INSTITUTO FEDERAL
Rio Grande do Norte
Campus Natal-Central

Plano de Testes

Desenvolvimento de Sistemas

Professor:

Marcelo Romulo Fernandes

Equipe:

Breno Nascimento de Almeida

Christian Bruno Gomes S. da Silva

Deivdson Pereira Oliveira da Silva

Jefferson Ferreira Ribeiro

Lucas Melo Nascimento dos Santos

Julho / 2021

Histórico de Revisões

Data	Versão	Descrição	Autor
<06/jul/23>	<1.0>	Definição inicial do documento de testes	Breno Nascimento de Almeida Christian Bruno Gomes S. da Silva Deivdson Pereira Oliveira da Silva Jefferson Ferreira Ribeiro Lucas Melo Nascimento dos Santos

Objetivos

O documento do plano de testes do Daily Schedule tem os seguintes objetivos:

- Identificar informações de projeto existentes e os componentes de software que devem ser testados.
- Listar os Requisitos a Testar recomendados (alto nível).
- Recomendar e descrever as estratégias de teste a serem empregadas.
- Identificar os recursos necessários e prover uma estimativa dos esforços de teste.
- Listar os elementos resultantes do projeto de testes.

O Daily Schedule

O Daily Schedule é um projeto que visa solucionar o problema da desorganização enfrentado pelos estudantes em suas demandas diárias. A plataforma permite aos usuários, que incluem estudantes, administradores e visitantes, organizar e gerenciar suas rotinas e estudos de forma eficiente.

Os estudantes podem criar cronogramas personalizados, cadastrar tarefas e visualizar seu progresso acadêmico. A interface intuitiva e minimalista do Daily Schedule facilita a usabilidade, enquanto recursos como lembretes e notificações por e-mail ajudam os estudantes a se manterem atualizados e cumprirem prazos importantes.

Além disso, o sistema oferece opções de compartilhamento e exportação de cronogramas, permitindo que os estudantes colaborem e tenham acesso aos seus horários em diferentes dispositivos.

Com foco exclusivo na organização acadêmica, o Daily Schedule se destaca como uma alternativa eficiente para melhorar a produtividade e o rendimento escolar dos estudantes.

Escopo

Os testes realizados serão: testes estáticos, testes unitários das classes de domínio, testes de integração dos repositórios da camada de persistência das classes de domínio e os testes de sistema dos controladores. Ainda nesse contexto, os testes críticos desse plano são os testes de integração e sistema.

Os testes que não serão realizados: teste funcional, teste do ciclo de negócios, teste da interface do usuário, teste de carga, teste de stress, teste de volume, teste de segurança e de controle de acesso, teste de recuperação e teste de instalação.

Identificação de Projeto

A tabela abaixo identifica a documentação e disponibilidade usados para desenvolver o plano de testes:

Documento	Criado ou Disponível	Recebido ou Revisado
Especificação de Requisitos	■ Sim ☐ Não	■ Sim ☐ Não
Plano de Projeto	■ Sim ☐ Não	■ Sim ☐ Não
Modelo de Análise	■ Sim ☐ Não	■ Sim ☐ Não
Modelo de Projeto	■ Sim ☐ Não	■ Sim ☐ Não
Documento de Arquitetura	■ Sim ☐ Não	■ Sim ☐ Não
Protótipo	■ Sim ☐ Não	■ Sim ☐ Não
Manual do Usuário	☐ Sim ■ Não	☐ Sim ■ Não
Lista de Riscos	■ Sim ☐ Não	■ Sim ☐ Não

Requisitos a testar

O plano de testes no contexto do PDS corporativo teve como foco testar requisitos do backend na aplicação Dailyschedule.

