

UNIVERSIDADE DE MOGI DAS CRUZES

LEONARDO NOGUEIRA GASPAR

PROJETO LÓGICA DE PROGRAMAÇÃO:

Sistema de gestão escolar

Mogi das Cruzes, SP

2024

UNIVERSIDADE DE MOGI DAS CRUZES

LEONARDO NOGUEIRA GASPAR

PROJETO LÓGICA DE PROGRAMAÇÃO:

Sistema de gestão escolar

Projeto apresentado ao curso de análise e desenvolvimento de sistemas da Universidade de Mogi das Cruzes como parte dos requisitos para obtenção de menção na média do segundo bimestre da disciplina de lógica de programação.

Mogi das Cruzes, SP

2024

RESUMO

Neste documento irá ser abordado o desenvolvimento de um sistema de gestão escolar de uma pequena escola. Para estruturar o projeto, foi utilizado HTML e CSS em conjunto com Javascript e PHP, conectado a um banco de dados SQL. O projeto visa criar páginas simples que permitam o gerenciamento da vida escolar, como turmas, notas, professores e alunos.

Palavras-chave: Gerenciamento, escolar, HTML, CSS, Javascript, PHP, SQL.

LISTA DE ILUSTRAÇÕES

Figura 1 -	Cronograma.....	7
Figura 2 -	Página de login.....	8
Figura 3 -	Conexão com a base de dados.....	8
Figura 4 -	Controlador da página de login.....	9
Figura 5 -	Modelo da página de login.....	10
Figura 6 -	Página de escolha de salas.....	11
Figura 7 -	Formulário da página.....	11
Figura 8 -	Controlador da página de escolha.....	12
Figura 9 -	Página principal.....	12
Figura 10 -	Restante da página principal.....	13
Figura 11 -	Controlador da página principal.....	14
Figura 12 -	Página com todos os alunos matriculados.....	14
Figura 13 -	Exemplo de um dos modelos utilizados na consulta de informações.....	15
Figura 14 -	Página de matricular aluno.....	16
Figura 15 -	Controlador da página de matricular aluno.....	17
Figura 16 -	Modelo de matricular alunos.....	17
Figura 17 -	View da tabela contendo as notas.....	18
Figura 18 -	Funções do Javascript.....	19
Figura 19 -	Modelo de alteração das notas.....	20
Figura 20 -	Continuação do modelo de alteração das notas.....	21
Figura 21 -	Calendário.....	21

SUMÁRIO

1	INTRODUÇÃO.....	6
1.1	Definição do problema.....	6
2	PLANEJAMENTO.....	7
3	DESENVOLVIMENTO.....	7
4	DISCUSSÃO, CONCLUSÕES E SUGESTÕES.....	22

1 INTRODUÇÃO

Controlar efetivamente um instituição de ensino é vital para assegurar o sucesso acadêmico dos alunos. Uma escola que não organiza os dados dos alunos, professores, turmas, notas e comunicados tende a ter diversos problemas. A operação é comprometida juntamente com a qualidade de ensino.

Para resolver tais questões, esse projeto foi desenvolvido. A ideia é concluir uma aplicação de gestão escolar utilizando PHP integrado com um banco de dados SQL, neste caso, o escolhido foi o MySQL.

Com esta ferramenta, será possível realizar as operações essenciais do dia a dia escolar, como matricular alunos em turmas, adicionar notas, cadastrar professores etc.

1.1 DEFINIÇÃO DO PROBLEMA

O problema de uma gestão escolar ineficiente é que a qualidade de ensino pode ser seriamente comprometida. Um sistema desorganizado irá afetar o funcionamento diário e a qualidade dos serviços oferecidos. Secretários perderiam muito tempo procurando os registros dos alunos, como notas, frequência e histórico. Além disso, a ausência de processos claros pode causar atrasos na troca de informações importantes, como em que turma um aluno pertence, quais professores dão aula em determinada sala.

Com isso definido, as soluções que esta aplicação visa alcançar é:

- Permitir matricular alunos, professores e funcionários da escola;
- Inserir ocorrências, notas e visualizar a média anual para cada aluno;
- Visualizar todos os alunos da instituição;
- Ver cada aluno e professor que leciona na classe selecionada;
- Calendário atualizado que indica o dia em que cada aluno faz aniversário;

Este artigo explorará os aspectos técnicos da implementação, com ênfase na interface do usuário e na escalabilidade da aplicação. Espera-se que, ao final, esta aplicação contribua significativamente para a eficiência operacional e facilite a gestão escolar na instituição.

