1) Modelo de dados (contrato) — JSON e Tipos

1.1. Tipos (TypeScript) para contrato de API (sem `File`)

> Isso é o que você pode manter no front como “API contract”.

```
export type WizardFileId = string;
export type IssueSeverity = "error" | "warning";

export type EquipmentRowData = {
  plant_code: string;
  plant_name: string;
  latitude: number | string;
  longitude: number | string; // mantém nome da planilha
  group_code: string;
  group_name: string;
  subgroup_code: string;
  subgroup_name: string;
  equipment_code: string;
  equipment_name: string | number;
  equipment_description: string;
  equipment_type: string;
};

export type EquipmentField = keyof EquipmentRowData;

export interface ApiUploadedFileMeta {
  id: WizardFileId;
  name: string;
  size: number;
  type: string;
}

export interface ValidationRow<TData = EquipmentRowData> {
  rowId: string;      // id interno (gerado pelo backend ou
front)
  rowIndex: number;   // linha da planilha
  data: TData;        // dados da linha
}

export interface ValidationIssue<TField extends string =
EquipmentField> {
  id: string;
  fileId: WizardFileId;
  rowId: string;
  rowIndex: number;
  field: TField;
  message: string;
  severity: IssueSeverity;
  resolved: boolean;
}

export interface FileValidationResult {
  fileId: WizardFileId;
  isValid: boolean;
  rowsToFix: ValidationRow<EquipmentRowData>[];
  issues: ValidationIssue<EquipmentField>[];
}

/** Import por arquivo */
export interface FileImportResult {
  fileId: WizardFileId;
  success: boolean;
  message: string;
}

/** Barra única (geral) pode ser calculada no front; backend
pode opcionalmente retornar */
export interface ApiProgress {
  progress: number; // 0..100
  status: "running" | "succeeded" | "failed";
  error?: string;
}
```

---

2) Endpoints sugeridos (Java REST)

Você tem 2 opções de arquitetura:

Opção A (mais simples): validação e importação síncronas (sem job)

- Front faz request
- Backend processa e devolve resultado no mesmo response

2.1 Upload inicial (opcional)

Se você quiser o backend gerar `fileId` e armazenar o arquivo temporariamente:

POST `/api/import/equipments/files`
Content-Type: `multipart/form-data`
Campos:

- `files`: um ou mais arquivos (`.xlsx`, `.xls`)

Response 200 (JSON):

```
{
  "files": [
    {
      "id": "0f51a40c-6c1b-40d5-9bdb-5c4e0cb98c88",
      "name": "Equipments4_Balsas.xlsx",
      "size": 123456,
      "type":
"application/vnd.openxmlformats-officedocument.spreadsheetml.s
heet"
    }
  ]
}
```

> Se você preferir, o `fileId` pode ser gerado no front e enviado como metadata junto (mais abaixo).

2.2 Validar arquivos

POST `/api/import/equipments/validate`
Content-Type: `application/json`

Request (síncrono)

Aqui você manda os `fileId` que já foram enviados no upload (ou um token de sessão).

```json
{
  "files": [
    { "id": "0f51a40c-6c1b-40d5-9bdb-5c4e0cb98c88" }
  ]
}
```

Response 200

```json
{
  "results": [
    {
      "fileId": "0f51a40c-6c1b-40d5-9bdb-5c4e0cb98c88",
      "isValid": false,
      "rowsToFix": [
        {
          "rowId": "2b7f1aab-5dc0-4bb8-b3b0-c9ecf9f8d6af",
          "rowIndex": 2,
          "data": {
            "plant_code": "N-S-MABS",
            "plant_name": "Balsas",
            "latitude": "",
            "longitude": -46.0355,
            "group_code": "N-S-MABS-MG601",
            "group_name": "",
            "subgroup_code": "N-S-MABS-MG601-IB230",
            "subgroup_name": "Módulo IB230",
            "equipment_code": "BSSD6010",
            "equipment_name": "",
            "equipment_description": "Equipamento 1",
            "equipment_type": "DJ"
          }
        }
      ],
      "issues": [
        {
          "id": "1b8d3a62-1f0a-4a58-9a7b-7b6c8c3f2de1",
          "fileId": "0f51a40c-6c1b-40d5-9bdb-5c4e0cb98c88",
          "rowId": "2b7f1aab-5dc0-4bb8-b3b0-c9ecf9f8d6af",
          "rowIndex": 2,
          "field": "latitude",
          "message": "Latitude inválida ou vazia.",
          "severity": "error",
          "resolved": false
        },
        {
          "id": "ab55d7fa-0b22-4d73-ae65-9b3a90f4b3b1",
          "fileId": "0f51a40c-6c1b-40d5-9bdb-5c4e0cb98c88",
          "rowId": "2b7f1aab-5dc0-4bb8-b3b0-c9ecf9f8d6af",
          "rowIndex": 2,
          "field": "group_name",
          "message": "Campo 'group_name' está vazio.",
          "severity": "warning",
          "resolved": false
        }
      ]
    }
  ]
}
```

✅ Note que esse JSON bate com seu `mockApi.ts` (campos e severidades).

2.3 Revalidar com correções (frontend manda linhas corrigidas)

Quando o usuário edita a tabela no passo 2, você manda as linhas que foram corrigidas.

