UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
CAMPUS PAU DOS FERROS
DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIA

JOSIMARA SILVA DE LIMA - 2021031245
PAULINA JÚLIA COSTA DE OLIVEIRA - 2023023644

SIMPLY +

PAU DOS FERROS - RN
2025

Introdução

Este documento apresenta a estrutura e as funcionalidades do Simply+, um sistema de gerenciamento de tarefas e hábitos desenvolvido utilizando Vue.js para a versão inicial. O sistema foi projetado para otimizar a organização de tarefas diárias e a criação de hábitos, proporcionando ferramentas que simplificam a gestão de atividades, melhoram a produtividade e garantem uma administração eficiente e personalizada para cada usuário.

1. Classe Tarefa

A classe Tarefa representa uma tarefa no sistema, permitindo a criação, busca, edição e exclusão de tarefas armazenadas no *localStorage* do navegador. Cada tarefa possui atributos como título, descrição, prazo, prioridade, status de conclusão, data de criação e usuário associado.

I. Campos:

- id (*Numbe*r): Identificador único da tarefa, gerado automaticamente caso não seja informado.
- titulo (*String*): Nome ou título da tarefa.
- descricao (*String*): Descrição detalhada da tarefa.
- prazo (*String*): Data limite para a conclusão da tarefa.
- prioridade (*String*): Indica o nível de prioridade da tarefa.
- concluida (*Boolean*): Status da tarefa, indicando se foi concluída ou não.
- data_criacao(*String*): Data e hora de criação da tarefa no sistema, gerada automaticamente caso não seja fornecida.
- user (*String*): ID do Usuário responsável pela tarefa.

II. Métodos:

1- buscarTodas()

- **Descrição**: Retorna todas as tarefas armazenadas no localStorage criadas pelo usuário logado.

- **Retorno**: Array de objetos representando as tarefas cadastradas.

2- buscarPorId(id)

- **Descrição**: Busca uma tarefa específica com base no seu identificador.
- **Parâmetros**:
    - id (*Number*): Identificador da tarefa a ser buscada.
- **Retorno**: Objeto da tarefa correspondente ou *undefined* caso não seja encontrada.

3- salvar(task)

- **Descrição**: Adiciona ou atualiza uma tarefa no localStorage.
- **Parâmetros**:
    - task (Objeto Tarefa): Objeto representando a tarefa a ser salva.
- **Retorno**: Nenhum.

4- excluir(id)

- **Descrição**: Remove uma tarefa do *localStorage* com base no seu identificador.
- **Parâmetros**:
    - id (*Numbe*r): Identificador da tarefa a ser removida.
- **Retorno**: *Array* atualizado de tarefas após a remoção.

5 - alterarStatus(task)

- **Descrição**: Atualiza o status de conclusão de uma tarefa e salva a alteração no *localStorage*.
- **Parâmetros**:
    - task (Objeto Tarefa): Objeto da tarefa com o status atualizado.
    - **Retorno**: Nenhum.

2. Classe Habito

A classe Habito representa um hábito no sistema, permitindo a criação, busca, edição e exclusão de hábitos armazenados no *localStorage* do navegador. Cada hábito possui atributos como título, descrição, frequência, prioridade, status de conclusão, data de criação e usuário associado.

I. Campos:

- id (*Number*): Identificador único do hábito, gerado automaticamente caso não seja informado.
- titulo (*String*): Nome ou título do hábito.
- descricao (*String*): Descrição detalhada do hábito.
- frequencia(*String*): Define a periodicidade do hábito (exemplo: "Diário", "Semanal", "Mensal").
- prioridade (*String*): Indica o nível de prioridade do hábito.
- concluida (*Boolean | Object*): Armazena o status de conclusão do hábito, podendo ser um objeto que registra os dias em que foi concluído.
- data_criacao (*String*): Data e hora de criação do hábito no sistema, gerada automaticamente caso não seja fornecida.
- user (*String*): Id do usuário responsável pelo hábito.

II. Métodos:

1-buscarTodos()

- **Descrição**: Retorna todos os hábitos armazenados no *localStorage* criados pelo usuário logado.
- **Retorno**: *Array* de objetos representando os hábitos cadastrados.

2-buscarPorId(id)

- **Descrição**: Busca um hábito específico com base no seu identificador.
- **Parâmetros**:
    - id (*Number*): Identificador do hábito a ser buscado.

- **Retorno**: Objeto do hábito correspondente ou *undefined* caso não seja encontrado.

3-salvar(habito)

- **Descrição**: Adiciona ou atualiza um hábito no *localStorage*. Caso o hábito seja atualizado, registra a conclusão do dia atual.
- **Parâmetros**:
    - habito(Objeto Habito): Objeto representando o hábito a ser salvo.
- **Retorno**: Nenhum.

3-excluir(id)

- **Descrição**: Remove um hábito do *localStorage* com base no seu identificador.
- **Parâmetros**:
    - id (*Number*): Identificador do hábito a ser removido.
- **Retorno**: *Array* atualizado de hábitos após a remoção.

4-alterarStatus(habito)

- **Descrição**: Atualiza o status de conclusão de um hábito e salva a alteração no *localStorage.*
- **Parâmetros**:
    - habito (Objeto Habito): Objeto do hábito com o status atualizado.
- **Retorno**: Nenhum.

3. Classe Usuario

A classe Usuario representa um usuário do sistema, permitindo sua criação, busca, edição e remoção. Além disso, a classe gerencia o login e logout dos usuários, armazenando os dados no *localStorage* do navegador.

I. Campos:

- id *(Number)*: Identificador único do usuário, gerado automaticamente caso não seja informado.

- nome (*String*): Nome do usuário.
- email (*String*): Endereço de e-mail do usuário.
- senha (*String*): Senha do usuário.
- logado *(Boolean*): Indica se o usuário está logado no sistema.
- data_criacao (*String*): Data e hora de criação do usuário no sistema, gerada automaticamente caso não seja fornecida.

II. Métodos:

1-buscarTodos()

- **Descrição**: Retorna todos os usuários armazenados no *localStorage*.
- **Retorno**: *Array* de objetos representando os usuários cadastrados.

2-buscarPorId(id)

- **Descrição**: Busca um usuário específico com base no seu identificador.
- **Parâmetros**:
  - id (*Number*): Identificador do usuário a ser buscado.
- **Retorno**: Objeto do usuário correspondente ou *undefined* caso não seja encontrado.

3-salvar(usuario)

- **Descrição**: Adiciona ou atualiza um usuário no *localStorage.*
- **Parâmetros**:
  - usuario (Objeto Usuario): Objeto representando o usuário a ser salvo.
- **Retorno**: Nenhum.

4-excluir(id)

- **Descrição**: Remove um usuário do *localStorage* com base no seu identificador.

- **Parâmetros**:
    - id *(Number):* Identificador do usuário a ser removido.

- **Retorno**: *Array* atualizado de usuários após a remoção.

5-login(email, senha)

- **Descrição**: Realiza a autenticação do usuário no sistema.

- **Parâmetros**:
    - email (*String*): Endereço de e-mail do usuário.
    - senha (*String*): Senha do usuário.

- **Retorno**: Objeto do usuário autenticado caso as credenciais sejam válidas, ou null caso contrário.

6-logout()

- **Descrição**: Realiza o logout do usuário, deslogando todos os usuários e removendo o token de autenticação.

- **Parâmetros**: Nenhum.

- **Retorno**: Nenhum.