# Documento de Requisitos - Sistema de Registro de Refeições

Versão: 1.1

Data: 2024-07-26

## 1. Introdução

### 1.1. Propósito

Este documento descreve os requisitos funcionais e não funcionais do Sistema de Registro de Refeições, projetado para gerenciar o controle de consumo de refeições (almoço e lanche) por alunos.

### 1.2. Escopo

O sistema abrange o cadastro e gerenciamento de sessões de refeição, a identificação e registro de consumo por alunos elegíveis, a sincronização de dados de alunos e reservas com fontes externas (Google Sheets), a exportação de dados de consumo e a interface gráfica para operação.

### 1.3. Definições, Acrônimos e Abreviações

* IFSP: Instituto Federal de Educação, Ciências e Tecnologia de São Paulo
* GUI: Interface Gráfica do Usuário (Graphical User Interface).
* CRUD: Create, Read, Update, Delete (Operações básicas de banco de dados).
* API: Interface de Programação de Aplicações (Application Programming Interface).
* OAuth2: Protocolo de autorização aberto.
* CSV: Comma-Separated Values (Valores Separados por Vírgula).
* JSON: JavaScript Object Notation.
* DB: Banco de Dados (Database).
* PK: Chave Primária (Primary Key).
* FK: Chave Estrangeira (Foreign Key).
* UI: Interface do Usuário (User Interface).
* Prontuário: Identificador único do aluno (contexto IFSP inferido, pode ser genérico "ID Aluno").
* Sessão: Um evento específico de serviço de refeição (ex: Almoço de 26/07/2024 às 11:30).
* Reserva: Registro indicando a intenção de um aluno consumir uma refeição específica.
* Consumo: Registro confirmando que um aluno consumiu uma refeição em uma sessão.
* Elegível: Aluno que atende aos critérios (turma, reserva se aplicável) para consumir na sessão atual e ainda não consumiu.
* Fuzzy Matching: Técnica de busca por similaridade textual.

## 

## 2. Requisitos Gerais

* RG-01: O sistema deve fornecer uma Interface Gráfica do Usuário (GUI) para interação do operador.
* RG-02: O sistema deve utilizar um banco de dados local (SQLite) para persistência dos dados principais (alunos, turmas, sessões, reservas, consumos).
* RG-03: O sistema deve realizar logging de eventos importantes, erros e atividades de sincronização em arquivos locais para fins de diagnóstico.
* RG-04: O sistema deve ser compatível primariamente com sistemas Windows e Linux, buscando a pasta "Documentos" do usuário de forma apropriada para cada plataforma.
* RG-05: Todos os textos visíveis na interface do usuário devem ser centralizados em um arquivo de constantes para facilitar a manutenção e futura tradução.
* RG-06: As datas devem ser exibidas na interface do usuário no formato DD/MM/YYYY. O formato YYYY-MM-DD pode ser usado internamente para armazenamento e APIs.

## 

## 3. Requisitos Funcionais

### FR-01: Gerenciamento de Sessão

* FR-01.1: O usuário deve ser capaz de iniciar uma nova sessão de refeição especificando: Data (DD/MM/YYYY), Hora (HH:MM), Tipo de Refeição (Almoço ou Lanche).
* FR-01.2: Se o tipo for "Lanche", o usuário deve poder especificar um nome para o lanche (ou usar um padrão), com opção de salvar novos nomes de lanche.
* FR-01.3: O usuário deve selecionar as turmas participantes da nova sessão.
* FR-01.4: O sistema deve validar as entradas da nova sessão (formatos, seleções obrigatórias).
* FR-01.5: Ao criar uma sessão de "Lanche" para uma data sem reservas de lanche existentes, o sistema deve tentar criar automaticamente reservas de lanche para todos os alunos cadastrados para essa data.
* FR-01.6: Ao criar uma sessão de "Almoço", o sistema deve verificar se existem reservas de almoço para a data; caso contrário, a criação da sessão deve falhar.
* FR-01.7: O usuário deve ser capaz de carregar uma sessão existente selecionando-a de uma lista de sessões anteriores (ordenadas por data/hora).
* FR-01.8: O sistema deve persistir o ID da sessão ativa entre execuções (em session.json).
* FR-01.9: O sistema deve exibir claramente as informações da sessão ativa (Tipo, Data, Hora, ID).

### FR-02: Registro de Consumo

* FR-02.1: O sistema deve permitir a busca de alunos elegíveis por nome ou prontuário/ID. A busca deve iniciar após o usuário digitar um mínimo de 2 caracteres, com um pequeno atraso (debounce).
* FR-02.2: A busca deve utilizar correspondência por similaridade (fuzzy matching) para encontrar alunos.
* FR-02.3: Os resultados da busca devem exibir Nome, Turma, Prontuário/ID (parcial/limpo) e Status/Prato da Reserva do aluno elegível.
* FR-02.4: O sistema deve permitir a seleção de um aluno na lista de resultados.
* FR-02.5: Ao selecionar um aluno, o sistema deve exibir seus detalhes (Prontuário/ID, Nome, Turma, Prato/Status).
* FR-02.6: O usuário deve ser capaz de registrar o consumo do aluno selecionado (via botão ou tecla Enter/Double-Click).
* FR-02.7: O sistema deve registrar a hora exata (HH:MM:SS) do consumo.
* FR-02.8: O sistema deve impedir o registro de consumo para um aluno que já consumiu na sessão atual, exibindo um aviso.
* FR-02.9: O sistema deve fornecer feedback visual claro sobre o sucesso ou falha do registro.

