UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

TIAGO SILVEIRA CAMILO

Desenvolvimento de Sistema Web para Gerenciamento de Salas de Integração e Recursos

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Leandro Krug Wives
Coorientador: Me. Francisco Dutra dos Santos Jr.

Porto Alegre
2019

AGRADECIMENTOS

Agradeço à minha companheira Caroline Mundel que viveu comigo essa etapa importante, agradeço também à minha família por todo o suporte, pois sem eles nada seria possível. Agradeço a todos os professores que compartilharam seu tempo e conhecimento, nos permitindo evoluir como pessoas e profissionais, em especial aos professores Leandro e Franscisco que foram vitais no desenvolvimento deste trabalho, bem como à professora Sibele que aceitou participar do processo de avaliação.

RESUMO

Este trabalho tem como objetivo desenvolver uma solução de software para apoiar educadores que atuam em Salas de Integração e Recursos (Serviço de Atendimento Educacional Especializado - AEE) no atendimento de alunos com necessidade educacionais especiais. Nas rotinas de trabalho, esses educadores necessitam gerenciar as informações a respeito do desenvolvimento do aluno, elaborar pareceres, preencher formulários pré-estabelecidos. A solução propõe o desenvolvimento de um sistema web, responsivo, pelo qual o professor poderá preencher formulários, fazer a inclusão de imagens e vídeos na base de dados do aluno, importar informações para elaboração de pareceres e armazenar o histórico do aluno em uma linha do tempo. Além disso, permite que sejam criados formulários modelos, por meio da própria solução, visando atender as necessidades específicas de cada escola, além de permitir fácil evolução, garantindo a longevidade da solução. Por se tratar de uma solução web, foi escolhida a linguagem de programação PHP junto com o *framework* Symfony, e na camada de persistência de dados foi utilizado o sistema de gerenciamento de banco de dados PostgreSQL. O desenvolvimento do trabalho teve como resultado uma solução que atende às necessidades primordiais de professores especializados que atuam nas salas de recursos, a qual foi submetida a testes em um ambiente real, com um resultado satisfatório.

Palavras-chave: Sistema Web. Salas de Integração e Recursos. Symfony. Serviço de Atendimento Educacional Especializado. PHP

Web System Development for Management of Integration of Resource Rooms

ABSTRACT

This work aims to develop a software solution to support educators working in Integration of Resource Rooms for the Specialized Educational Attendance Service in the care of students with special educational needs. In their working routines, these educators need to manage information about student development, prepare opinions, and fill out pre-established forms. The solution here presented proposes the development of a responsive Web system whereby the teacher can fill out forms, include images and videos in the student database, import information for feedback and store student history on a timeline. In addition, it allows template forms to be created to meet the specific needs of each school, and allows for easy evolution, ensuring the longevity of the solution. Because it is a web solution, the PHP programming language was chosen along with the Symfony framework, and PostgreSQL database management system was used as the persistent layer. The development of the work has resulted in a solution that meets the overriding needs of specialist teachers working in resource rooms, which has been tested in a real environment, with a satisfactory result.

Keywords: Web System. Integration Rooms and Features. Symfony. Specialized Educational Attendance Service. PHP

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AEE	Atendimento Educacional Especializado
CRUD	Create, Read, Update and Delete
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
MVC	Model-View-Controller
NEE	Necessidades Educacionais Especiais
ORM	Object Relational Mapping
PDI	Plano de Desenvolvimento Individual
SASS	Syntactically Awesome Style Sheets
SIR	Salas de Integração e Recursos
SUS	System Usability Scale

SUMÁRIO

1 INTRODUÇÃO

A Inclusão Escolar de alunos com Necessidades Educacionais Especiais - NEEs, no sistema educacional brasileiro é decorrente de amplo debate social e operada a partir de marcos legais que garantem os direitos de acesso às escolas comuns em todo território nacional. Nesse sentido, foi necessário implementar políticas educacionais que viabilizassem esses direitos constitucionais. Dentre elas, o serviço de Atendimento Educacional Especializado - AEE, regulamentado pelo Decreto Nº 7.611 de 17 novembro de 2011, que prevê entre seus objetivos o fomento ao desenvolvimento de recursos didáticos e pedagógicos que eliminem as barreiras no processo de ensino e aprendizagem, a fim de assegurar condições para a continuidade de estudos nos demais níveis, etapas e modalidades de ensino (BRASIL, 2011).

Porém, este serviço configura-se em diferentes estruturas nas rede públicas de ensino. No presente estudo será adotada a proposta de rede municipal de ensino de Porto Alegre, que denomina o AEE como Sala de Integração e Recursos - SIR. Sendo assim, o aluno com necessidades educacionais especiais precisa de acompanhamento personalizado e periódico para que possa desenvolver, ao máximo, suas capacidades. As Salas de Integração e Recursos constituem-se em um espaço e em uma modalidade de trabalho pedagógico especialmente planejados para investigação e atendimento aos educandos do ensino regular que apresentam dificuldades além das habituais, por apresentarem necessidades educativas especiais, e que precisam de um trabalho pedagógico complementar específico que venha a contribuir para sua adequada integração nesta modalidade de ensino (ROCHA, 1997).

Nessas salas atuam profissionais com formação em educação especial que, por meio de atividades adequadas às necessidades de cada aluno, ajudam no seu desenvolvimento de maneira complementar à sala de aula regular. O educador especial tem um importante papel neste projeto, ser um interlocutor entre os demais educadores (SANTOS Jr., 2003).

O atendimento nas salas de recursos exige que os educadores executem or diversas atividades. Mundel (2019), ao modelar o processo de atendimento nas salas de recursos, listou, dentre outras, as que seguem:

- registrar os atendimentos com fotos ou vídeos, evidenciando avanços do aluno;
- registrar o acompanhamento de cada aula, com comentários a respeito do trabalho realizado e dos avanços identificados;

- elaborar pareceres que resumem a evolução e destacam fatos importantes no desenvolvimento do aluno em um período de tempo;
- elaborar o Plano de Desenvolvimento Individualizado (PDI), que contempla o perfil do aluno e define as áreas que serão desenvolvidas durante o atendimento na SIR no decorrer do ano;
- avaliar o plano de Adequação Curricular, que é uma proposta de como o professor referência de sala de aula deve abordar cada conteúdo de acordo com as necessidades do aluno.

Conforme mencionado, os profissionais que atuam nas SIR apresentam um perfil altamente qualificado e o processo de atualização do serviço de atendimento educacional especializado está em evolução constante, já tendo atingido um significativo nível de qualidade. Porém, apesar desses fatores positivos, o trabalho desses educadores ainda apresenta limitações ao uso da tecnologia, visto que não existe uma solução de software sendo utilizada em larga escala.

Profissionais, por iniciativa própria, utilizam a tecnologia para algumas das atividades, mas sempre por conta própria e, por consequência, de forma descentralizada. O resultado dessa ausência de uma solução difundida é que, em uma eventual mudança do educador, o acesso ao histórico do aluno passa a ser uma atividade relativamente complexa, visto que toda a documentação é realizada de forma manual. Essa questão é agravada quando, por algum motivo, o aluno troca de escola e, por vezes, informações importantes do seu desenvolvimento podem não estar contempladas na íntegra.

Vale destacar, também, que o modelo atual, onde as informações têm que ser pesquisadas e replicadas manualmente, pode apresentar forte risco de descontinuidade na elaboração dos pareceres.

Tendo em vista esse cenário, a solução que será desenvolvida neste trabalho visa permitir a execução das atividades do profissional das Sala de Integração e Recursos. Algumas destas atividades serão desenvolvidas para execução na própria sala de recursos, através de smartphones como, por exemplo, a captura de fotos/vídeos e o registro dos atendimentos, visando maior aderência ao processo atualmente executado. As demais atividades, que passam pela elaboração de pareceres ou formulários de maior complexidade, podem ser desenvolvidas através do computador.

No processo de desenvolvimento, o professor Francisco Dutra dos Santos Júnior, que atuou no Atendimento Educacional Especializado das Salas de Integração e Recursos do Município de Porto Alegre desde sua implantação, assumiu o papel de stakeholder[1].

Ressalta-se que já foram desenvolvidos protótipos em trabalhos anteriores com a finalidade de atender às necessidades dos educadores que atuam nas salas de recursos. Entretanto, os Sistemas SIR-EDU (Ferreira, 2017) e Sistema de Gestão e Acompanhamento Educacional (Lucas, 2018), não tiveram seguimento após o período de testes com usuários.

Como diferencial, esta solução também atenderá à necessidade de que os formulários de "Plano de Desenvolvimento Individual" e "Adequação Curricular" possam ser personalizados para cada escola. Assim como, através do próprio sistema, será possível a criação de novos formulários para atender eventuais melhorias no processo.

Por ser uma solução para arquitetura Web, não demandará processos de instalação nem recursos sofisticados de hardware.

O modelo de dados adotado, que será abordado em detalhes no decorrer do trabalho, tem como principais entidades: escolas, educadores e alunos. Com isso, cada escola será independente, podendo ter tantos alunos e professores quanto necessário. Adotando esse modelo, viabilizamos que em uma eventual troca de escola, o aluno possa ser transferido com todo seu histórico.

O documento está organizado da seguinte forma: no Capítulo seguinte apresenta-se uma visão geral das tecnologias e arquitetura utilizadas; em seguida, o Capítulo 3 apresenta a modelagem da solução, detalhando as histórias de usuários e a estrutura do banco de dados; o capítulo 4 aborda o processo de desenvolvimento do projeto; já o Capítulo 5 descreve as funcionalidades do sistema; no Capítulo 6 é apresentada sua validação; e, finalmente, no Capítulo 7 são descritas as conclusões deste trabalho, suas limitações e trabalhos futuros.

[1] No período deste estudo o professor Francisco Dutra dos Santos Jr. já desenvolvia pesquisa sobre a temática no Programa de Pós graduação em Informática na Educação - PPGIE - UFRGS, nível doutorado. O papel de stakeholder, é devido a sua expertise na área, articulada com o interesse ao desenvolvimento de tecnologia neste campo.

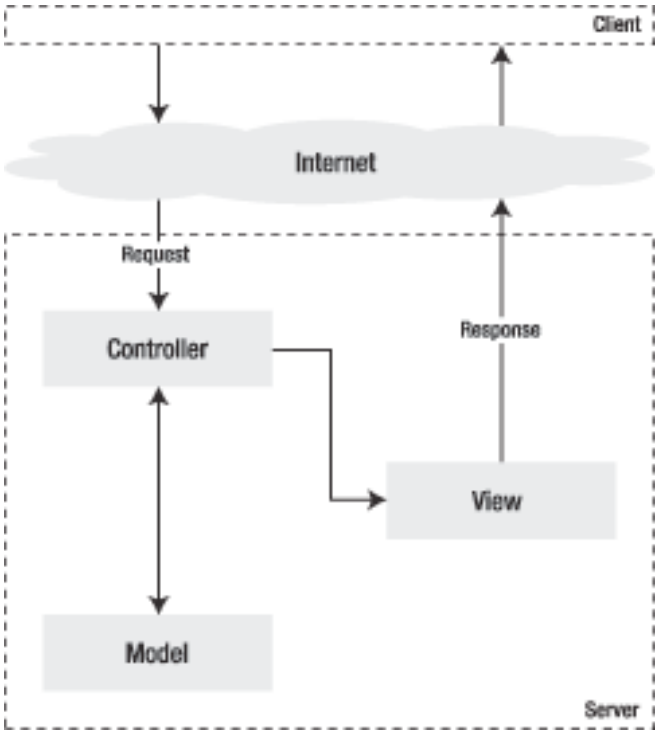
2 TECNOLOGIAS E ARQUITETURA UTILIZADAS

Este capítulo descreve as principais características, do ponto de vista de tecnologias e arquitetura utilizadas para desenvolvimento do trabalho.

A solução foi desenvolvida para plataforma Web, sendo acessada através de um navegador, caracterizando assim um modelo de uso cliente/servidor.

O projeto foi desenvolvido utilizando a linguagem de programação PHP, na versão 7.3, em conjunto com o *framework* Symfony[2], versão 4.3. O *framework*, por sua vez, utiliza a arquitetura MVC (*Model-View-Controller*), ilustrada pela Figura 2.1. Vale destacar que o *framework* Symfony provê internamente todos os recursos necessários para a camada de *Controller*, porém utiliza ferramentas especializadas nas camadas de *Model* (Doctrine[3]) e *View* (Twig[4]).