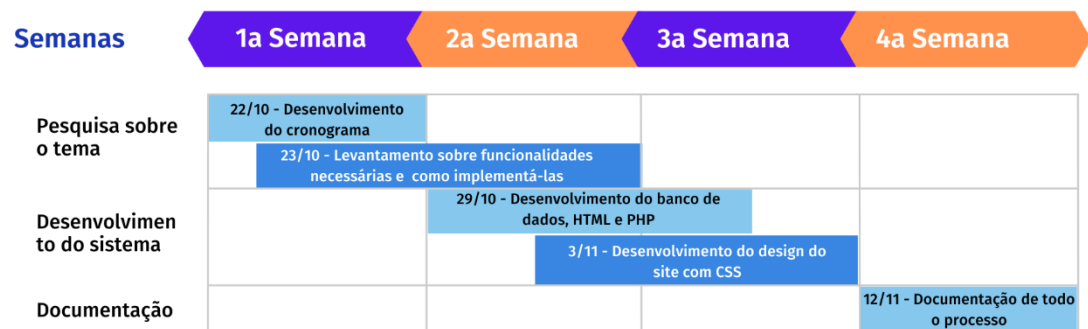
2 PLANEJAMENTO

Antes de qualquer decisão, primeiro é necessário ter uma linha para guiar o rumo do projeto. E para isso, um cronograma eficiente que separe o desenvolvimento em etapas é crucial. Neste projeto, as etapas ficaram separadas da seguinte forma:

Figura 1 – Cronograma

CRONOGRAMA DE DESENVOLVIMENTO

Sistema de gestão escolar

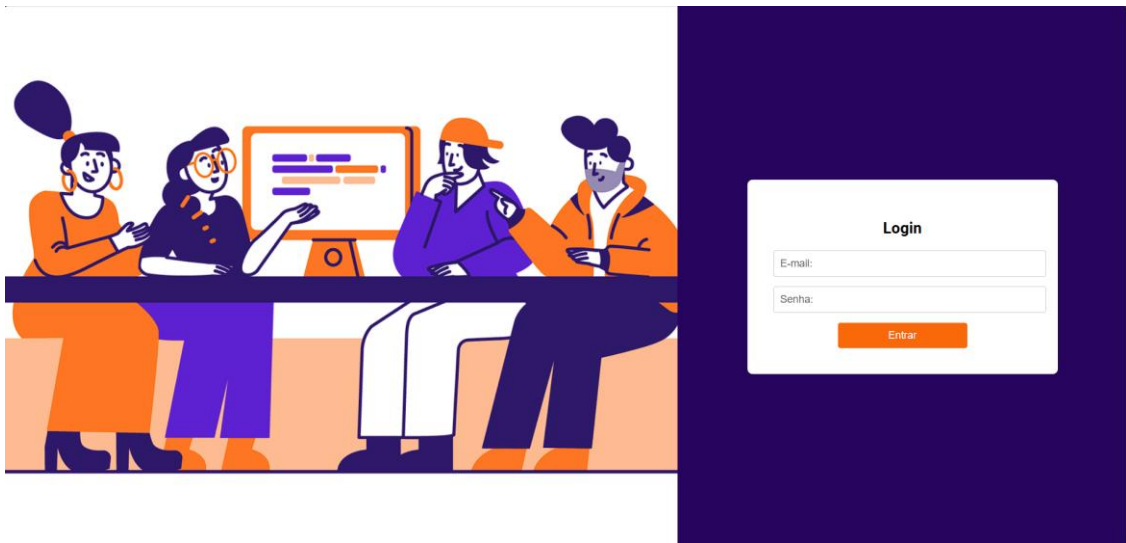


Fonte: Autoral

3 DESENVOLVIMENTO

Para desenvolver o projeto, a primeira página a ser desenvolvida foi a de login, ela teria uma caixa para inserir e-mail e senha e ao pressionar o botão irá verificar no banco de dados se essas informações existem e estão corretas, caso esteja algo errado uma mensagem de erro irá aparecer. Caso esteja correto irá redirecionar à próxima página.