### FR-03: Visualização e Gerenciamento de Consumos

* FR-03.1: O sistema deve exibir uma tabela com os alunos que já tiveram seu consumo registrado na sessão atual.
* FR-03.2: A tabela de registrados deve incluir Prontuário/ID, Nome, Turma(s), Hora do Consumo e Prato/Status da Reserva.
* FR-03.3: A tabela de registrados deve ser ordenável por qualquer coluna (exceto ação).
* FR-03.4: O usuário deve ser capaz de remover um registro de consumo da sessão atual (clicando em um ícone de ação na linha ou usando a tecla Delete na linha selecionada).
* FR-03.5: O sistema deve solicitar confirmação antes de remover um registro de consumo.
* FR-03.6: O sistema deve exibir contadores atualizados do número de alunos registrados e restantes/elegíveis.

### FR-04: Filtragem de Alunos Elegíveis

* FR-04.1: O usuário deve poder abrir um diálogo para filtrar quais turmas são consideradas na lista de alunos elegíveis.
* FR-04.2: Para cada turma, o usuário deve poder escolher mostrar os alunos: Com Reserva, Sem Reserva.
* FR-04.3: O diálogo de filtro deve permitir selecionar/desmarcar todas as opções rapidamente.
* FR-04.4: A aplicação do filtro deve atualizar imediatamente a lista de alunos elegíveis (resultados da busca).
* FR-04.5: O filtro aplicado deve ser associado à sessão ativa no banco de dados.

### FR-05: Sincronização de Dados Mestre (Google Sheets -> DB)

* FR-05.1: O usuário deve poder iniciar uma sincronização para buscar dados atualizados de Alunos e Reservas das Planilhas Google configuradas.
* FR-05.2: A sincronização deve ocorrer em segundo plano (thread) exibindo uma barra de progresso.
* FR-05.3: O sistema deve baixar os dados da planilha de Alunos, salvá-los em CSV e importar/atualizar os alunos no banco de dados local.
* FR-05.4: O sistema deve baixar os dados da planilha de Reservas, salvá-los em CSV e importar/atualizar as reservas no banco de dados local.
* FR-05.5: Durante a importação de reservas, se um prontuário/ID exato não for encontrado no DB, o sistema deve tentar localizar o aluno por similaridade (fuzzy matching) de nome e prontuário/ID. Se um match acima de um limiar for encontrado, a reserva deve ser associada a esse aluno; caso contrário, a reserva deve ser ignorada com um aviso.
* FR-05.6: O sistema deve exibir feedback sobre o sucesso ou falha da sincronização.

### FR-06: Sincronização de Consumos Servidos (DB -> Google Sheets)

* FR-06.1: O usuário deve poder iniciar uma sincronização para enviar os dados dos alunos servidos na sessão atual para a Planilha Google configurada.
* FR-06.2: A sincronização deve ocorrer em segundo plano (thread) exibindo uma barra de progresso.
* FR-06.3: O sistema deve formatar os dados de consumo (Prontuário/ID, Data da Sessão, Nome, Turma Simplificada, Prato/Status, Hora Consumo) para envio.
* FR-06.4: O sistema deve adicionar apenas linhas únicas à planilha de destino (evitar duplicatas).
* FR-06.5: O sistema deve determinar a aba correta da planilha com base no tipo de refeição da sessão (ex: "Almoço", "Lanche").
* FR-06.6: O sistema deve exibir feedback sobre o sucesso ou falha da sincronização.

### FR-07: Exportação de Dados

* FR-07.1: O usuário deve poder exportar os dados de consumo da sessão atual para um arquivo Excel (.xlsx).
* FR-07.2: O arquivo Excel deve ser salvo na pasta "Documentos" do usuário.
* FR-07.3: O nome do arquivo deve ser gerado automaticamente com base no Tipo de Refeição, Data (YYYY-MM-DD) e Hora (HH.MM) da sessão.
* FR-07.4: O arquivo Excel deve conter as colunas: Matrícula (ou ID Aluno), Data, Nome, Turma, Refeição (Prato/Status), Hora (Consumo).
* FR-07.5: O cabeçalho do Excel deve ser formatado (negrito, centralizado, borda).
* FR-07.6: O usuário deve ser notificado sobre o sucesso e o local do arquivo exportado ou sobre eventuais falhas.
* FR-07.7: A função "Exportar e Encerrar" deve executar a exportação local (FR-07.1 a FR-07.6) e, em seguida, limpar o estado da sessão ativa (session.json) e fechar a aplicação.