Teste do Backend

- Verifique que as entidades principais da camada de domínio possuem métodos de igualdade e comparação natural.
- Verifique que os métodos de igualdade das principais entidades de domínio confirmem se de fato uma instância é igual a outra ou não.
- Verifique que os métodos de ordenação natural das entidades de domínio confirmem qual entidade será posta antes ou depois de outra, considerando a modelagem particular de cada classe de domínio.
- Verifique que as entidades de domínio estão corretamente persistidas.
- Verifique que as entidades de domínio sejam armazenadas corretamente ao serem criadas.
- Verifique que as entidades de domínio sejam corretamente recuperadas da camada de persistência.
- Verifique que as atualizações nos atributos das entidades de domínio sejam corretamente refletidas na camada de persistência.
- Verifique que as entidades de domínio deixem de existir na camada de persistência ao serem removidas.
- Verifique que as camadas de aplicação estejam sendo respeitadas durante a implementação.
- Verifique que os controladores da API do backend estejam implementados, acessíveis e documentados.
- Verifique que as entidades de domínio sejam corretamente cadastradas através do uso da API.
- Verifique que as entidades de domínio sejam corretamente recuperadas através do uso da API.
- Verifique que as entidades de domínio sejam corretamente atualizadas através do uso da API.
- Verifique que as entidades de domínio sejam corretamente removidas através do uso da API.
- Verifique que o usuário conseguirá se autenticar para realizar o login através do uso da API.
- Verifique que o usuário conseguirá realizar o logout e ter sua sessão removida através do uso da API

Estratégia de Teste

No contexto do plano de testes, temos os seguintes tipos de testes:

- Testes estáticos: Revisões do código ou documentação para identificar problemas antes da execução do software.
- Testes unitários das classes de domínio: Testes das unidades individuais de código para garantir que funcionem corretamente.
- Testes de integração dos repositórios da camada de persistência: Testes da interação entre as classes de domínio e os repositórios de armazenamento de dados.
- Testes de sistema dos controladores: Testes da funcionalidade geral do sistema e integração de componentes em um nível mais alto.

Teste Unitário (Entidade Aluno)

Objetivo do Teste:	Verificar se a entidade Aluno da camada de domínio possui métodos de igualdade e comparação natural.
Técnica:	Implementação de testes unitários para os métodos de igualdade (__eq__) e comparação natural (__lt__, __gt__, etc.) da classe Aluno.
Critério de Finalização:	Todos os métodos de igualdade e comparação natural da classe Aluno foram testados e validados.
Considerações Especiais:	Certificar-se de que os testes abrangem a classe Aluno, verificando se os métodos de igualdade e comparação natural estão implementados corretamente. Os métodos devem confirmar se uma instância de Aluno é igual, diferente ou se uma instância deve ser colocada antes ou depois de outra, respeitando a modelagem específica da classe Aluno. Isso garantirá um comportamento correto ao comparar e ordenar instâncias de Aluno no contexto do domínio do sistema.

Teste Unitário (Entidade Tarefa)

Objetivo do Teste:	Verificar se a entidade Tarefa da camada de domínio possui métodos de igualdade e comparação natural.
Técnica:	Implementação de testes unitários específicos para os métodos de igualdade e comparação natural da classe Tarefa.
Critério de Finalização:	Todos os métodos de igualdade e comparação natural da classe Tarefa foram testados e validados.

Considerações Especiais:	Garantir que os testes cubram os métodos de igualdade e comparação natural da entidade Tarefa, verificando se estão implementados corretamente e produzem os resultados esperados. Além disso, certificar-se de que outros métodos relevantes da classe Tarefa não são afetados pelos testes e estão funcionando conforme o esperado.

Teste Unitário (Entidade Cronograma)

Objetivo do Teste:	Verificar se a entidade Cronograma da camada de domínio possui métodos de igualdade e comparação natural.
Técnica:	Implementação de testes unitários específicos para os métodos de igualdade (__eq__) e comparação natural (__lt__, __gt__, etc.) da classe Cronograma.
Critério de Finalização:	Todos os métodos de igualdade e comparação natural da classe Cronograma foram testados e validados.
Considerações Especiais:	Certificar-se de que os testes abrangem todas as funcionalidades de igualdade e comparação natural da classe Cronograma. Isso inclui verificar se a implementação dos métodos __eq__, __lt__, __gt__, entre outros, está correta, refletindo adequadamente as regras de igualdade e ordenação para a entidade Cronograma.