Figura 2 – Página de login



Fonte: Autoral

O projeto foi desenvolvido seguindo o padrão de projeto MVC (Model, View e Controller), permitindo a modularidade e a reutilização de código. Neste caso, ao pressionar o botão de entrar, um formulário com os dados será enviado ao controlador que puxará a função do modelo de verificar o login conforme as imagens:

Figura 3 – Conexão com a base de dados

```
<?php
class Database {
    private $host = '127.0.0.1';
    private $user = 'root';
    private $pass = '';
    private $db = 'escola';
    private $conn;

    public function getConnection() {
        $this->conn = new mysqli($this->host, $this->user, $this->pass, $this->db);
        if ($this->conn->connect_error) {
            die("Erro na conexao: " . $this->conn->connect_error);
        }
        return $this->conn;
    }

    public function closeConnection() {
        if ($this->conn) {
            $this->conn->close();
        }
    }
}
?>
```

Fonte: Autoral

Figura 4 – Controlador da página de login

```
<?php
session_start();
require_once '../models/conta.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $email = $_POST['email'];
    $senha = $_POST['senha'];

    $conta = new Conta();
    $loginValido = $conta->verificarLogin($email, $senha);

    if ($loginValido) {
        $_SESSION['email'] = $email;
        $_SESSION['senha'] = $senha;
        header("Location: salasController.php");
        exit();
    } else {
        $_SESSION['login_error'] = "E-mail ou senha incorretos. Tente novamente.";
        header("Location: ../views/templates/login.php");
        exit();
    }
}
?>
```

Fonte: Autoral

Figura 5 – Modelo da página de login

```
<?php
require_once 'database.php';

class Conta {
    private $conn;

    public function __construct() {
        $db = new Database();
        $this->conn = $db->getConnection();
    }

    public function verificarLogin($email, $senha) {
        $query = "SELECT * FROM funcionario WHERE email = ? AND senha = ?";
        $stmt = $this->conn->prepare($query);
        if ($stmt === false) {
            throw new Exception("Erro ao preparar a query: " . $this->conn->error);
        }

        $stmt->bind_param("ss", $email, $senha);
        if (!$stmt->execute()) {
            throw new Exception("Erro ao executar a query: " . $stmt->error);
        }

        $result = $stmt->get_result();
        if ($result === false) {
            throw new Exception("Erro ao obter o resultado: " . $stmt->error);
        }

        $stmt->close();
        return $result->num_rows > 0;
    }

    public function closeConnection() {
        if ($this->conn) {
            $this->conn->close();
        }
    }

    public function __destruct() {
        $this->closeConnection();
    }
}
?>
```

Fonte: Autoral

Após essa verificação, a página seguinte deve ser a de escolher uma turma para visualizar as informações importantes, para isso, uma página com todas as turmas foi desenvolvida, nela, ao selecionar uma opção, o servidor irá receber uma variável chamada de “id”, dependendo do id, a próxima página exibe determinadas informações.

Figura 6 – Página de escolha de salas

Escolha uma Sala

Sala 1	Sala 2	Sala 3
Sala 4	Sala 5	Sala 6

Fonte: Autoral

Figura 7 – Formulário da página

```
<!DOCTYPE html>
<html lang="pt-BR">
<?php
include 'head_sala.php';
?>
<body>
<h1>Escolha uma Sala</h1>
<form method="post" action="../../controllers/salasController.php">
  <div class="linha">
    <button type="submit" name="id" value="1">Sala 1</button>
    <button type="submit" name="id" value="2">Sala 2</button>
    <button type="submit" name="id" value="3">Sala 3</button>
  </div>
  <div class="linha">
    <button type="submit" name="id" value="4">Sala 4</button>
    <button type="submit" name="id" value="5">Sala 5</button>
    <button type="submit" name="id" value="6">Sala 6</button>
  </div>
</form>
</body>
</html>
```

Fonte: Autoral

Figura 8 – Controlador da página de escolha

```
<?php
require_once 'sessaoController.php';

if (isset($_POST['id'])) {
    $_SESSION['id'] = $_POST['id'];
    header("Location: indexController.php");
    exit();
} else {
    header("Location: ../views/templates/salas.php");
    exit();
}

include '../views/templates/salas.php';
?>
```

Fonte: Autoral

Essa página não possui um modelo pois apenas armazena a variável do id na seção e redireciona para a página principal. Caso tudo esteja certo, esta é a página que o usuário será enviado:

Figura 9 – Página principal

Sala - 1

Alunos **Mensagem**

Informações do Perfil

- Nome: Maria Souza
- Email: secretaria@gmail.com
- Senha: 12345
- [Configurações da Conta](#)