Figura 2.1 – Arquitetura MVC



Fonte: https://symfony.com/

A escolha das tecnologias foi embasada por experiências prévias e bem sucedidas, tanto com a linguagem PHP quanto com o *framework*.

[2] https://symfony.com/
[3] https://www.doctrine-project.org/
[4] https://twig.symfony.com/

Em diversas pesquisas sobre utilização de *frameworks* para PHP, o *Symfony* está entre os três melhores posicionados, normalmente atrás de Laravel, que é atualmente o mais difundido.

Vale ressaltar, também, que para escolha do *framework* foram considerados seus utilitários por linha de comando, na geração de *entities* e *CRUDs*, seu serviço nativo de injeção de dependências e o uso do *Doctrine* como ORM, pois esta é uma ferramenta consolidada e bem documentada.

2.1 Front-end

O *front-end* é a parte do sistema com a qual o usuário efetivamente realiza interações. Por ser uma solução baseada na arquitetura MVC, o *front-end* é responsabilidade exclusiva da camada de *View.*

Foram utilizadas tecnologias comuns a qualquer solução Web, como HTML, CSS e Javascript. Além dessas, foram utilizadas tecnologias que facilitam a manutenibilidade das anteriormente citadas, sendo elas Twig, SASS e Jquery.

Ressalta-se que na camada de *View* do *framework* Symfony todos os arquivos são chamados de *templates.* Um *template* é a maneira sugerida de organizar e renderizar o HTML de dentro do projeto. No Symfony os *templates* são criados com o Twig: um mecanismo de modelo flexível, rápido e seguro (SYMFONY TEMPLATE, 2019).

2.1.1 HTML

Hypertext Markup Language (HTML) é o bloco de construção mais básico da web. Define o significado e a estrutura do conteúdo da web (MOZILLA HTML, 2019).

No projeto foi utilizado Twig que é um "*template engine*" para linguagem PHP, que provê velocidade, segurança e flexibilidade na construção dos *templates* (TWIG, 2019).

A utilização do Twig evita o uso de código PHP puro interpolado com o HTML, facilitando assim a leitura e manutenção. Além disso, o Twig fornece construções de blocos que podem ser reutilizados e/ou sobrescritos, permitindo a criação de componentes dos elementos.

2.1.2 CSS

CSS (Cascading Style Sheets) é uma linguagem de estilo usada para descrever a apresentação de um documento escrito em HTML ou em XML (MOZILLA CSS, 2019).

Além do CSS puro, foi utilizado SASS (*Syntatically Awesome Style Sheets*) que é uma tecnologia que realiza um pré-processamento do CSS (SASS, 2019).

2.1.3 JavaScript

JavaScript é uma linguagem leve, interpretada e baseada em objetos com funções de primeira classe, mais conhecida como a linguagem de script para páginas Web, mas usada também em vários outros ambientes sem browser, tais como node.js, Apache CouchDB e Adobe Acrobat. O JavaScript é uma linguagem baseada em protótipos, multi-paradigma e dinâmica, suportando estilos de orientação a objetos, imperativos e declarativos (MOZILLA JavaScript, 2019).

Junto ao Javascript nativo foi utilizado JQuery que é uma biblioteca de Javascript que visa facilitar manutenções através de funções de manipulação de elementos do documento, animações e controles de eventos. A biblioteca garante, também, compatibilidade com os principais navegadores do mercado (JQuery, 2019).

Ambas tecnologias foram utilizadas para eventuais validações "*client-side*" e animações/manipulação de elementos do HTML.

2.2 Back-end

Back-end é o nome dado à parte do sistema que é executada exclusivamente no servidor. Dentre suas responsabilidades está o tratamento das requisições e respostas que são, por sua vez, a forma como *o front-end* e *back-end* se comunicam. Além disso, o *back-end* é responsável pela persistência de dados e validações de regras de negócio.

No *framework* utilizado, consideramos as camadas *Model* e o *Controller* do MVC como *back-end*. Destacando, no entanto, que a camada *View* foi tratada na parte de *front-end* porque produz o resultado para o usuário, mas sua "execução" também é realizada no servidor e apenas o resultado final, em HTML, é retornado ao navegador.

2.2.1 Controller

Uma *Controller* é uma classe PHP criada com o objetivo de tratar informações do objeto *Request* e criar e retornar um objeto *Response*. O *Response* pode ser uma página HTML, JSON, XML, um download de arquivo, um redirecionamento, uma mensagem de erro para o navegador (SYMFONY CONTROLLER, 2019).

Além da classe de controller propriamente dita, essa camada possui mais dois recursos importantes, abordados a seguir: route e forms.

2.2.1.1 Route

Quando o sistema recebe uma solicitação, ele executa uma ação de uma *Controller* para gerar uma resposta. A configuração de *Route* define qual ação deverá ser executada para cada requisição recebida (SYMFONY ROUTE, 2019).

No projeto foi decidido realizar a configuração de *Route* diretamente nas classes de *Controller*, através do recurso de *annotation* fornecido pelo *framework*, embora o mesmo também suporte a opção de haver um arquivo exclusivamente para configuração de *Route*.

2.2.1.2 Forms

São as classes utilizadas para a entrada de dados na aplicação. Nelas são definidas características como obrigatoriedade dos dados para aquela requisição. Vale destacar que, ao contrário das classes de *entity* (que serão abordadas posteriormente), a classe form representa as regras naquele ponto, logo, podemos ter classes *forms* com regras diferentes para uma mesma entidade.

O *framework* utilizado possui diversos recursos para esse tipo de classe, permitindo que sejam definidos tipo dos campos e características como tamanho e obrigatoriedade. Além disso, provê proteção aos ataques mais comuns, como submissão de campos que não estavam previstos e tratamento de inserção de caracteres de controle.

A classe *form* é utilizada pelo *framework* em dois momentos: primeiro para renderizar um formulário HTML propriamente dito, segundo para validar os dados recebidos na submissão deste formulário.

2.2.2 Model

A camada de *Model* é responsável pela persistência de dados assim como validações relacionadas ao domínio do negócio da solução. Tendo em vista sua grande responsabilidade, o *framework* sugere sua separação em subcamadas, sendo as principais abordadas a seguir.

2.2.2.1 Entity

São as classes de mapeamento das entidades da aplicação propriamente ditas.

O *framework* Symfony sugere a utilização do *Doctrine* como *Object Relational Mapping* (ORM), logo, as classes *entities* possuem todas as *annotation* necessárias para que a ferramenta possa gerar a base de dados baseada apenas nas classes. O contrário também seria possível, ou seja, gerar as classes de uma base de dados já existente.

Para cada tabela do banco de dados é gerada uma classe de *entity*, e para cada coluna da tabela um atributo nesta classe. Através de *annotation* definimos suas características como tipo, obrigatoriedade e tamanho. O doctrine suporta também atributos relacionais, permitindo representar na classe *entity* os relacionamentos entre as tabelas.

2.2.2.2 Repository

São classes responsáveis por realizar operações não-triviais sobre a base de dados, as quais são desenvolvidas visando melhor escalabilidade e performance.

Seguindo o padrão do *framework*, para cada *entity* deve haver um *repository*.

Cabe salientar que a ferramenta de ORM já oferece operações triviais, tanto de escrita, através de comandos para persistência, quanto de leitura, por meio de comandos para obter todos registros ou filtrar um conjunto de registros através de determinado critério.

2.2.2.3 Service

Sempre que uma operação demanda algum tratamento diferenciado e/ou complexo, esse comportamento específico deve ser implementado em uma classe de *service*. Essas,

por sua vez, ao contrário das classes de *entity* e *repository*, podem possuir de forma simultânea referências a diversas entidades da aplicação, visando executar regras de negócio mais elaboradas.

3 MODELAGEM DA SOLUÇÃO

Neste capítulo serão apresentadas as histórias de usuários, elaboradas através dos encontros com o professor Francisco Dutra dos Santos Jr., além da modelagem de dados gerada para atender as necessidades elencadas.

Durante o processo de análise foram identificados três atores principais para a solução, sendo eles:

- Administrador: o único ator que não possui um vínculo com uma escola em específico, tendo como principal responsabilidade o cadastro das escolas e seus respectivos coordenadores;
- Coordenador: tem a responsabilidade de cadastrar os usuários para os professores da SIR, assim como gerenciar a criação dos formulários modelos;
- Educador: são os professores da SIR propriamente ditos.

3.1 Histórias de usuário

Histórias de usuário são descrições curtas de funcionalidade contadas a partir da perspectiva de um usuário (Wiegers, 2013).

A sintaxe de uma história de usuário sugerida por Cohn (2004) deve conter:

- Como um... (papel ou ator) (Quem)
- Eu quero... (capacidade ou funcionalidades necessárias) (O que)
- de modo que... (por que é de valor do negócio ou benefício) (Por que)

Inicialmente, para o desenvolvimento deste trabalho, foram identificadas as seguintes histórias de usuário:

- Como um professor da SIR eu quero poder cadastrar um aluno para poder iniciar seu acompanhamento;
- Como um professor da SIR eu quero fotografar/gravar a evolução de um aluno em um encontro e salvar na sua linha do tempo para, futuramente, adicionar a pareceres;
- Como um professor da SIR eu quero fazer anotações referentes aos encontros com os alunos, para ter registros de acompanhamento;

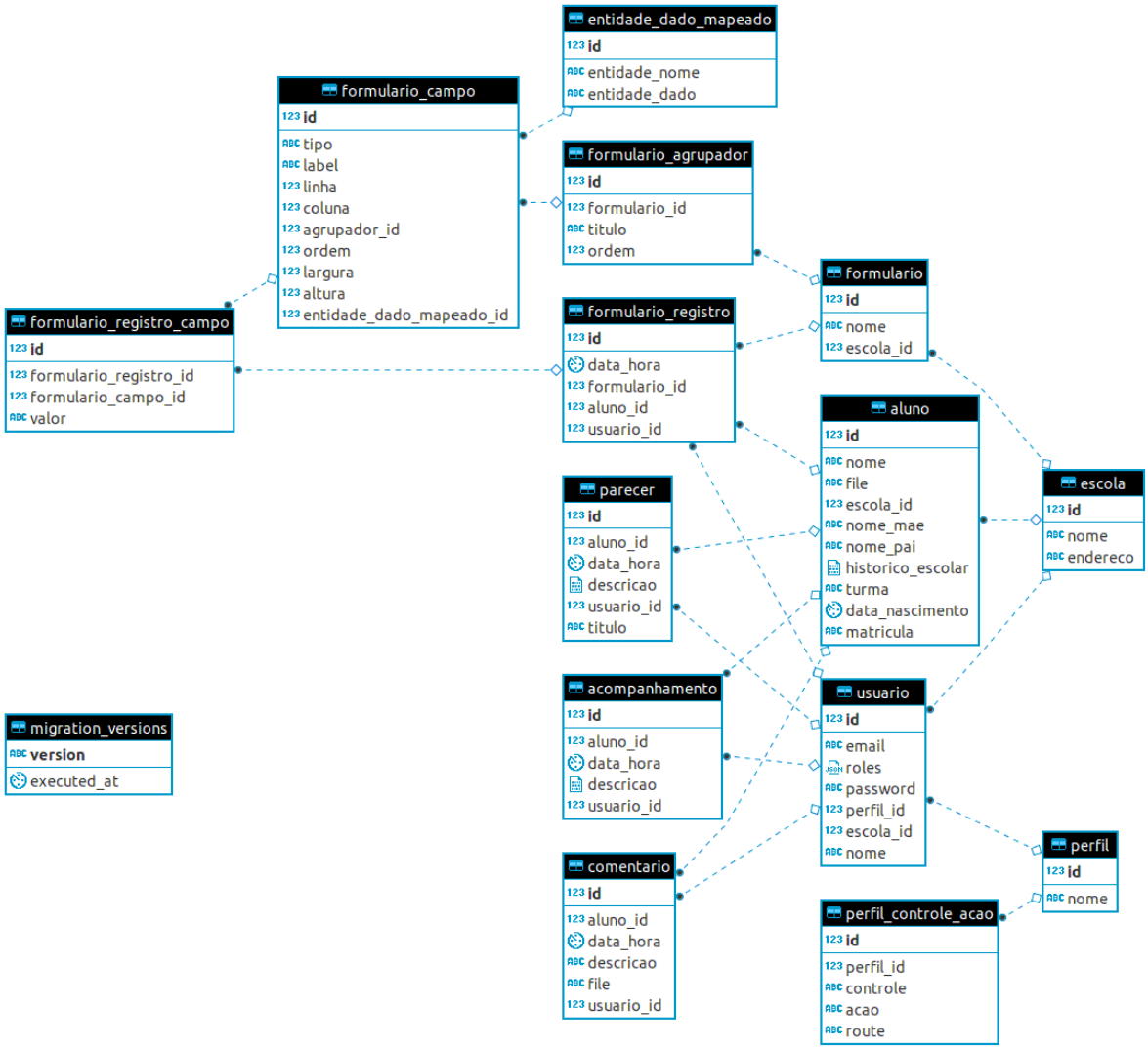
- Como um professor da SIR eu quero elaborar pareceres para destacar os pontos mais importantes do avanço dos alunos;
- Como um professor da SIR eu quero visualizar o histórico do aluno em uma linha do tempo;
- Como um professor da SIR eu quero poder importar o conteúdo de elementos da linha do tempo para os pareceres;
- Como um professor da SIR eu quero elaborar um “Plano de desenvolvimento individual” do aluno para destacar todas suas características;
- Como um professor da SIR eu quero poder criar “Adequações curriculares” para adaptar o plano de ensino de cada disciplina;
- Como um administrador do sistema eu quero poder cadastrar escolas para que iniciem o uso do sistema;
- Como um coordenador de escola eu quero poder cadastrar os usuários para os professores da SIR;
- Como um coordenador de escola eu quero alterar campos dos formulários de Plano de Desenvolvimento e Adequação Curricular para serem aderentes à realidade da escola.