Teste de Integração (Entidade Aluno)

Objetivo do Teste:	Verificar se a entidade Aluno está corretamente persistida e se as operações de criação, leitura, atualização e exclusão funcionam adequadamente.
Técnica:	Implementação de testes de integração específicos para interagir com o repositório de persistência da entidade Aluno e executar as operações de CRUD (Create, Read, Update, Delete).
Critério de Finalização:	As operações de persistência da entidade Aluno foram testadas e validadas, garantindo que os dados do Aluno são armazenados corretamente e que as operações CRUD funcionam conforme o esperado.
Considerações Especiais:	Certificar-se de que os testes abrangem todos os requisitos de persistência para a entidade Aluno, incluindo a correta criação, recuperação, atualização e exclusão dos dados. Além disso, é importante verificar se as atualizações nos atributos do Aluno são refletidas corretamente na camada de persistência. Os testes devem ser escritos considerando as práticas recomendadas para testes de integração, garantindo a consistência dos dados e a correta interação com o repositório de persistência.

Teste de Integração (Entidade Tarefa)

Objetivo do Teste:	Verificar se a entidade Tarefa está corretamente persistida e se as operações de criação, leitura, atualização e exclusão funcionam adequadamente.
Técnica:	Implementação de testes de integração específicos para interagir com o repositório de persistência da entidade Tarefa e executar as operações de CRUD (Create, Read, Update, Delete).
Critério de Finalização:	As operações de persistência da entidade Tarefa foram testadas e validadas, garantindo que os dados das Tarefas são armazenados corretamente e que as operações CRUD funcionam conforme o esperado.
Considerações Especiais:	Certificar-se de que os testes abrangem todos os requisitos de persistência para a entidade Tarefa, incluindo a correta criação, recuperação, atualização e exclusão dos dados. Além disso, é importante verificar se as atualizações nos atributos da Tarefa são refletidas corretamente na camada de persistência. Os testes devem ser escritos considerando as práticas recomendadas para testes de integração, garantindo a consistência dos dados e a correta interação com o repositório de persistência.

Teste de Integração (Entidade Cronograma)

Objetivo do Teste:	Verificar se a entidade Cronograma está corretamente persistida e se as operações de criação, leitura, atualização e exclusão funcionam adequadamente.
Técnica:	Implementação de testes de integração específicos para interagir com o repositório de persistência da entidade Cronograma e executar as operações de CRUD (Create, Read, Update, Delete).
Critério de Finalização:	As operações de persistência da entidade Cronograma foram testadas e validadas, garantindo que os dados dos Cronogramas são armazenados corretamente e que as operações CRUD funcionam conforme o esperado.
Considerações Especiais:	Certificar-se de que os testes abrangem todos os requisitos de persistência para a entidade Cronograma, incluindo a correta criação, recuperação, atualização e exclusão dos dados. Além disso, é importante verificar se as atualizações nos atributos do Cronograma são refletidas corretamente na camada de persistência. Os testes devem ser escritos considerando as práticas recomendadas para testes de integração, garantindo a consistência dos dados e a correta interação com o repositório de persistência.

Teste de Sistema (Aluno)

Objetivo do Teste:	Verificar se a entidade Aluno é corretamente cadastrada, recuperada, atualizada e removida por meio da API REST.
Técnica:	Implementação de testes de sistema que realizem interações com a API REST do backend, realizando operações de CRUD (Create, Read, Update, Delete) na entidade Aluno.
Critério de Finalização:	Todos os controladores da API relacionados à entidade Aluno foram testados e validados, garantindo que as operações de CRUD funcionam conforme o esperado.
Considerações Especiais:	Certificar-se de que os testes abrangem todas as operações fornecidas pela API relacionadas ao Aluno, incluindo o cadastro, recuperação, atualização e remoção da entidade. Além disso, verificar se as validações de entrada estão sendo aplicadas corretamente e se os erros são tratados de forma adequada. É importante considerar a autenticação do usuário, caso necessário, e garantir que a API esteja documentada corretamente para facilitar o uso por parte dos consumidores. Os testes devem realizar as solicitações HTTP relevantes para cada operação e verificar se as respostas da API estão corretas e conforme os requisitos esperados.