Alunos

- Helio Maria
- Ana Pereira
- Bianca Silva
- Debora Pinheiro
- Claudio Oliveira
- Eduardo Souza
- Fabio Rodrigues
- Giulia Ferreira

Professores

- Rafael Lopes - Portugues
- Jose Cipriano - Mecanica
- Bruno Medina - Sistemas Embarcados
- Vinessa Felix - Banco de Dados
- Daniel Bela - CNC

Aniversários dos Alunos da Sala 1

< > today **novembro de 2024** month week day

dom.	seg.	ter.	qua.	qui.	sex.	sáb.
	27	28	29	30	31	1
						2

Notas dos Alunos


Nome do Aluno	Nota 1º Semestre	Nota 2º Semestre	Média
Helio Maria	0,00	0,00	0,00
Ana Pereira	0,00	0,00	0,00

Fonte: Autoral

Nela será possível acessar todas as informações do sistema. Na barra de navegação estão todas as guias como matricular aluno, adicionar professor, adicionar funcionário, registrar ocorrência, escolher sala ou desconectar do sistema,

além de exibir a foto do usuário e o nome dele. Já na seção principal, é exibido o número da sala, que altera conforme o id que foi enviado, informações do perfil, como nome, e-mail e senha, uma lista com todos os alunos e professores da sala, o calendário com todos os aniversários, uma guia com as notas e médias, permitindo alterar conforme necessário e as ocorrências que cada aluno da sala possui.

Figura 10 – Restante da página principal



Maria Souza

- Site da Escola
- Calendário Escolar
- Regimento Escolar
- Matricular Aluno
- Adicionar Professor
- Adicionar Funcionário
- Registrar Ocorrência
- Visualizar Documentos

Escolher sala

Logout/Deslogar

< > today

novembro de 2024

month week day

dom.	seg.	ter.	qua.	qui.	sex.	sab.
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

Grade Horária da Sala

Material de Estudo

Eventos

Nome do Aluno	Nota 1º Semestre	Nota 2º Semestre	Média
Helio Maria	<input type="text" value="0,00"/>	<input type="text" value="0,00"/>	0,00
Ana Pereira	<input type="text" value="0,00"/>	<input type="text" value="0,00"/>	0,00
Bianca Silva	<input type="text" value="0,00"/>	<input type="text" value="0,00"/>	0,00
Debora Pinheiro	<input type="text" value="0,00"/>	<input type="text" value="0,00"/>	0,00
Claudio Oliveira	<input type="text" value="0,00"/>	<input type="text" value="0,00"/>	0,00
Eduardo Souza	<input type="text" value="0,00"/>	<input type="text" value="0,00"/>	0,00
Fabio Rodrigues	<input type="text" value="0,00"/>	<input type="text" value="0,00"/>	0,00
Giulia Ferreira	<input type="text" value="0,00"/>	<input type="text" value="0,00"/>	0,00

Salvar Notas

Ocorrências

Nome do Aluno	Descrição
Helio Maria	Briga física com o aluno id = 403
Giulia Ferreira	Desrespeito com o professor id = 20

Fonte: Autoral

Contendo a maioria das funções, o controlador dessa página faz a requisição na base de dados de todas as informações necessárias enviando como parâmetro a variável “id”, necessária para consulta.