3.2 Banco de dados

No projeto foi utilizado o sistema de gerenciamento de banco de dados PostgreSQL, que é um banco de dados relacional de código aberto (POSTGRESQL, 2019).

O diagrama atual da solução será apresentado na Figura 3.1.

Figura 3.1 – Diagrama base de dados



Fonte: Elaborada pelo autor

3.2.1 Tabelas

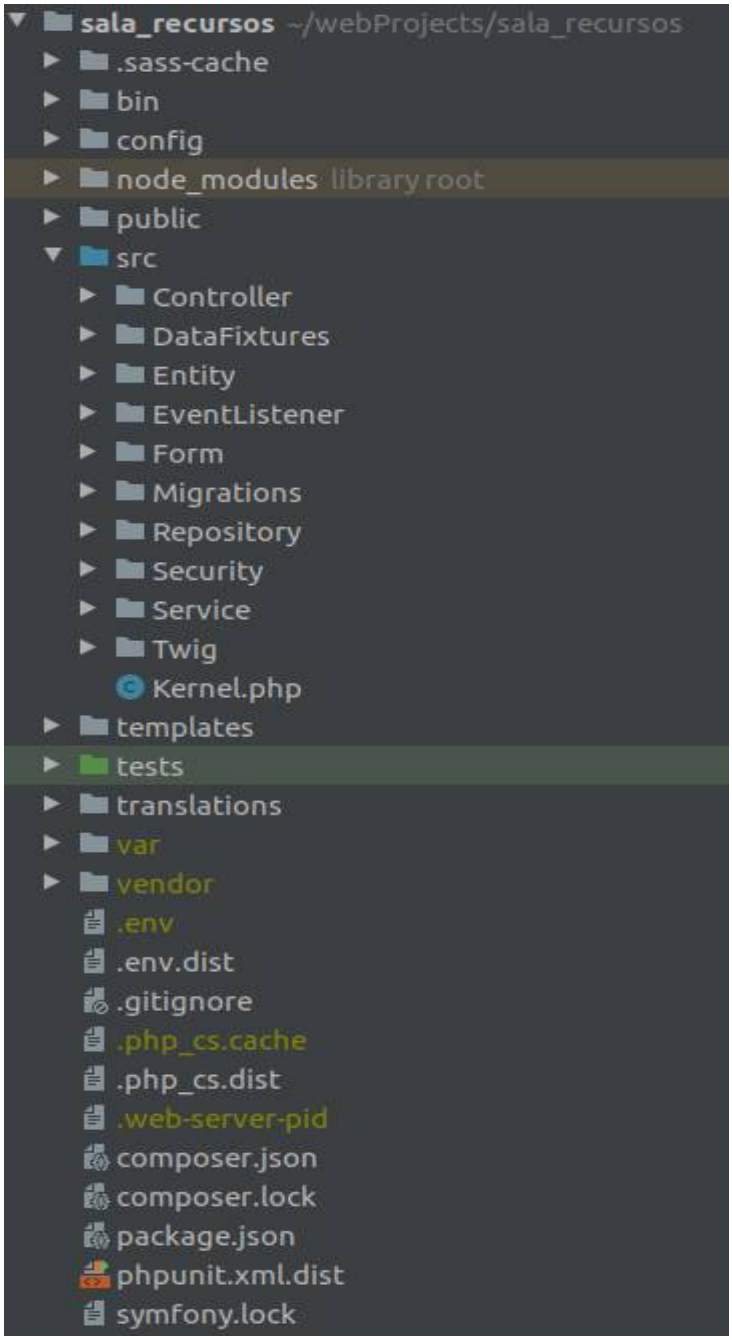
- *escola*
    - Armazena informações básicas de uma escola para o sistema.
- *aluno*
    - Armazena informações básicas de um aluno, assim como o relacionamento da escola atual do aluno.
- *usuario*
    - Armazena informações das pessoas autorizadas a acessar o sistema. Nestas tabelas estão contidos todos os atores do sistema, diferenciados através de sua referência à tabela de perfil.
    - Todo usuário, com exceção do usuário com perfil de Administrador, está relacionado a uma escola, visando restringir o acesso aos registros de sua responsabilidade.
- *perfil*
    - Armazena o nome do perfil de acesso, sendo eles: Administrador, Coordenador e Educador.
- *perfil_controle_acao*
    - Armazena referência a um perfil de acesso.
    - Nome amigável ao usuário da classe *Controller*. Exemplo: Aluno, Escola.
    - Nome amigável ao usuário da ação de uma *Controller*. Exemplo: Salvar, Visualizar.
    - Nome da rota propriamente dita (é um nome técnico que deve representar uma *Route*). Exemplo: aluno_edit, escola_show.
- *comentario*
    - Armazena a descrição e o nome do arquivo de foto/vídeo salvo.
    - Armazena também referência do usuário que gerou assim como referência para qual aluno foi gerado.
- *acompanhamento*
    - Armazena o texto do acompanhamento.
    - Armazena também referência do usuário que gerou assim como referência para qual aluno foi gerado.
- *parecer*
    - Armazena o texto do parecer.
    - Armazena também referência do usuário que gerou assim como referência para qual aluno foi gerado.

- *formulario*
    - Armazena o nome do formulário modelo. Exemplo: Plano de desenvolvimento individual, Adequação curricular.
- *formulario_agrupador*
    - Armazena o nome do agrupador de campos dentro de um formulário modelo. Exemplo: Anacronismos e Habilidades, Plano de Intervenção Pedagógica.
- *formulario_campo*
    - Armazena informações dos campos dos formulários modelos, como legenda e ordem de disposição em tela.
    - Armazena também seus tipos. Exemplo: Texto, Área de texto, Data.
    - Todo campo deve possuir uma referência de um formulario_agrupador.
    - Possui uma referência à entidade_dado_mapeado.
- *entidade_dado_mapeado*
    - Armazena o nome da entidade e o nome do atributo/método que pode ser acessado e utilizado como valor para um registro de formulario_campo.
- *formulario_registro*
    - Armazena os dados de um formulário preenchido, originado com base em um formulário modelo pré-existente.
    - Os registros desta tabela são os formulários dos alunos propriamente ditos.
- *formulario_registro_campo*
    - Armazena os valores preenchidos para cada campo existente em um formulário modelo.
- *migration_versions*
    - Controle interno do *framework* para organização das execuções dos arquivos de migração da base de dados.

4 DESENVOLVIMENTO DO PROJETO

Neste capítulo serão apresentados os principais pontos do desenvolvimento do projeto. Para isso, serão exemplificadas as classes envolvidas em cada uma das camadas do MVC, dentro do contexto do *framework* utilizado. Para fins de referência, abaixo uma imagem da estrutura completa do projeto.

Figura 4.1 – Estrutura completa do projeto



Fonte: Elaborada pelo autor

4.1 Model

As pastas diretamente relacionadas à camada de model são Entity, Repository e Service. Conforme dito anteriormente, um dos pontos fortes do *framework symfony* são seus utilitários por linha de comando. Todas as entidades do projeto foram criadas a partir de comandos, o que garante uma padronização e diminui as chances de erros.

Visando demonstrar o uso prático, será apresentado um cenário de exemplo no qual será adicionada a entidade Turma ao projeto, fazendo com que uma Escola possa ter N Turmas.

O comando utilizado para criar/editar *entities* é o "*./bin/console make:entity*" tendo um exemplo prático ilustrado pela Figura 4.2.

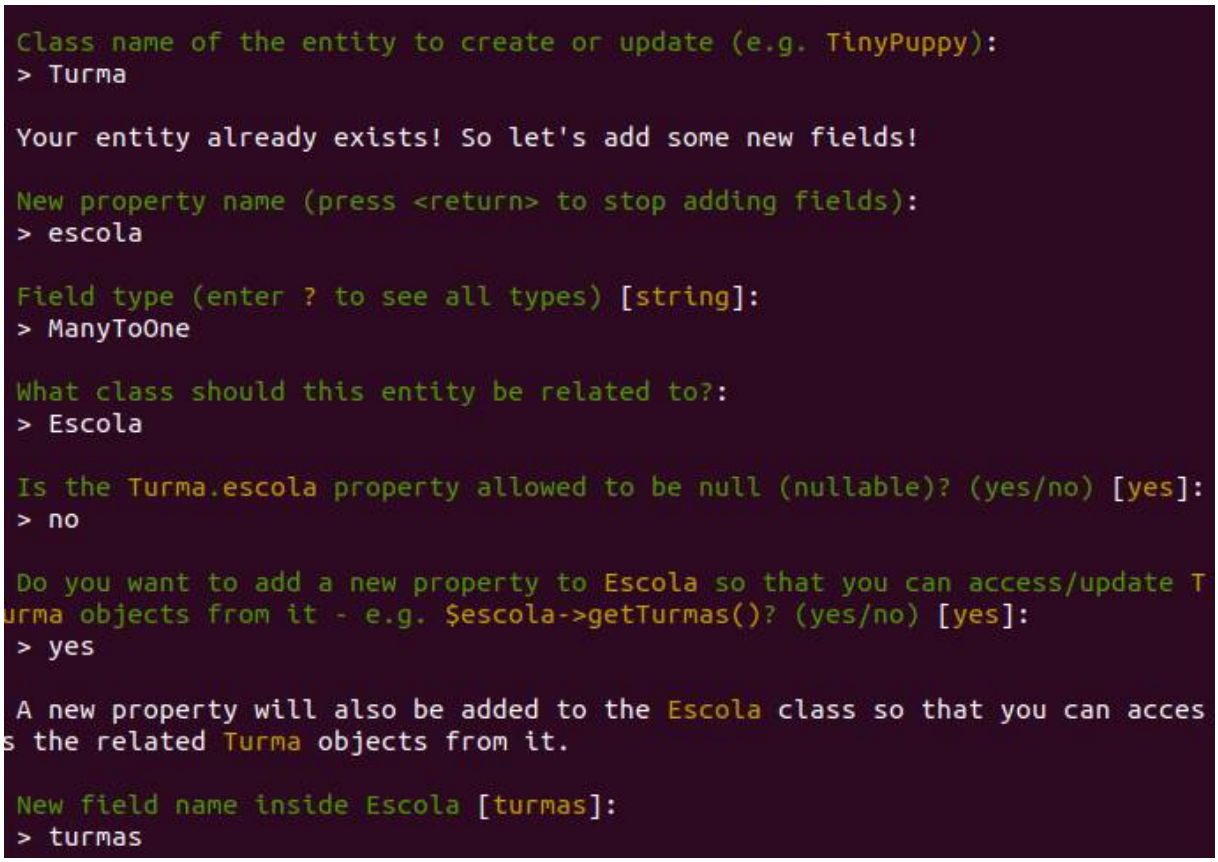
Figura 4.2 Exemplo comando make:entity para criar entidade

```
Class name of the entity to create or update (e.g. VictoriousChef):
> Turma

created: src/Entity/Turma.php
created: src/Repository/TurmaRepository.php

Entity generated! Now let's add some fields!
You can always add more fields later manually or by re-running this command.

New property name (press <return> to stop adding fields):
> nome

Field type (enter ? to see all types) [string]:
> string

Field length [255]:
> 25

Can this field be null in the database (nullable) (yes/no) [no]:
> no

updated: src/Entity/Turma.php
```

Fonte: Elaborada pelo autor

Conforme imagem acima, a primeira coisa a se fazer é definir o nome, assim o *framework* saberá se é uma nova entidade que está sendo criada ou a edição de uma existente. Nesse exemplo, apenas foi definido que a entidade Turma possui um atributo chamado nome.

