada usuário cadastrado
      - Utilizar o e-mail da DBC (@dbccompany.com.br)

        - Os usuários cadastrados deverão ser listados e podem ser editados ou excluídos pelo administrador.
        - Menu ou abas para as demais páginas autorizadas.
    - ➢ Listar todos os colaboradores

- ❖ Perfil
    - ➢ página para editar o perfil do usuário (TODOS)
        - o usuário pode alterar as seguintes informações:
            - nome
            - foto de perfil
            - senha
        - o usuário pode alterar a senha, desde que informe a senha antiga
        - deve ter um campo de confirmação de senha

- ❖ Cadastrar a edição do vem ser
    - ➢ Página onde o gestor de pessoa poderá cadastrar a edição do Vem Ser
    - ➢ Deve ser aberto um formulário com os seguintes campos
        - edição do Vem Ser
        - data de início
        - data de término

- ❖ Cadastro de etapa
    - ➢ Página onde o gestor de pessoa poderá cadastrar uma etapa.
        - exemplos: (Preparação, Contrato, Treinamento, Contratação)
            - Nome
    - ➢ Pode ser cadastrada outras etapas

- ❖ Cadastro de processo
    - ➢ Página onde o gestor de pessoa poderá cadastrar um processo.
        - Selecionar uma etapa
        - Nome (exemplo: etapa 1, etapa 2)
        - Ordem de execução
        - Área envolvida (gestão de pessoas, instrutores, DP etc.)
        - Responsável (gestão de pessoas, instrutores, DP etc.)
        - Duração do processo (dias ou semanas)
        - Dias Úteis
        - Edições

- ❖ Cadastro de período não útil
  - ➢ Página onde o gestor de pessoas poderá cadastrar um período não útil
    - conterá os seguintes campos:
      - Descrição
      - Data Inicial
      - Data Final
      - Deve permitir selecionar se deve repetir para todos os anos
- ❖ Gerar calendário
  - ➢ Página onde o gestor de pessoa poderá gerar o calendário da edição do Vem Ser.
  - ➢ Após cadastro de todas as etapas e processos selecionar a edição do Vem Ser e gerar o calendário com todos os dados
  - ➢ No calendário terá que constar os seguintes dados:
    - Edição do Vem Ser
    - Data do início e término
    - Listagem das etapas e seus processos
  - ➢ Ao gerar o calendário, deve ser feita uma cópia de todos os dados daquela geração atual (processos, etapas, dias não uteis) para edição gerada (para manter o calendário sem alterações).
- ❖ Ver calendário
  - ➢ Página onde qualquer colaborador poderá ver o cronograma da edição escolhida do Vem Ser.

2.2. Requisitos não funcionais

2.2.1. Front-End

- O sistema deve ser navegável apenas com o usuário devidamente logado no sistema
- Deve ser construído utilizando a biblioteca react com typescript
- Deve utilizar como ferramenta de gerenciamento de estado Context ou Redux
- A aplicação frontend deverá ser publicada no Vercel.
- Deve utilizar a biblioteca de estilização Material UI.
- Deve implementar IDs nos componentes para que a qualidade consiga realizar os testes automatizados.
- O sistema deve ser obrigatoriamente responsivo.

- Deve ter no mínimo 60% de testes unitários utilizando React Test Library ou Jest (somente em componentes).
- A paginação é obrigatória nas principais páginas que utilizam GET.

2.2.2. Back-End

- A Paginação é obrigatória nos principais endpoints GETS
- Backend deve ser desenvolvido utilizando Spring Boot, com RestController, Service, Repository, Entity, respeitando a responsabilidade de cada classe.
- Deve ser utilizada autenticação com Token JWT.
- Deve ter 100% de cobertura de testes unitários na camada de serviço (services).
- A aplicação deverá ser publicada na DBC.