Figura 11 – Controlador da página principal

```
<?php
require_once 'sessaoController.php';
require_once 'usuarioController.php';

$id = $_SESSION['id'];

require_once '../models/aluno.php';
require_once '../models/nota.php';
require_once '../models/ocorrencia.php';
require_once '../models/professor.php';

$alunoModel = new Alunos();
$notaModel = new Notas();
$ocorrenciaModel = new Ocorrencias();
$professorModel = new Professores();

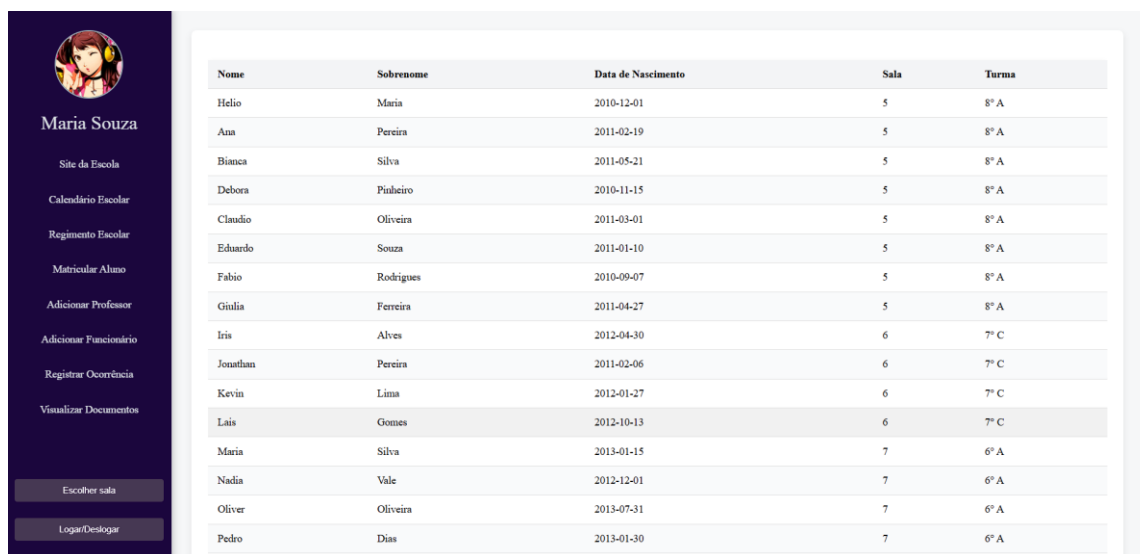
$alunos = $alunoModel->buscarAlunosPorClasse($id);
$aniversarios = $alunoModel->buscarAniversariosPorClasse($id);
$notas = $notaModel->buscarNotasPorClasse($id);
$ocorrencias = $ocorrenciaModel->buscarOcorrenciasPorClasse($id);
$professores = $professorModel->buscarProfessoresPorTurma($id);

include '../views/templates/index.php';
?>
```

Fonte: Autoral

Uma página que pode ser acessada através da principal, é a de visualizar todos os alunos, que será aberta após clicar no botão laranja chamado de “Alunos” e nela irá ser exibido sem as restrições de cada classe, permitindo uma visão ampla e facilitada.

Figura 12 – Página com todos os alunos matriculados



Nome	Sobrenome	Data de Nascimento	Sala	Turma
Helio	Maria	2010-12-01	5	8º A
Ana	Pereira	2011-02-19	5	8º A
Bianca	Silva	2011-05-21	5	8º A
Debora	Pinheiro	2010-11-15	5	8º A
Claudio	Oliveira	2011-03-01	5	8º A
Eduardo	Souza	2011-01-10	5	8º A
Fabio	Rodrigues	2010-09-07	5	8º A
Giulia	Ferreira	2011-04-27	5	8º A
Iris	Alves	2012-04-30	6	7º C
Jonathan	Pereira	2011-02-06	6	7º C
Kevin	Lima	2012-01-27	6	7º C
Lais	Gomes	2012-10-13	6	7º C
Maria	Silva	2013-01-15	7	6º A
Nadia	Vale	2012-12-01	7	6º A
Oliver	Oliveira	2013-07-31	7	6º A
Pedro	Dias	2013-01-30	7	6º A

Fonte: Autoral

Figura 13 – Exemplo de um dos modelos utilizados na consulta de informações

```
<?php
require_once 'database.php';

class Professores {
    private $conn;

    public function __construct() {
        $db = new Database();
        $this->conn = $db->getConnection();
    }

    public function buscarProfessoresPorTurma($id_turma) {
        if (!is_numeric($id_turma)) {
            throw new Exception("ID da turma invalido");
        }

        $query = "SELECT p.nome_prof, p.sobrenome_prof, p.materia
        FROM aulas_turma_$id_turma a
        JOIN professor p ON a.id_prof_fk = p.id_prof";
        $result = $this->conn->query($query);
        if ($result === false) {
            throw new Exception("Erro na consulta de professores: " . $this->conn->error);
        }

        $professores = [];
        while ($professor = $result->fetch_assoc()) {
            $professores[] = [
                'nome_prof' => $professor['nome_prof'],
                'sobrenome_prof' => $professor['sobrenome_prof'],
                'materia' => $professor['materia']
            ];
        }

        return $professores;
    }

    public function closeConnection() {
        if ($this->conn) {
            $this->conn->close();
        }
    }

    public function __destruct() {
        $this->closeConnection();
    }
}
?>
```

Fonte: Autoral

Com toda essa base de consulta, restou criar as páginas de alterar e criar dados. Todas as novas páginas irão funcionar de forma semelhante. Acessadas pela barra de navegação, e com caixas de texto que ao pressionar o botão, irão inserir os dados na tabela correspondente do banco de dados.