Figura 4.3 – Exemplo comando make:entity para editar entidade



Fonte: Elaborada pelo autor

Nesta segunda etapa, apresentada na Figura 4.3, foi executado novamente o comando "*./bin/console make:entity*". Nesse momento, ao informar o nome de uma entidade já existente, o *framework* apenas editará o arquivo. Dessa vez foi definido que uma turma possui um atributo chamado escola e que o relacionamento dele é *ManyToOne*, ou seja, N Turmas pertencem a 1 Escola. O framework também realizará, de forma automática, a alteração na entidade Escola, adicionando métodos para acesso às turmas.

O resultado destes comandos é a criação dos arquivos das classes Repository/Turma e Entity/Turma, exemplificado pela Figura 4.4.

Figura 4.4 – Exemplo de classe *Entity* gerada pelo comando make:entity

```php
/**
 * @ORM\Entity(repositoryClass="App\Repository\TurmaRepository")
 */
class Turma
{
    /**
     * @ORM\Id()
     * @ORM\GeneratedValue()
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=25)
     */
    private $nome;

    /**
     * @ORM\ManyToOne(targetEntity="App\Entity\Escola", inversedBy="turmas")
     * @ORM\JoinColumn(nullable=false)
     */
    private $escola;

    public function getId(): ?int
    {
        return $this->id;
    }

    public function getNome(): ?string
    {
        return $this->nome;
    }

    public function setNome(string $nome): self
    {
        $this->nome = $nome;

        return $this;
    }

    public function getEscola(): ?Escola
    {
        return $this->escola;
    }

    public function setEscola(?Escola $escola): self
    {
        $this->escola = $escola;

        return $this;
    }
}
```

Fonte: Elaborada pelo autor

Além dos arquivo gerados, o comando realizará alterações em todos os arquivos existentes que tenham referências à nova entidade. A Figura 4.5 exibe trecho adicionado na class Entity/Escola.

Figura 4.5 – Exemplo de alteração realizada pelo comando make:entity em classe existente

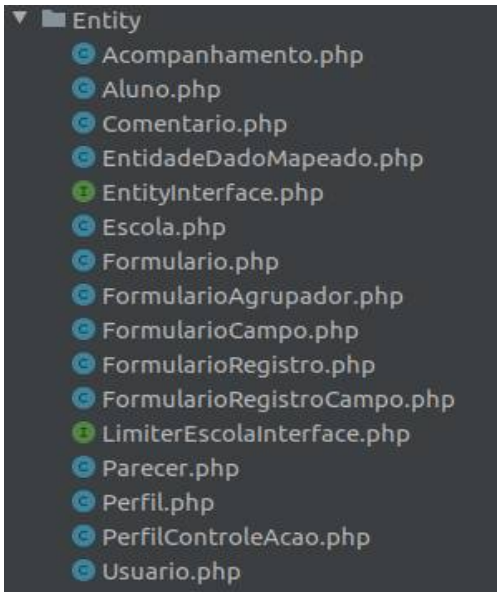
```php
/**
* @return Collection|Turma[]
*/
public function getTurmas(): Collection
{
  return $this->turmas;
}

public function addTurma(Turma $turma): self
{
  if (!$this->turmas->contains($turma)) {
     $this->turmas[] = $turma;
     $turma->setEscola($this);
  }

  return $this;
}

public function removeTurma(Turma $turma): self
{
  if ($this->turmas->contains($turma)) {
     $this->turmas->removeElement($turma);
     // set the owning side to null (unless already changed)
     if ($turma->getEscola() === $this) {
        $turma->setEscola(null);
     }
  }

  return $this;
}
```

Fonte: Elaborada pelo autor

O *framework*, através da ferramenta *Doctrine*, fornece também o comando "*./bin/console doctrine:migrations:diff*" que identificará na pasta de Entity todas essas alterações e gerará um arquivo na pasta Migrations com todos os comandos SQL para evolução da base dados. Esses arquivos podem ser aplicados automaticamente sobre a base de dados através do comando "*./bin/console make:migration*".

Na Figura 4.6 constam todas as entities geradas para o estado atual do projeto.

Figura 4.6 – Lista de todas as *Entity* do projeto

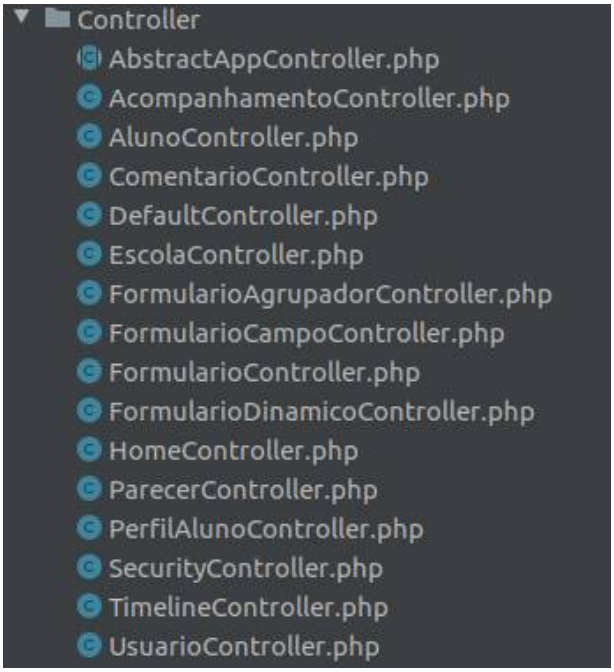


Fonte: Elaborada pelo autor

4.2 Controller

Na camada de *controller* são abordadas as pastas *Controller* e *Form.* As *controllers* do projeto são apresentadas na Figura 4.7.

Figura 4.7 – Lista de todas as *Controller* do projeto



Fonte: Elaborada pelo autor

No decorrer do desenvolvimento, foram feitas melhorias no sentido de diminuir repetições de códigos através de reutilização e compartilhamento de comportamento. Na Figura 4.8 será demonstrado o estado atual através da classe EscolaController no método edit.

Figura 4.8 – Exemplo de método de uma *controller*

```
/**
* @Route("/{id}/edit", name="escola_edit", methods="GET|POST")
* @ParamConverter("entity", class="App\Entity\Escola")
* @IsGranted("escola_edit", subject="entity")
*/
public function edit(Request $request, EntityInterface $entity): Response
{
  return parent::edit($request, $entity);
}
```

Fonte: Elaborada pelo autor

Através da técnica de *annotation*, na primeira linha foi utilizado o recurso de *Route* para definir o nome da rota e os métodos HTTP pelos quais este método está acessível. A segunda linha utiliza o recurso de *ParamConverter*, que faz com que o atributo chamado entity seja instanciado para um class Entity/Escola. Neste exemplo, por se tratar de uma edição, o registro já é carregado do banco de dados e injetado no método. Por último, o recurso *IsGranted* garante a segurança, verificando se o usuário logado possui acesso à rota e se a instância da entidade em questão está acessível ao seu escopo.

O método em si apenas utiliza o método genérico de edição, implementado em sua classe pai, tendo sua existência justificada apenas para definir as configurações explicadas no parágrafo acima. Obviamente essa abordagem funciona apenas para entidades que tenham um comportamento simples e possam ser tratadas de formas genéricas, em outros casos, os métodos são sobrescritos para que seu comportamento possa ser contemplado.

As classes de Form seguem o padrão do *framework*, conforme ilustrado na Figura 4.9.

Figura 4.9 – Exemplo de classe *Form*

```
class EscolaType extends AbstractType
{
  public function buildForm(FormBuilderInterface $builder, array $options)
  {
     $builder
        ->add('nome')
        ->add('endereco', TextType::class, [
          'label' => 'Endereço'
        ]);
  }

  public function configureOptions(OptionsResolver $resolver)
  {
     $resolver->setDefaults([
        'data_class' => Escola::class,
     ]);
  }
}
```

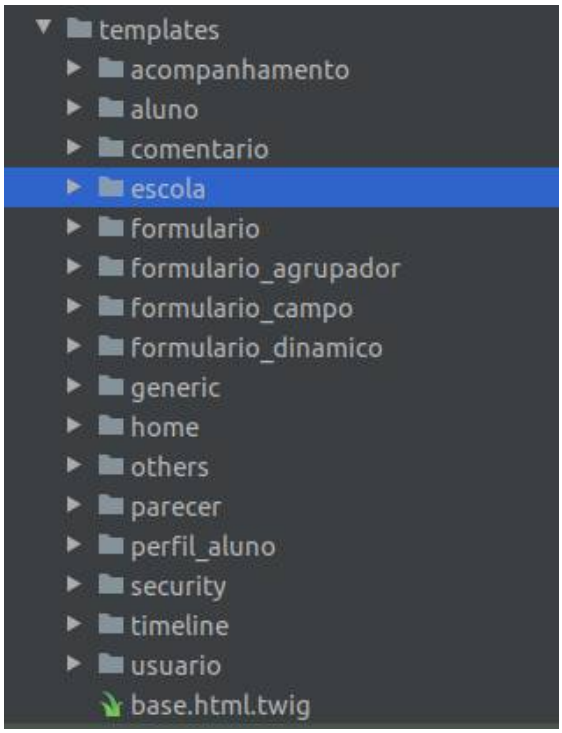
Fonte: Elaborada pelo autor

Seguindo este modelo, somente é necessário definir quais campos devem ser renderizados e processados para este formulário, através do método *buildForm*. O *framework* realiza diversas configurações automaticamente, observando os atributos da entidade em questão e seu formulário. Vale destacar que as principais são: inferir o campo adequado do HTML para o atributo, padronizar a primeira letra do label como maiúscula, definir tamanho máximo para o campo e definir sua obrigatoriedade.

4.3 View

A pasta relacionada à camada de View é a *template*, tendo no projeto, a estrutura apresentada na Figura 4.10.

Figura 4.10 – Lista de todas as pastas de *templates* do projeto



Fonte: Elaborada pelo autor

Ressalta-se que todo conteúdo na pasta é escrito com comandos da ferramenta Twig mesclada com HTML. A figura 4.11 apresenta exemplo do trecho que exibe o formulário para criar/editar uma entidade.

Figura 4.11 – Exemplo de código *Twig* em um arquivo de *view*

```
{{ form_start(form) }}
  {{ form_widget(form) }}
  <button id="submitBtn" class="btn btn-primary">{{ button_label|default('Salvar')
}}</button>

  {% set referer = app.request.server.get('HTTP_REFERER')|default('/') %}

  <a class="btn btn-success float-right" href="{{ referer }}">Voltar</a>
{{ form_end(form) }}
```

Fonte: Elaborada pelo autor

Os comandos entre chaves, por exemplo *form_start*, são comandos Twig que, por sua vez, trabalham em conjunto com as definições do *Symfony*, renderizando o conteúdo definido na classe *Form*, que foi abordada no tópico anterior.

5 FUNCIONALIDADES DO SISTEMA

Neste capítulo serão apresentadas as principais funcionalidades do sistema, sob o ponto de vista do Educador.