2.2.3. QA

- Testes automatizados de Front-end devem ser desenvolvidos utilizando a ferramenta Selenium com Java ou Cypress com JavaScript.
- Testes automatizados de API devem ser desenvolvidos utilizando a ferramenta REST-Assured com Java ou Cypress com JavaScript.
- Os testes de qualidade devem cobrir no mínimo 90% da aplicação sendo: a. 20% testes manuais.
- b. 70% testes automatizados.
- Deve ser elaborado o plano de testes.
- O teste automatizado deve ser reportado utilizando a ferramenta Allure Report.
- Os usuários não podem ter acesso a dados de outros usuários.
- Disponibilizar o código no github.

3.Estratégia de teste

3.1. Princípios de teste

- Os testes serão divididos em fases distintas, cada uma com objetivos e metas claramente definidas.
- Não haverá uma descrição completa dos cenários de testes. Informações sobre os cenários serão repassadas detalhadamente para a equipe de desenvolvimento em caso de bug.
- Atividades de teste serão construídas em cima dos estágios anteriores para evitar redundância e repetição do esforço.

3.2. Suposições sobre o sistema

- Testes exploratórios serão conduzidos manualmente em cada feature a partir do momento que for liberado pelo time de desenvolvimento, podendo serem reconduzidos a qualquer momento.
- O sistema será tratado como uma caixa-preta, caso as informações estejam aparecendo corretamente será considerado que o banco de dados está operando corretamente, descartando a necessidade de testar a persistência dos dados.
- A equipe de testes supõe que todas as inputs necessárias durante a fase de testes serão prontamente providenciadas pelo time de desenvolvimento e pelo time de negócio.
- Bugs serão reportados através do board da equipe no Trello e pelo canal da equipe no discord.
- O ambiente de testes e sua preparação serão uma responsabilidade compartilhada entre o time de desenvolvimento e o time de testes

3.3. Features a serem/não serem testadas

Todas as funcionalidades descritas no documento serão testadas, contudo, alguns itens como roteamento e autorização de páginas no front, recuperação de senha e etc serão testados de forma completa exclusivamente de forma manual por seu grau de complexidade.

3.4. Abordagem de dados

- Para todos os testes, será requerido o acesso a usuários com 3 tipos de acesso: um usuário com acesso de ‘ADMIN’, um usuário com acesso de Gestor e um usuário de acesso total, pré carregado no banco de dados.
- Para os testes automatizados, todo e qualquer outro dado necessário será carregado no banco de dados no momento da execução, devendo ser prontamente deletado do sistema ao final da tarefa.

3.5. Critérios de suspensão e resumo de testes

Os testes da aplicação só serão pausados nos casos de: má função do servidor, componentes para testes indisponíveis ou falha crítica demonstrada em algum componente chave que impeça a realização dos testes seguintes.

Os testes devem ser resumidos imediatamente após a remoção do impedimento que gerou a suspensão.

3.6. Features que serão testadas exclusivamente de forma manual

As features que serão testadas apenas de forma manual são:

Front:

- Geração de calendários
- Edição de perfil com sucesso

Ambos:

- Upload de foto de perfil

4. Ferramentas e versões:

1. Teste manual

- Postman v10.1.2
- Navegador Chrome v107.0.5304.122

2. Teste automatizado

Compartilhados:

- Java v8
- Gson v2.8.7

Front:

- Selenium v3.141.59
- Junit v4.13.2
- Allure-Junit v2.19
- Java Faker v1.0.2

Back

- Rest Assured v4.4.0
- Junit v5.8.0-M1
- Allure Junit v2.14.0
- Data Faker v1.6.0

5. Tipos de teste e escopo

O plano de testes abrange todas as funcionalidades do sistema, utilizando os tipos de teste mais oportunos para cada funcionalidade.

Os testes unitários farão parte das responsabilidades do time de desenvolvimento e não serão contemplados neste documento.

5.1. Testes exploratórios

Propósito: O propósito destes testes é dar uma visão geral sobre como está a funcionalidade do projeto, comparando os resultados obtidos com a documentação, e também assegurar que falhas críticas sejam removidas antes do início dos testes automatizados.

Escopo: Toda aplicação.

Método: A aplicação será manipulada manualmente sem qualquer tipo de script, roteiro ou documentação.