POST `/api/import/equipments/revalidate`
Content-Type: `application/json`

Request

```
{
  "fileId": "0f51a40c-6c1b-40d5-9bdb-5c4e0cb98c88",
  "rows": [
    {
      "rowId": "2b7f1aab-5dc0-4bb8-b3b0-c9ecf9f8d6af",
      "rowIndex": 2,
      "data": {
        "plant_code": "N-S-MABS",
        "plant_name": "Balsas",
        "latitude": -7.5325,
        "longitude": -46.0355,
        "group_code": "N-S-MABS-MG601",
        "group_name": "Grupo MG601",
        "subgroup_code": "N-S-MABS-MG601-IB230",
        "subgroup_name": "Módulo IB230",
        "equipment_code": "BSSD6010",
        "equipment_name": 10001340,
        "equipment_description": "Equipamento 1",
        "equipment_type": "DJ"
      }
    }
  ]
}
```

Response 200 (válido)

```
{
  "fileId": "0f51a40c-6c1b-40d5-9bdb-5c4e0cb98c88",
  "isValid": true,
  "rowsToFix": [],
  "issues": []
}
```

2.4 Importar dados (Passo 3)

POST `/api/import/equipments/import`
Content-Type: `application/json`

Request

```
{
  "files": [
    { "fileId": "0f51a40c-6c1b-40d5-9bdb-5c4e0cb98c88" }
  ]
}
```

Response 200

```
{
  "results": [
    {
      "fileId": "0f51a40c-6c1b-40d5-9bdb-5c4e0cb98c88",
      "success": true,
      "message": "Importado com sucesso."
    }
  ]
}
```

Response 200 (falha por arquivo)

```
{
  "results": [
    {
      "fileId": "0f51a40c-6c1b-40d5-9bdb-5c4e0cb98c88",
      "success": false,
      "message": "Falha ao importar: erro na persistência do
equipamento."
    }
  ]
}
```

---

3) DTOs Java sugeridos

3.1 `EquipmentRowDataDTO`

```
public class EquipmentRowDataDTO {
  public String plant_code;
  public String plant_name;
  public Object latitude;   // pode ser String ou Number vindo
do front
  public Object longitude;

  public String group_code;
  public String group_name;

  public String subgroup_code;
  public String subgroup_name;

  public String equipment_code;
  public Object equipment_name; // pode ser String ou Number
  public String equipment_description;
  public String equipment_type;
}
```

3.2 `ValidationRowDTO`

```
public class ValidationRowDTO {
  public String rowId;
  public int rowIndex;
  public EquipmentRowDataDTO data;
}
```

3.3 `ValidationIssueDTO`

```
public class ValidationIssueDTO {
  public String id;
  public String fileId;
  public String rowId;
  public int rowIndex;
  public String field;      // deve ser um dos campos de
EquipmentRowDataDTO
  public String message;
  public String severity;   // "error" | "warning"
  public boolean resolved;
}
```

3.4 `FileValidationResultDTO`

```
import java.util.List;

public class FileValidationResultDTO {
  public String fileId;
  public boolean isValid;
  public List<ValidationRowDTO> rowsToFix;
  public List<ValidationIssueDTO> issues;
}
```

3.5 Requests/Responses

Validate request

```
import java.util.List;

public class ValidateRequestDTO {
  public List<FileRefDTO> files;

  public static class FileRefDTO {
    public String id;
  }
}
```

Validate response

```
import java.util.List;

public class ValidateResponseDTO {
  public List<FileValidationResultDTO> results;
}
```

Revalidate request

```
import java.util.List;

public class RevalidateRequestDTO {
  public String fileId;
  public List<ValidationRowDTO> rows;
}
```

Import request

```
import java.util.List;

public class ImportRequestDTO {
  public List<FileImportRefDTO> files;

  public static class FileImportRefDTO {
    public String fileId;
  }
}
```

Import result

```
public class FileImportResultDTO {
  public String fileId;
  public boolean success;
  public String message;
}
```

Import response

```
import java.util.List;

public class ImportResponseDTO {
  public List<FileImportResultDTO> results;
}
```

---

4) Observações importantes pro backend implementar

1. `rowIndex`: deve ser a linha real do Excel (ex.: começa em 2 por causa do header).

2. `rowId`: pode ser gerado pelo backend ao ler a planilha (UUID) e retornar. O front reaproveita esse `rowId` pra editar/revalidar.

3. `field`: deve ser exatamente um dos nomes:

    - `plant_code`, `plant_name`, `latitude`, `longitude`, `group_code`, `group_name`, `subgroup_code`, `subgroup_name`, `equipment_code`, `equipment_name`, `equipment_description`, `equipment_type`

4. `resolved`: o backend pode devolver `false` inicialmente; após revalidate ele recalcula.

---

5) JSON "único"

```
{
  "endpoints": [
    {
      "method": "POST",
      "path": "/api/import/equipments/files",
      "contentType": "multipart/form-data",
      "request": { "files": "File[]" },
      "response": {
        "files": [
          { "id": "string", "name": "string", "size": 0,
"type": "string" }
        ]
      }
    },
    {
      "method": "POST",
      "path": "/api/import/equipments/validate",
      "contentType": "application/json",
      "request": { "files": [ { "id": "string" } ] },
      "response": { "results": [ "FileValidationResult" ] }
    },
    {
      "method": "POST",
      "path": "/api/import/equipments/revalidate",
      "contentType": "application/json",
      "request": { "fileId": "string", "rows": [
"ValidationRow<EquipmentRowData>" ] },
      "response": "FileValidationResult"
    },
    {
      "method": "POST",
      "path": "/api/import/equipments/import",
      "contentType": "application/json",
      "request": { "files": [ { "fileId": "string" } ] },
      "response": { "results": [ "FileImportResult" ] }
    }
  ]
}
```