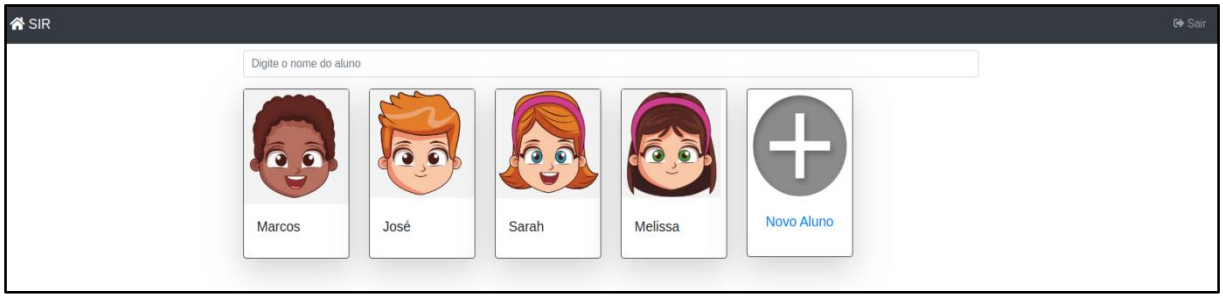
Destaca-se aqui, um recurso que está disponível em todas as telas do sistema: ao clicar no nome do aluno, destacado em verde no canto esquerdo superior, o usuário será encaminhado para a tela de Perfil do aluno, permitindo uma rápida navegação durante o atendimento do aluno.

5.1 Tela inicial

A tela inicial do sistema, ilustrada pela Figura 5.1, apresenta a lista de alunos da escola à qual o usuário está associado e possui as seguintes funcionalidades:

- Ao clicar em uma foto de aluno, o sistema encaminhará para tela de Perfil do Aluno;
- Ao digitar um nome no campo acima da listagem, o sistema aplicará o valor digitado como filtro;
- Ao clicar em Novo Aluno, o sistema encaminhará para tela de Adicionar Aluno.

Figura 5.1 – Tela inicial

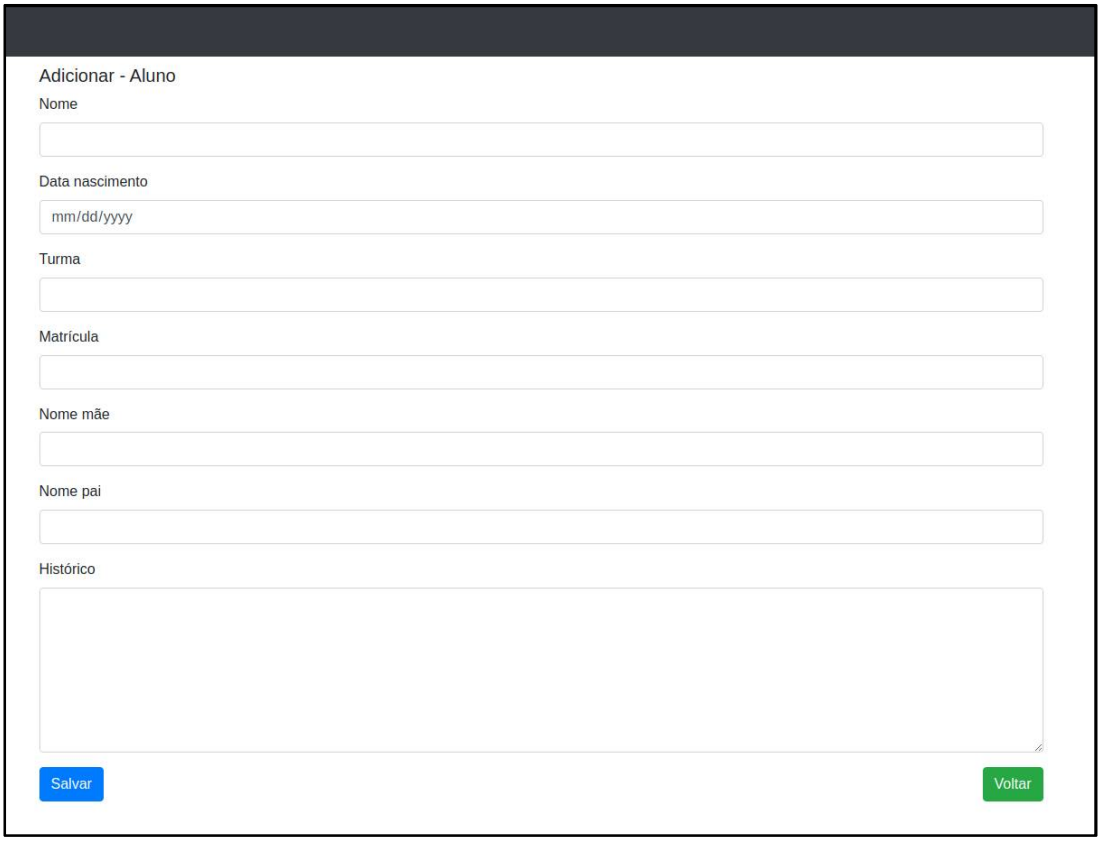


Fonte: Elaborada pelo autor

5.2 Adicionar Aluno

A tela Adicionar Aluno, apresentada na Figura 5.2, permite gerar um novo registro de aluno. Quando um aluno é adicionado, o sistema infere sua escola com base na escola do usuário logado.

Figura 5.2 – Tela adicionar aluno



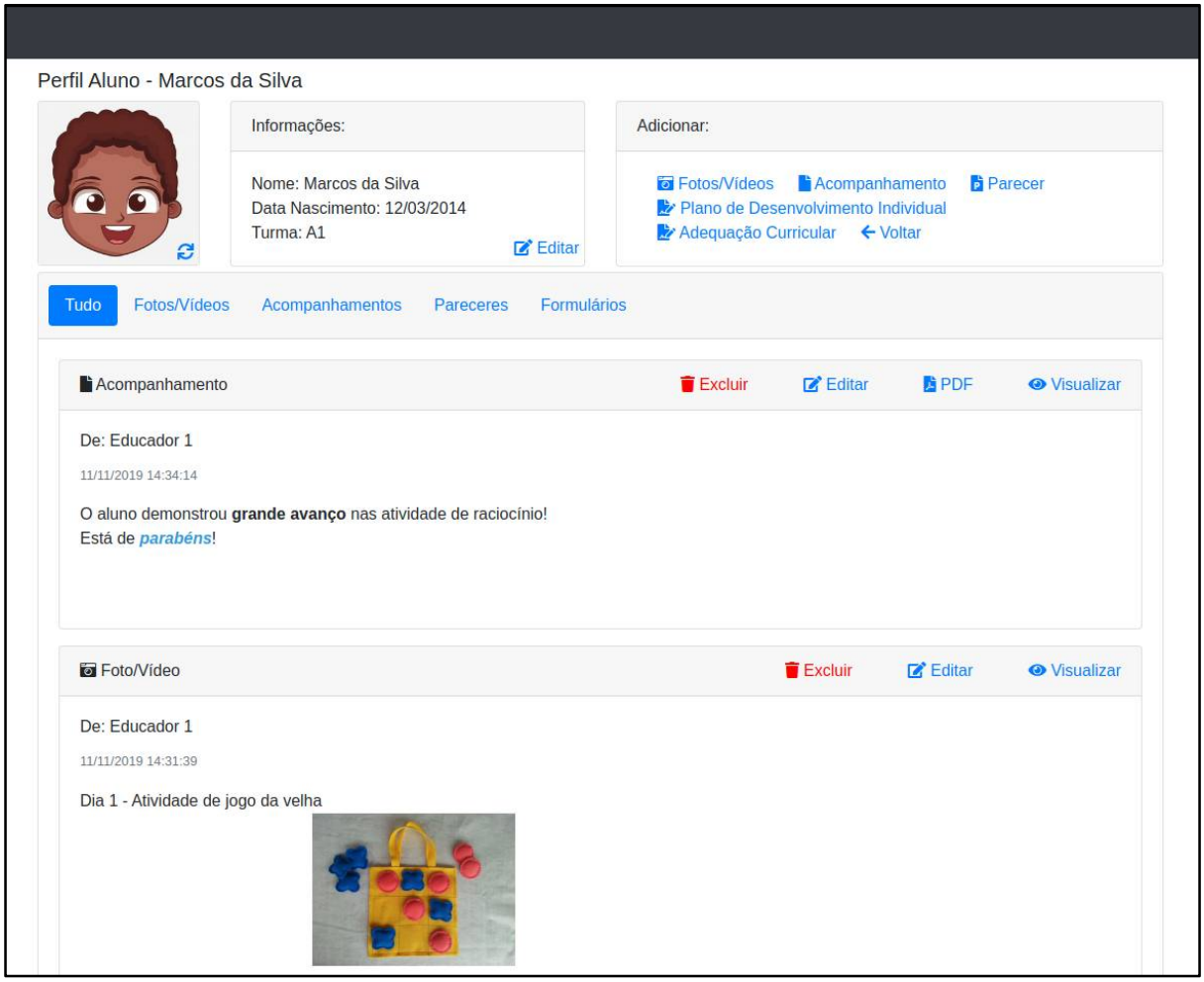
Fonte: Elaborada pelo autor

5.3 Perfil do Aluno

A tela Perfil do Aluno contempla todas as informações já registradas para um aluno, conforme verificado na Figura 5.3, e tem as seguintes funcionalidades:

- Ao clicar na foto do aluno, o sistema encaminhará para tela de Alterar foto do aluno;
- Na parte do centro superior, temos as informações básicas do aluno e acesso a editar o cadastro do mesmo;
- No canto direito superior, na parte de Adicionar, é possível acessar as telas de adicionar: Fotos/Vídeos, Acompanhamentos, Pareceres, Plano de desenvolvimento individual e Adequação curricular;
- Na parte inferior da tela existe um filtro para cada tipo de registro onde, ao clicar, será aplicada a seleção na *timeline* horizontal, que se encontra logo abaixo.

Figura 5.3 – Tela perfil do aluno

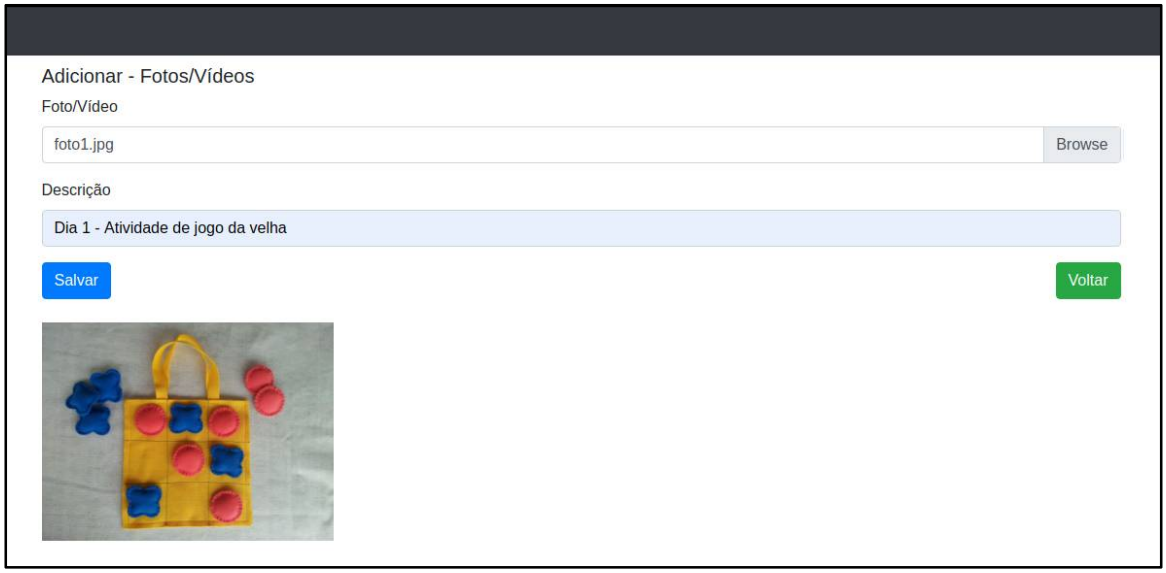


Fonte: Elaborada pelo autor

5.4 Adicionar - Foto/Vídeo

Na tela Adicionar Foto/Vídeo, ilustrada na Figura 5.4, ao fazer a inclusão de alguma dessas mídias, é possível definir uma descrição para o arquivo. Esse arquivo pode ser enviado tanto através de *upload*, quanto obtido diretamente pela câmera do *smartphone*.