Ferramentas: Navegador Chrome para os testes de Front, POSTMAN e o swagger para os testes de Back.

5.2. Testes de contrato

Propósito: O propósito desses testes é testar a API, validando as respostas obtidas nos métodos GET com um esquema JSON previamente definido

Escopo: Métodos GET da API

Método: Os testes serão executados por automação com base num script.

Ferramentas: Rest Assured

5.3. Testes de aceitação

Propósito: O propósito dos testes de aceitação é fazer a verificação das regras de negócio em toda a aplicação e servir como validação do funcionamento da mesma e será realizado majoritariamente no back end

Escopo: Toda aplicação

Método: O teste vai ser executado de acordo com os scripts de testes gerados a partir dos cenários de teste

Ferramentas: Rest Assured para backend e Selenium para front end

5.4. Testes E2E

Propósito: Os testes e2e tem como proposta simular a operação da aplicação por um usuário real, validando assim a aplicação como um todo num cenário mais concreto.

Escopo: Módulos críticos da aplicação

Método: O teste será executado tanto manualmente, quanto automaticamente no front-end

Ferramentas: Navegador e Selenium

6. Estratégia de execução

6.1. Ciclos de teste

Durante o STLC, diversas baterias de testes serão realizadas gradativamente, sendo estas acionadas por uma entrega do time de desenvolvimento ou pelo cronograma, estando organizados da seguinte forma:

- Testes exploratórios serão executados em 3 ocasiões distintas: após cada funcionalidade entregue pelo time de desenvolvimento na funcionalidade que foi disponibilizada, no começo de cada jornada de trabalho do time de teste na funcionalidade que tiver testes automatizados sendo implementadas e aleatoriamente, em funcionalidades já entregues pelo time de desenvolvimento, quando o time de teste julgar necessário.
- Testes de contrato serão executados após a entrega de um endpoint crítico com operação get por parte do time de desenvolvimento de backend e só serão repetidos caso haja alteração no código relevante a esse endpoint ou no final do ciclo de vida dos testes.
- Testes de aceitação serão realizados a cada feature maior a ser entregue e ao início de cada dia, sendo repetidos como testes de regressão cada vez que um novo teste de aceitação for liberado para execução.

6.2. Validação e gerenciamento de falhas

É esperado que todos os testes sejam executados de acordo com os requisitos validados pelo PO do projeto e dentro dos ciclos que estão descritos acima. Contudo, se reconhece que a equipe de teste pode (e deve) construir e conduzir testes adicionais, principalmente quando for identificado alguma lacuna entre os testes.

É responsabilidade do time de teste observar os bugs/defeitos que ocorrem durante o teste, descrever qual operação que resultou neste bug e reportar para o time de desenvolvimento, além de atribuir grau de severidade, retestar o cenário quando necessário além de manipular seu status.

É responsabilidade do time de desenvolvimento comunicar com o time de qa quando os testes precisarem parar ou puderem continuar, verificar no board os detalhes do bug/defeito a ser corrigido, pedir mais detalhes se for necessário, implementar as correções solicitadas e requisitar um novo teste quando o reparo for concluído.

Qualquer malfunção do sistema será categorizada por gravidade, de acordo com os impactos causados na aplicação da seguinte maneira:

Gravidade	Impacto
1 (Blocker)	• O bug é grave o suficiente para quebrar o sistema, causar corrupção de arquivos ou perda de dados

	• Causa um retorno anormal ao sistema, gerando um crash ou mensagem de erro de sistema • Faz a aplicação ficar irresponsiva e necessitar de um reboot para voltar a operar
2(Crítica)	• Causa a falta de uma funcionalidade vital do programa
3(Normal)	• Causa um impedimento de uso de alguma funcionalidade, mas existe uma 'gambiarra' que torna a utilização possível • O bug impede que certas áreas sejam testadas, mas existe a possibilidade de testar estas áreas de forma independente
4(Menor)	• O bug permite que ações não válidas sejam executadas, mas sem ferir a regra de negócio ou resultar num defeito.
5(Trivial)	• Algum item visual está fora do padrão, mensagens de erro ou sucesso insuficiente ou não clara.