Teste de Sistema (Tarefa)

Objetivo do Teste:	Verificar se a entidade Tarefa é corretamente cadastrada, recuperada, atualizada e removida por meio da API REST.
Técnica:	Implementação de testes de sistema que realizem interações com a API REST do backend, realizando operações de CRUD (Create, Read, Update, Delete) na entidade Tarefa.
Critério de Finalização:	Todos os controladores da API relacionados à entidade Tarefa foram testados e validados, garantindo que as operações de CRUD funcionam conforme o esperado.
Considerações Especiais:	Certificar-se de que os testes abrangem todas as operações fornecidas pela API relacionadas à Tarefa, incluindo o cadastro, recuperação, atualização e remoção da entidade. Verificar se as validações de entrada estão sendo aplicadas corretamente e se os erros são tratados de forma adequada. É importante considerar a autenticação do usuário, caso necessário, e garantir que a API esteja documentada corretamente para facilitar o uso por parte dos consumidores. Os testes devem realizar as solicitações HTTP relevantes para cada operação e verificar se as respostas da API estão corretas e conforme os requisitos esperados.

Teste de Sistema (Cronograma)

Objetivo do Teste:	Verificar se a entidade Cronograma é corretamente cadastrada, recuperada, atualizada e removida por meio da API REST.
Técnica:	Implementação de testes de sistema que realizam interações com a API REST do backend, realizando operações de CRUD (Create, Read, Update, Delete) na entidade Cronograma.
Critério de Finalização:	Todos os controladores da API relacionados à entidade Cronograma foram testados e validados, garantindo que as operações de CRUD funcionam conforme o esperado.
Considerações Especiais:	Certificar-se de que os testes abrangem todas as operações fornecidas pela API relacionadas ao Cronograma, incluindo o cadastro, recuperação, atualização e remoção da entidade. Verificar se as validações de entrada estão sendo aplicadas corretamente e se os erros são tratados de forma adequada. É importante considerar a autenticação do usuário, caso necessário, e garantir que a API esteja documentada corretamente para facilitar o uso por parte dos consumidores. Os testes devem realizar as solicitações HTTP relevantes para cada operação e verificar se as respostas da API estão corretas e conforme os requisitos esperados.

Teste de Sistema (Autenticação)

Objetivo do Teste:	Verificar se as camadas de aplicação estão sendo respeitadas durante a implementação e se os controladores da API do backend estão corretamente implementados, acessíveis e documentados.
Técnica:	Implementação de testes de sistema que simulam interações com a API do backend, verificando a correta execução das operações de criação, leitura, atualização e exclusão das entidades de domínio, bem como a autenticação e o logout do usuário.
Critério de Finalização:	Todos os controladores da API foram testados e validados, garantindo o funcionamento adequado das operações de CRUD, autenticação e logout do usuário.
Considerações Especiais:	Certificar-se de que os testes abrangem todas as operações fornecidas pela API, incluindo a criação, recuperação, atualização e remoção das entidades de domínio, bem como a autenticação e o logout do usuário. Além disso, garantir que a documentação da API esteja atualizada e acessível.

Testes Estáticos (Pylint)

Objetivo do Teste:	Identificar e corrigir problemas relacionados a code smells e código duplicado no software.
Técnica:	Utilização do Pylint para análise estática do código-fonte, identificando code smells e código duplicado.
Critério de Finalização:	Todos os problemas identificados pelo Pylint relacionados a code smells e código duplicado foram revisados e corrigidos.
Considerações Especiais:	É importante revisar e tratar todos os problemas encontrados pelo Pylint, garantindo a qualidade do código-fonte. Além disso, é necessário considerar a configuração adequada do Pylint para atender aos padrões e diretrizes da equipe de desenvolvimento.

Testes Estáticos (Sonar Cloud)

Objetivo do Teste:	Utilizar o Sonar Cloud para identificar e resolver falhas e problemas encontrados durante a análise estática do código.
Técnica:	Integrar o projeto ao Sonar Cloud e analisar o código-fonte em busca de vulnerabilidades, bugs, dívidas técnicas, problemas de segurança, entre outros.
Critério de Finalização:	Todas as falhas e problemas encontrados pelo Sonar Cloud foram revisados e resolvidos conforme as melhores práticas.
Considerações Especiais:	É fundamental revisar e resolver todas as falhas e problemas identificados pelo Sonar Cloud, a fim de melhorar a qualidade e a segurança do software. Além disso, é importante considerar a configuração adequada do Sonar Cloud, definindo as regras e os critérios de análise que sejam relevantes para o projeto.