Figura 5.4 – Tela Adicionar Foto/Vídeo

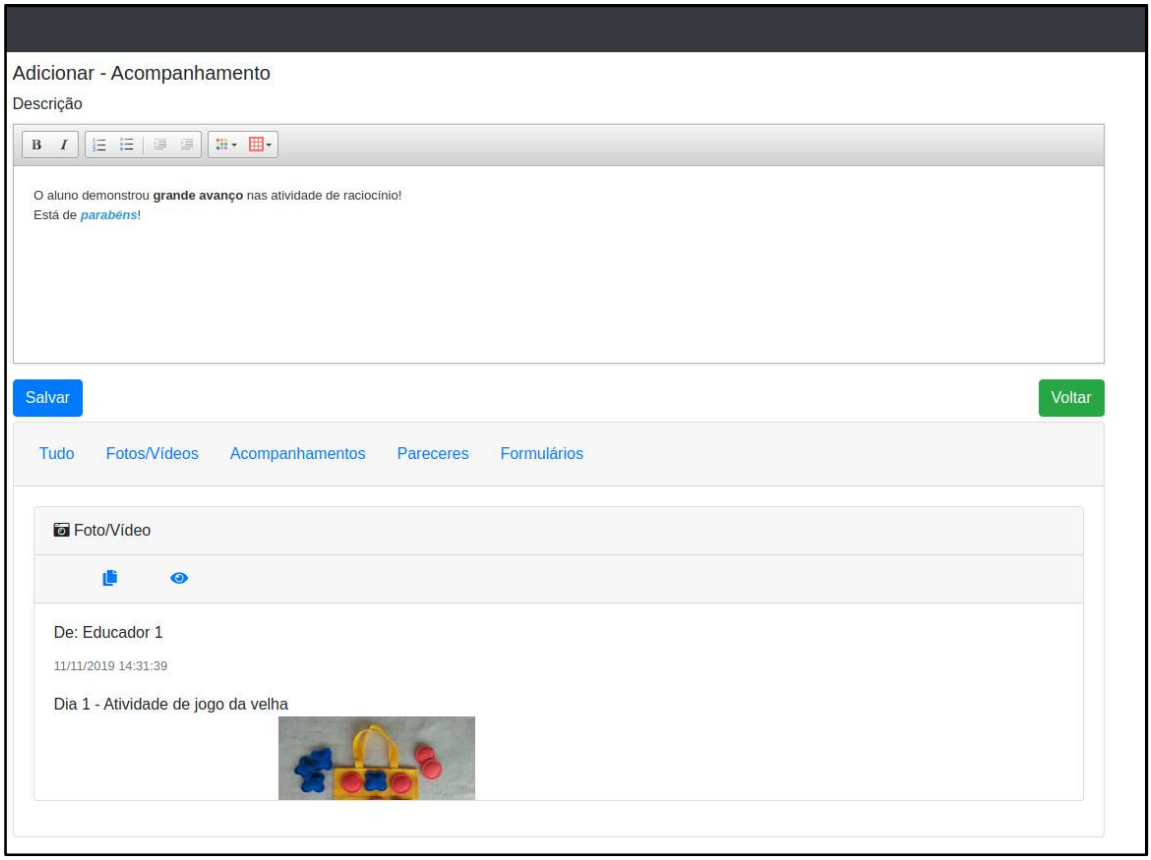


Fonte: Elaborada pelo autor

5.5 Adicionar - Acompanhamento

Na tela Adicionar Acompanhamento, apresentada na Figura 5.5, é possível definir o texto do acompanhamento. Nela foi utilizado um editor que permite algumas formatações básicas de texto. Na parte de baixo da tela há uma timeline horizontal, na qual temos acesso aos registros previamente salvos, os quais podem ter seu conteúdo importado para o acompanhamento através de um clique.

Figura 5.5 – Tela Adicionar Acompanhamento

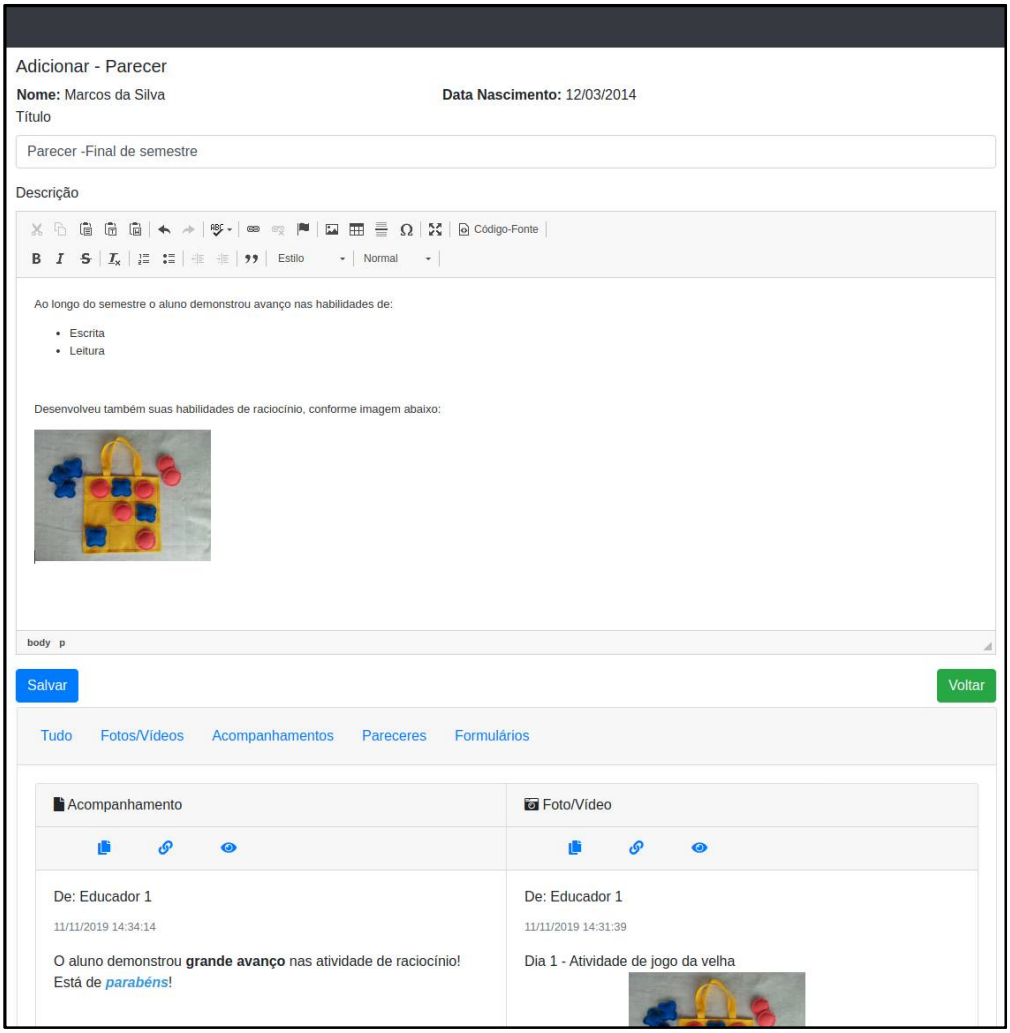


Fonte: Elaborada pelo autor

5.6 Adicionar - Parecer

Na tela Adicionar Parecer, ilustrada pela Figura 5.6, é possível definir o texto de um parecer. Assim como na tela de Adicionar Acompanhamento, foi utilizado um editor texto, porém este permite formatações avançadas e, na parte de baixo da tela, há uma timeline horizontal, na qual temos acesso aos registros previamente salvos, os quais podem ter seu conteúdo importado para o parecer através de um clique.

Figura 5.6 – Tela Adicionar Parecer

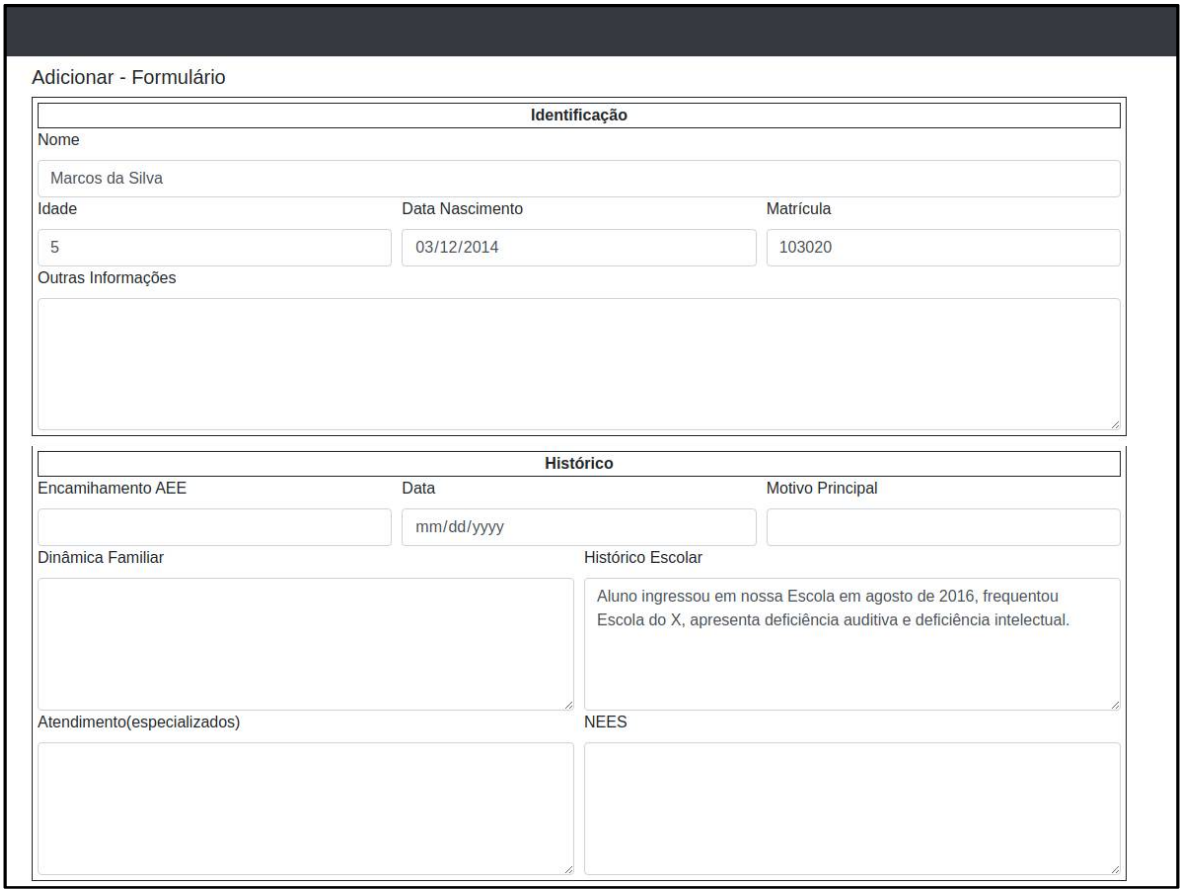


Fonte: Elaborada pelo autor

5.7 Adicionar - Plano de desenvolvimento

A tela Adicionar Plano de Desenvolvimento, exibida na Figura 5.7, permite a criação de um Plano de Desenvolvimento Individualizado para o aluno. Todos os campos da tela foram configurados através do próprio sistema, pelo cadastro de formulários. O layout desse formulário foi desenvolvido para remeter o usuário ao modelo atualmente utilizado no papel. Alguns campos já iniciam com valores obtidos do cadastro do aluno em questão.