6.3. Métricas de teste

As métricas para a análise do progresso e nível de sucesso do STLC será desenvolvido e compartilhado com o time de desenvolvedores e com o Product Owner para aprovação, sendo elas:

Relatório	Descrição	Frequência
Status da preparação de testes e execução	Relatar a porcentagem de testes completos, passados, falhados Relatar os status e severidade dos bugs previamente encontrados	diária
Status das execuções diárias	Reportar os testes passados, falhados, total de falhas	diária
Relatório de status geral da automação do projeto	Relatório contemplando todos os testes automatizados realizados no projeto	a cada 2 dias

7. Riscos

Os riscos que ameaçam este STLC certamente são eventos/anomalias que podem causar um efeito extremamente negativo no progresso das atividades e, por isso, serão identificados e categorizados neste plano para posterior tratamento prévio.

Risco	Impacto	Prob.	Plano de mitigação
Cronograma Cronograma de testes é apertado e se os testes forem atrasados por algum motivo não existe a possibilidade extensão do cronograma	ALTO	ALTO	A preparação dos testes pela equipe de QA irá iniciar antes mesmo das features estiverem completas.
Equipamento Algum problema ocorre nos equipamentos da equipe de QA, falha em conexão de rede	ALTO	BAIXA	Projeto instalado e configurado em múltiplas máquinas, atualização de repositório remoto constante, mais de uma fonte de internet.
Bugs Bugs encontrados num estágio avançado do ciclo de vida do teste.	MÉDIO	MÉDIA	Execução constante dos cenários de teste, comunicação constante sobre o status da aplicação com o time de desenvolvimento
Funcionalidades Surgimento de nova funcionalidade a ser implementada no sistema durante o projeto ou features alteradas no decorrer do processo.	MÉDIO	BAIXA	Escopo do teste bem definido, testes automatizados criados de forma a serem facilmente alterados
Cronograma Testes sendo adiados por causa de Bugs	MÉDIO	ALTA	Procedimentos de validação e gerenciamento de falhas bem definido para que as soluções sejam entregues imediatamente Análise de todos os requisitos a fundo com o PO para evitar erros devido a documentação de projeto obscura.

8.Cronograma

	Semana 1	Semana 2
Domingo	Criação do cenário de testes de login Início da configuração do projeto de automação - Back Expansão deste documento.	Criação dos testes e2e front Criação dos relatórios do allure Ajuste do relatório final Organização dos últimos detalhes do projeto
Segunda	Execução dos testes de login Início do projeto de automação - Front Expansão do planejamento de cenários de teste Edição do plano de testes	Entrega do Projeto.
Terça	Criação dos testes de usuário - back Criação dos testes de criar colaborador - back Edição do plano de testes	X
Quarta	Finalização dos testes de criar colaborador - back Criação dos testes de edição -back	X
Quinta	Criação dos testes de processos - back Criação dos testes de etapa -back Criação dos testes de login - front	X
Sexta	Criação dos testes de calendário Criação dos testes de dia útil Finalização dos cenários	X
Sábado	Criação dos testes e2e front Organização dos testes do front	X

9.Relatórios e resultados

Durante o STLC serão produzidos dois relatórios finais com os resultados da automação, utilizando a ferramenta allure-report, contendo o resultado de todos os testes automatizados executados durante o processo além de duas tabelas, contendo a descrição e o status de todos os testes executados, uma pro front e outra pro back.

Apêndice A - Histórico do documento

Versão	Data	Descrição das mudanças
0.1	26/11	Rascunho geral do documento
0.2	27/11	Criação de alguns cenários de teste, expansão das estratégias de teste
0.3	28/11	Finalização do cronograma, edição das secções estratégias e tipos de teste
0.4	29/11	Criação dos cenários de teste de backend relacionados a usuário
0.5	04/12	Remoção dos cenários (não há tempo hábil para finalizá-los), remoção das referências ao mesmo
1.0	05/12	Editado para a primeira release