## 

## 4. Requisitos Não Funcionais

### RNF-01: Usabilidade

* RNF-01.1: A interface deve ser intuitiva e apresentar as informações de forma clara.
* RNF-01.2: O sistema deve fornecer feedback visual imediato para ações do usuário (ex: registro, erros, status da sincronização).
* RNF-01.3: A busca por alunos deve ser rápida e responsiva (uso de debounce).
* RNF-01.4: Ações comuns (registrar, deletar) devem ter suporte a atalhos de teclado (Enter, Delete).
* RNF-01.5: O estilo visual deve ser consistente (uso de tema ttkbootstrap).

### RNF-02: Desempenho

* RNF-02.1: A interface do usuário deve permanecer responsiva durante operações potencialmente demoradas (busca, sincronização).
* RNF-02.2: As operações de banco de dados devem ser eficientes (uso de bulk operations para importação, índices).
* RNF-02.3: A busca por similaridade (fuzzy matching) durante a importação de reservas não deve impactar excessivamente o tempo total de sincronização.

### RNF-03: Confiabilidade

* RNF-03.1: O sistema deve tratar erros de forma graciosa, exibindo mensagens informativas ao usuário sem travar.
* RNF-03.2: O logging detalhado deve permitir o diagnóstico de problemas.
* RNF-03.3: O estado da sessão ativa deve ser persistente de forma confiável.
* RNF-03.4: O sistema deve garantir a integridade dos dados através de constraints no banco de dados (chaves únicas).

### RNF-04: Segurança

* RNF-04.1: A autenticação com a API Google deve seguir o fluxo seguro OAuth2.
* RNF-04.2: Tokens de acesso e refresh devem ser armazenados localmente de forma segura (a segurança real depende das permissões do sistema de arquivos onde token.json é salvo).
* RNF-04.3: IDs de aluno podem ser ofuscados em certas representações internas ou logs (via utils.to\_code). *Nota: Isso não é uma medida de segurança forte.*

### RNF-05: Manutenibilidade

* RNF-05.1: O código deve ser organizado em camadas (controle, modelo, visão).
* RNF-05.2: Textos da UI e constantes devem ser centralizados.
* RNF-05.3: O código deve conter docstrings e comentários (em pt-br) para explicar funcionalidades e partes complexas.

### RNF-06: Portabilidade

* RNF-06.1: O sistema deve funcionar em Windows e Linux.
* RNF-06.2: A localização de diretórios padrão (Documentos) deve ser tratada de forma compatível com o sistema operacional.

## 

## 5. Requisitos de Dados

* RD-01: O sistema deve armazenar informações sobre Alunos (ID, ID Externo/Prontuário, Nome).
* RD-02: O sistema deve armazenar informações sobre Grupos/Turmas (ID, Nome).
* RD-03: Deve haver uma relação Muitos-para-Muitos entre Alunos e Grupos.
* RD-04: O sistema deve armazenar informações sobre Reservas (ID, ID Aluno, Prato, Data, É Lanche?, Cancelada?).
* RD-05: O sistema deve armazenar informações sobre Sessões (ID, Tipo Refeição, Período, Data, Hora, Turmas Participantes \[JSON]).
* RD-06: O sistema deve armazenar informações sobre Consumos (ID, ID Aluno, ID Sessão, ID Reserva?, Hora Consumo, Consumiu Sem Reserva?).
* RD-07: Devem existir constraints de unicidade para evitar dados duplicados (ID Externo/Prontuário, Nome do Grupo, Reserva por Aluno/Data/Tipo, Sessão por Refeição/Data/Hora, Consumo por Aluno/Sessão).
* RD-08: O sistema deve ser capaz de importar dados de Alunos e Reservas de arquivos CSV com cabeçalhos específicos (mapeados via EXTERNAL\_KEY\_TRANSLATION).
* RD-09: O sistema deve gerar exportações em Excel com cabeçalho definido (EXPORT\_HEADER).
* RD-10: O sistema deve gerenciar arquivos de configuração JSON (credentials.json, token.json, session.json, spreadsheet.json, lanches.json).

## 

## 6. Requisitos de Interface Externa

* RIE-01: O sistema deve interagir com a API do Google Sheets para:
  + RIE-01.1: Ler dados das abas "Discentes" (ou equivalente) e "DB" (ou equivalente para reservas).
  + RIE-01.2: Adicionar (append) dados de consumo servido nas abas correspondentes ao tipo de refeição ("Lanche", "Almoco"), evitando duplicatas.
* RIE-02: O sistema deve utilizar o fluxo de autenticação Google OAuth2 para obter permissão do usuário para acessar as planilhas (escopos spreadsheets e drive).
* RIE-03: O sistema deve ser capaz de usar um refresh token para obter novos tokens de acesso sem exigir re-autenticação constante do usuário.

## 

## 7. Considerações Futuras

* CF-01: A gestão de nomes de lanches poderia ser mais robusta (ex: UI para adicionar/remover).
* CF-02: A estratégia lazy dos relacionamentos *SQLAlchemy* pode ser otimizada (joined, subquery) para casos de uso específicos para melhorar o desempenho.