Figura 5.7 – Tela Adicionar Plano de Desenvolvimento

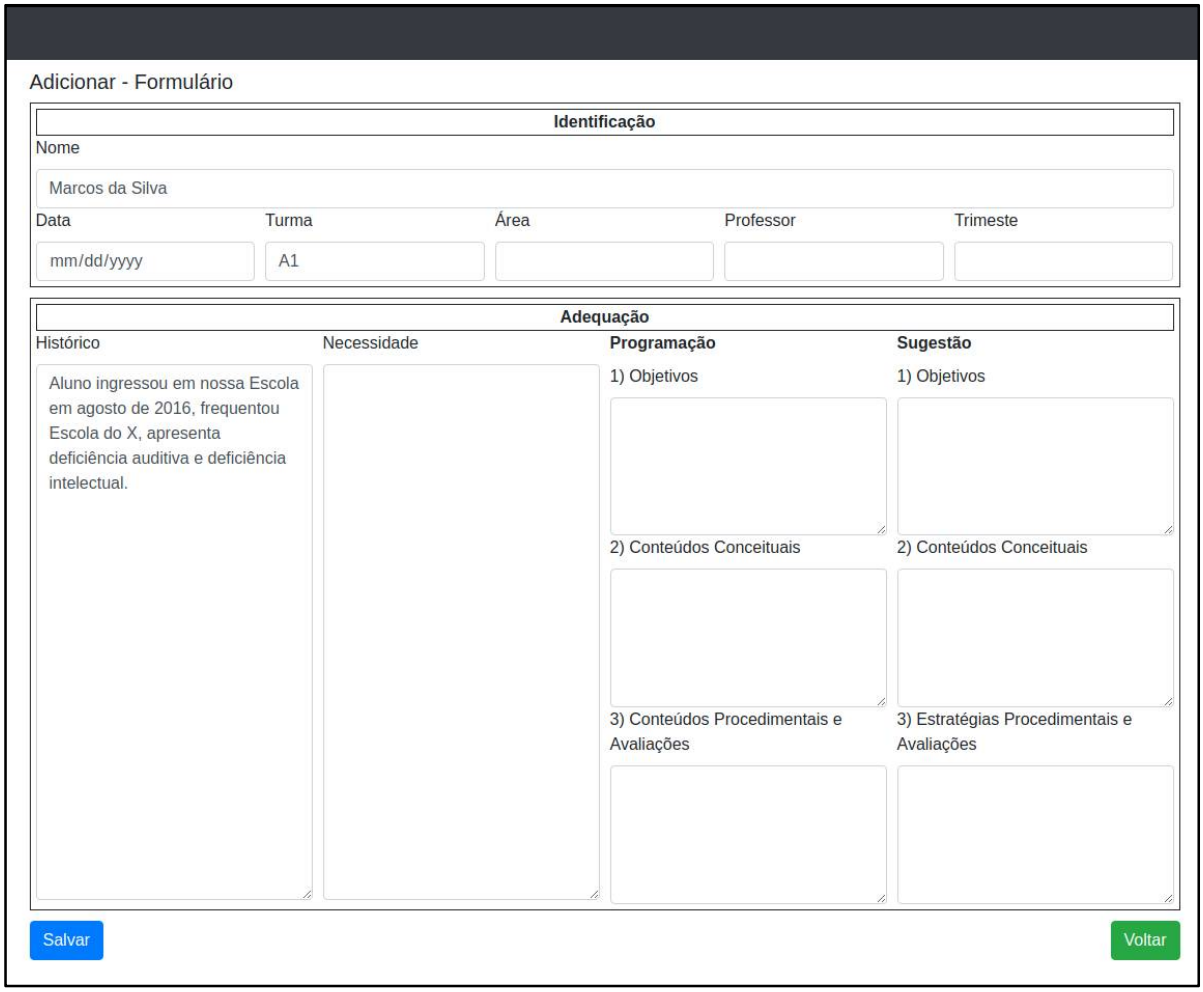


Fonte: Elaborada pelo autor

5.8 Adicionar - Adequação curricular

Nesta tela de Adicionar Adequação Curricular, ilustrada pela Figura 5.8, o educador da SIR registra a proposta de como o professor de sala de aula deve abordar cada conteúdo de acordo com as necessidades do aluno. Assim como na tela Adicionar Plano de Desenvolvimento, todos os campos foram configurados através do próprio sistema, pelo cadastro de formulários, o seu layout foi desenvolvido para remeter o usuário ao modelo atualmente utilizado no papel, e alguns campos já iniciam com valores obtidos do cadastro do aluno.

Figura 5.8 – Tela Adicionar Adequação Curricular



Fonte: Elaborada pelo autor

5.9 Adicionar Formulário Modelo

Diferentemente das telas anteriores, esta não faz parte do escopo de acesso do perfil de Educador. Este acesso é permitido apenas para os perfis de Coordenador e Administrador.

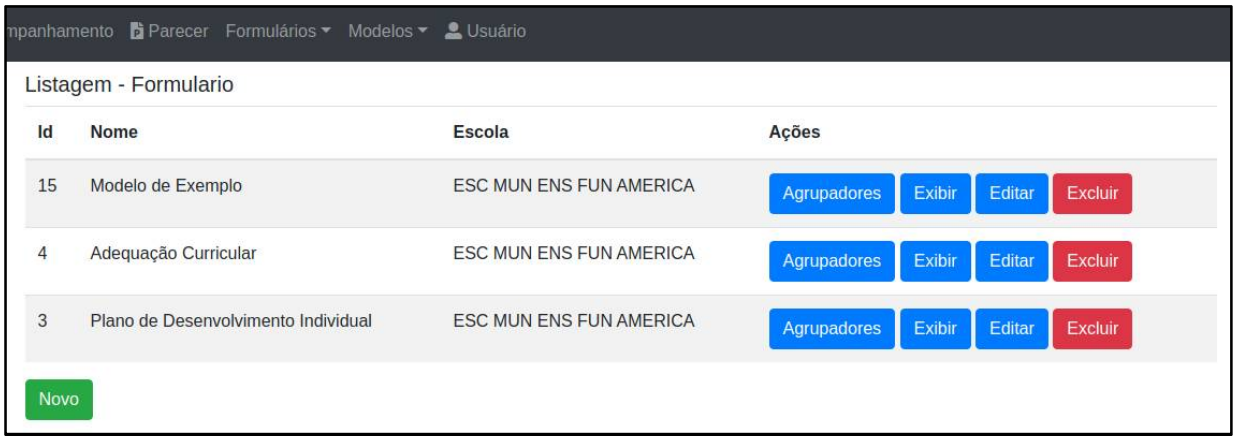
Na criação de um formulário modelo, basta definirmos o nome do mesmo, conforme Figura 5.9.

Depois de criado, na tela de listagem, teremos acesso ao botão de “Agrupadores”, permitindo fazer a gestão do mesmo para esse formulário, conforme Figura 5.10.

Figura 5.9 – Tela Adicionar Formulário Modelo

Fonte: Elaborada pelo autor

Figura 5.10 – Tela Listagem Formulário Modelo



Fonte: Elaborada pelo autor

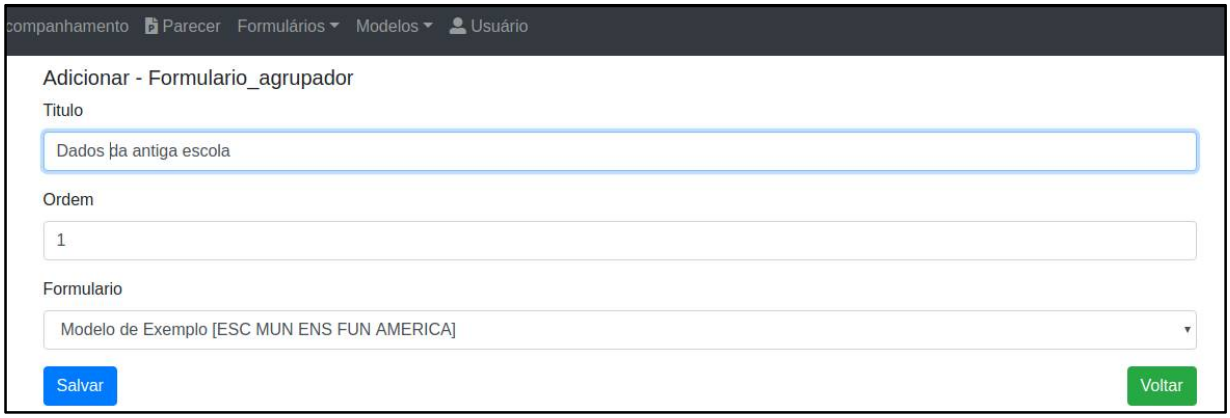
5.10 Adicionar Agrupador

O agrupador permite separar os campos em blocos de assuntos relacionados. No preenchimento do formulário será gerada uma borda ao redor de cada agrupador. Por exemplo, na Figura 5.7 é possível observar os agrupadores de Identificação e Histórico.

Na tela de adicionar agrupador, precisamos definir seu título e ordem que o mesmo será renderizado dentro do formulário, conforme Figura 5.11.

Depois de criado, na tela de listagem, teremos acesso ao botão de “Campos”, permitindo fazer a gestão do mesmo para esse agrupador, conforme Figura 5.12.

Figura 5.11 – Tela Adicionar Agrupador



Fonte: Elaborada pelo autor

Figura 5.12 – Tela Listagem Agrupadores

Fonte: Elaborada pelo autor

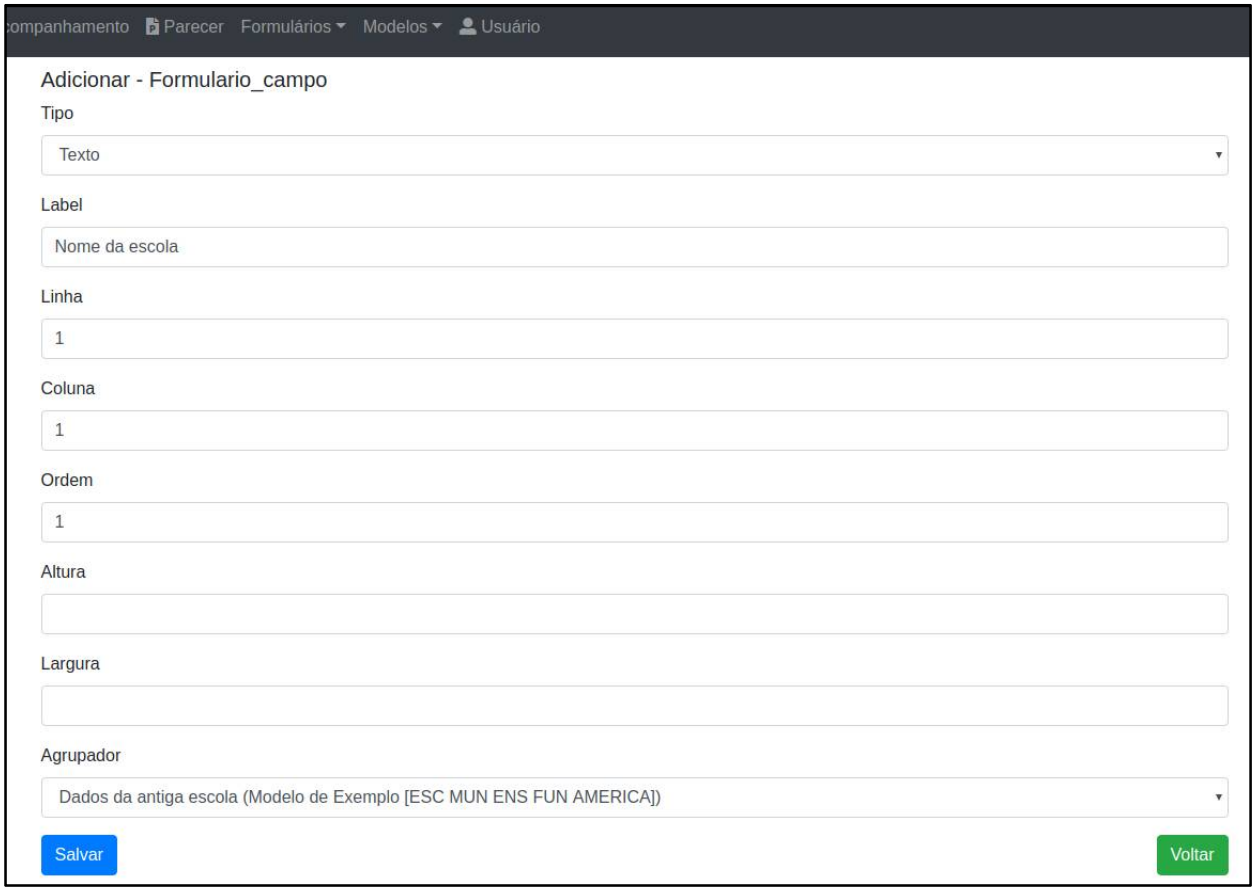
5.11 - Adicionar Campo

O campo representa a informação a ser preenchida pelo usuário. A Figura 5.13 apresenta a tela Adicionar Campo.

O sistema suporta atualmente os seguintes tipos de campo: Texto, Área de Texto, Data e Label. Além disso, todo campo deve possuir a informação do label a ser exibido em tela e as informações de linha e coluna para definir sua posição dentro do agrupador em relação aos demais. As informações de ordem, altura e largura são opcionais e permitem um grau maior de personalização, como pode ser observado na Figura 5.8, onde o campo de Histórico ocupa toda a altura do lado esquerdo.

Na Figura 5.14 é apresentado o resultado de um formulário de exemplo confeccionado nas etapas acima.

Figura 5.13 – Tela Adicionar Campo



Fonte: Elaborada pelo autor

Figura 5.14 – Tela Exemplo do Formulário Criado

Fonte: Elaborada pelo autor

6 AVALIAÇÃO

Para avaliação dos resultados, o sistema desenvolvido foi apresentado para a professora Sibele de Lima Lemos, que atua nas Salas de Integração e Recursos da Rede Municipal de Ensino de Porto Alegre desde 2001. Após a apresentação, a professora Sibele aceitou o convite para utilizar o sistema de forma experimental e, ao final dos testes, preencheu o formulário de avaliação que será apresentado.

Para efeitos da avaliação foi utilizado o método SUS, que será apresentado a seguir.

6.1 Utilização do Sistema

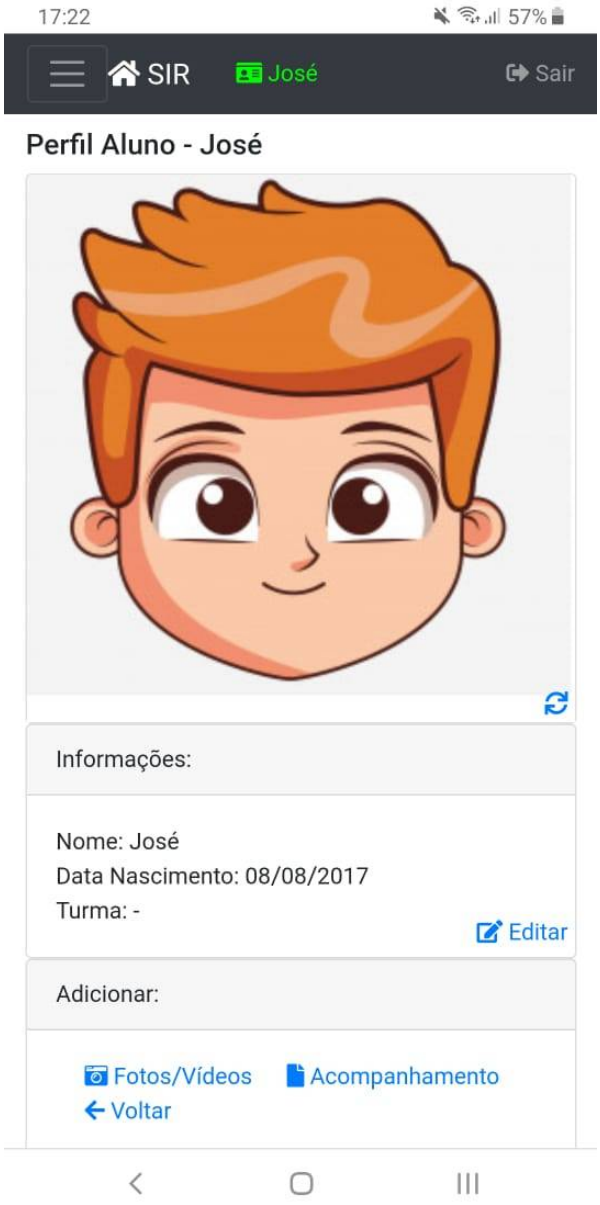
A seguir serão apresentados alguns dados sobre a utilização do sistema pela Professora Sibele. Todas as tarefas foram realizadas sem supervisão técnica, durante seus atendimentos periódicos na SIR:

- Foram cadastrados 3 alunos, todos com suas informações e fotos devidamente registrados;
- Entre fotos/vídeos foram gerados 30 registros;
- Registrados 6 acompanhamentos;
- Registrados 3 pareceres;
- Registrados 2 Planos de desenvolvimento individual.

Nenhum registro de Adequação Curricular foi efetuado, além dos testes durante o desenvolvimento.

Cabe salientar que a professora Sibele acessou o sistema e relizou as tarefas tanto pelo computador, quanto pelo smartphone, reforçando o objetivo de fornecer uma solução responsiva. Na Figura 6.1 é possível observar a tela "Perfil do Aluno" do ponto de vista de quem acessa o sistema pelo smartphone.

Figura 6.1 - Perfil do Aluno visualizado a partir de um smartphone



Fonte: Elaborada pelo autor

6.2 Dúvidas e dificuldades decorrentes da utilização

Durante o período de uso, a comunicação foi mantida através de ferramentas de troca de mensagens. Dessa forma, foi possível mapear as seguintes dúvidas/dificuldades durante a utilização:

- A ação de “colar” precisa ser realizada com atalho “CTRL+V”, não sendo possível através do botão direito do mouse.

- No registro de fotos/vídeos, foi necessária orientação para clicar no ícone de “Filmadora” do smartphone, quando fosse registrar um vídeo.
- Algumas fotos foram registradas em duplicidade, pois o sistema, inicialmente, não tinha indicação de que o registro estava sendo salvo.
- Houve erro ao salvar registro de Plano de desenvolvimento inicial, pois alguns campos tinham conteúdo maior que o inicialmente imaginado.
- Solicitado que o sistema salve automaticamente os registros, pelo menos os mais longos, pois ocorreu de a internet falhar ao salvar, gerando retrabalho.

6.3 System Usability Scale

O método SUS (System Usability Scale ou Escala de Usabilidade do Sistema) fornece uma ferramenta confiável para medir a usabilidade (SUS, 2019). Consiste em um questionário de 10 itens com cinco opções de resposta para os entrevistados (as quais variam de “concordo fortemente” a “discordo fortemente”).

6.3.1 Questionário

Após o período de testes pela professora Sibele, o questionário de avaliação foi respondido tendo os resultados apresentados na Tabela 6.1.

Além das perguntas, foi adicionado um campo de “Observações em geral”, tendo resultado conforme abaixo:

“Considero o uso deste sistema muito bom e importante para cotidiano da Sala de Recursos, como facilitador e organizador do registro do trabalho diário com os alunos para visualizarmos a trajetória e condensarmos no final do ano o trabalho realizado com aluno. Logo que foi apresentado e explicado este sistema, foi de fácil uso e acesso aos recursos oferecidos.”

Tabela 6.1 – Resultados da Avaliação

Perguntas	**Respostas**
1) Eu gostaria de usar este sistema frequentemente.	5 (Concordo Fortemente)
2) Achei o sistema desnecessariamente complexo.	1 (Discordo Fortemente)
3) Eu achei o sistema fácil de usar.	5 (Concordo Fortemente)
4) Penso que precisaria do apoio de uma pessoa técnica para poder utilizar este sistema.	1 (Discordo Fortemente)
5) Eu achei que várias funções deste sistema estavam bem integradas.	5 (Concordo Fortemente)
6) Eu achei que havia muita inconsistência neste sistema.	1 (Discordo Fortemente)
7) Eu imagino que a maioria das pessoas aprenderia a usar esse sistema muito rapidamente.	5 (Concordo Fortemente)
8) Achei o sistema muito complicado de usar.	1 (Discordo Fortemente)
9) Eu me senti muito confiante usando o sistema.	5 (Concordo Fortemente)
10) Eu precisaria aprender muitas coisas antes de poder continuar utilizando este sistema.	1 (Discordo Fortemente)

6.3.2 Pontuação

O cálculo para obter a pontuação seguiu as regras abaixo:

- Nas respostas ímpares (1, 3, 5) foi subtraído um do valor.
- Nas respostas pares (2 e 4) foi subtraído cinco, mantendo o valor absoluto.
- Sendo assim, todas respostas ficaram entre o intervalo de 0 a 4, sendo 4 o maior valor possível.
- Todas as respostas são então somadas e o valor da soma é multiplicado por 2,5, gerando um resultado entre 0 a 100.

A interpretação dos resultados indica que sistemas com pontuação menor que 68 estão abaixo da média. A pontuação obtida pelas respostas da professora Sibele foi de 100, sendo o maior valor possível, o que indica um ótimo grau de aceitação.

7 CONCLUSÃO

O projeto teve como objetivo principal o desenvolvimento de uma solução capaz de auxiliar as principais atividades dos professores especializados das Salas de Integração e Recursos da rede municipal de ensino de Porto Alegre. Esse objetivo foi atingido, tendo em vista tanto a avaliação positiva dos envolvidos quanto o fato de estar sendo efetivamente utilizado no momento e com grande possibilidade de ampliação.

O trabalho desenvolvido resultou em uma solução aderente às rotinas já executadas e principalmente adaptável a evoluções no processo, tendo em vista a capacidade de criar novos formulários modelos que, tão logo confeccionados, já podem ser utilizados, sem necessidade do envolvimento de um desenvolvedor.

Assim como em qualquer solução, existe muito espaço para aprimoramento, tanto os de carácter técnico que envolvem evoluções das tecnologias aplicadas e melhorias na arquitetura, quanto do ponto de vista de novas funcionalidades para o sistema.

Dessas possíveis novas funcionalidades, destacam-se duas sugeridas em conjunto pelos Professores Francisco e Sibele:

- Capacidade de selecionar determinados trechos de acompanhamentos e pareceres e utilizá-los na geração de um novo relatório chamado terminalidade específica.
- Permitir o uso do sistema por mais atores, como profissionais da saúde que realizam acompanhamento dos estudantes fora do ambiente escolar.

Ressalta-se também a necessidade de melhorar a usabilidade da parte de criação de formulários modelos que, apesar do resultado final estar de acordo com o esperado, sua confecção poderia ser mais amigável ao usuário. Essa atividade não pode ser realizada no decorrer do trabalho devido à complexidade e priorização das demais.

Visto que o projeto foi desenvolvido com tecnologias modernas e com uma estrutura sólida, essas funcionalidades poderão ser desenvolvidas, assim como eventuais outras, dando continuidade a solução deste ponto em diante.

REFERÊNCIAS

BRASIL. Decreto nº 7.611/2011 (Decreto do Executivo), 2011. Disponível em: <http://legislacao.planalto.gov.br/legisla/legislacao.nsf/Viw_Identificacao/DEC%207.611-2011?OpenDocument>. Acesso em: 28/11/2019.

COHN, M. **User Stories Applied**: For Agile Software Development, Addison-Wesley, 2004.

FERREIRA, G. M. **SIR-EDU** : sistema integrado de recursos educacionais para a gestão do acompanhamento de alunos com necessidades especiais. Monografia (Graduação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2017.

JQuery. **What is jQuery?**. 2019. Disponível em: <https://jquery.com>. Acesso em 21/10/2019.

LUCAS, E. A. **Sistema de Gestão e Acompanhamento Educacional**. Monografia (Graduação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2018.

MOZILLA. **HTML: Linguagem de Marcação de Hipertexto**. 2019. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTML> . Acesso em 20/10/2019.

MOZILLA. **CSS**. 2019. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/CSS>. Acesso em 20/10/2019.

MOZILLA. **JavaScript**. 2019. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em 20/10/2019.

MUNDEL, C. F. **Modelagem do processo de atendimento em salas de recursos para alunos com necessidades educacionais especiais**. Monografia (Graduação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2019.

SANTOS JÚNIOR, F. D. dos. **As Políticas de Educação Especial na Rede Municipal de Ensino de Porto Alegre: 1989-2000**. Dissertação (Mestrado em Educação)- Faculdade de Educação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002.

SASS. **SASS: Documentation**. 2019. Disponível em: <https://sass-lang.com/documentation>. Acesso em 20/10/2019.

JQuery. **What is jQuery?**. 2019. Disponível em: <https://jquery.com>. Acesso em 21/10/2019.

POSTGRESQL. About. 2019. Disponível em: <https://www.postgresql.org/about/>. Acesso em 17/11/2019.

ROCHA, S. **Novas perspectivas educacionais: caminhada coletiva e reestruturação curricular nas escolas municipais de Porto Alegre**. Porto Alegre: SMED, 1997.

SUS. **System Usability Scale.** 2019. Disponível em: <https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html>. Acesso em 17/11/2019.

SYMFONY. **Introduction**. 2019. Disponível em: <https://symfony.com/doc/current/create_framework/introduction.html>. Acesso em 21/10/2019.

SYMFONY. Template. 2019. Disponível em: <https://symfony.com/doc/current/templates.html>. Acesso em 17/11/2019.

SYMFONY. Controller. 2019. Disponível em: <https://symfony.com/doc/current/controller.html>. Acesso em 17/11/2019.

SYMFONY. Route. 2019. Disponível em: <https://symfony.com/doc/current/routing.html>. Acesso em 17/11/2019.