Figura 14 – Página de matricular aluno

Matricular Aluno

ID do Aluno:

Nome:

Sobrenome:

Data de Nascimento:

ID da Classe:

Matricular aluno

localhost:8080/projeto/controlers/AdAlunoController.php

Fonte: Autoral

Por exemplo, nesta página de matricular aluno, ao preencher o formulário, será enviado pelo método POST os dados do id do aluno, nome, sobrenome, data de nascimento e id da classe que ele será matriculado. Esses dados serão armazenados em variáveis e estas por sua vez irão ser enviadas como parâmetros ao modelo de inserção. Ao final, uma mensagem informando o estado da requisição irá aparecer na tela.

Figura 15 – Controlador da página de matricular aluno

```
<?php
require_once 'sessaoController.php';
require_once 'usuarioController.php';
require_once '../models/add_aluno.php';

$addAlunoModel = new AddAluno();

$success_message = '';
$error_message = '';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $id_aluno = $_POST['id_aluno'];
    $nome_aluno = $_POST['nome_aluno'];
    $sobrenome_aluno = $_POST['sobrenome_aluno'];
    $data_nasc = $_POST['data_nasc'];
    $id_classe_fk = $_POST['id_classe_fk'];

    $addAluno = $addAlunoModel->inserirAluno($id_aluno, $nome_aluno, $sobrenome_aluno, $data_nasc, $id_classe_fk);

    $success_message = $addAlunoModel->getMessage();
    $error_message = $addAlunoModel->getErrorMessage();
}

include '../views/templates/matricular_aluno.php';
?>
```

Fonte: Autoral

Figura 16 – Modelo de matricular alunos

```
<?php
require_once 'database.php';

class AddAluno {
    private $conn;
    private $message = '';
    private $message_error = '';

    public function __construct() {
        $db = new Database();
        $this->conn = $db->getConnection();
    }

    public function inserirAluno($id_aluno, $nome_aluno, $sobrenome_aluno, $data_nasc, $id_classe_fk) {
        $sql = "INSERT INTO alunos (id_aluno, nome_aluno, sobrenome_aluno, data_nasc, id_classe_fk) VALUES (?, ?, ?, ?, ?)";
        $stmt = $this->conn->prepare($sql);
        if ($stmt === false) {
            throw new Exception("Erro ao preparar a query: " . $this->conn->error);
        }

        $stmt->bind_param("issisi", $id_aluno, $nome_aluno, $sobrenome_aluno, $data_nasc, $id_classe_fk);

        if ($stmt->execute()) {
            $this->message = "Novo registro de aluno inserido com sucesso!";
        } else {
            $this->message_error = "Erro ao inserir aluno: " . $this->conn->error;
        }

        $stmt->close();
    }

    public function getMessage() {
        return $this->message;
    }

    public function getErrorMessage() {
        return $this->message_error;
    }

    public function closeConnection() {
        if ($this->conn) {
            $this->conn->close();
        }
    }

    public function __destruct() {
        $this->closeConnection();
    }
}
```

Fonte: Autoral

Isso se repete para todas as páginas de inserção, menos a de inserir notas aos alunos, pois esta funciona utilizando Javascript, jQuery, AJAX e PHP, para permitir que atualize somente essa parte sem precisar recarregar a página inteira e enviar uma mensagem no navegador utilizando “alert”. O processo é bem simples, ao clicar no botão de salvar as notas, uma função do Javascript será acionada, calculando a média e salvando as notas. Em seguida, esse script irá enviar os dados para o model que se comunica com a base de dados.

Figura 17 – View da tabela contendo as notas

```
<table id="notas-table">
  <h3>Notas dos Alunos</h3>
  <thead>
    <tr>
      <th>Nome do Aluno</th>
      <th>Nota 1º Semestre</th>
      <th>Nota 2º Semestre</th>
      <th>Média</th>
    </tr>
  </thead>
  <tbody>
    <?php foreach($notas as $nota): ?>
      <tr data-id-aluno="<?php echo $nota['id_aluno']; ?>">
        <td><?php echo htmlspecialchars($nota['nome_aluno'] . ' ' . $nota['sobrenome_aluno']); ?></td>
        <td><input type="number" step="0.1" class="nota1" value="<?php echo $nota['nota_primeiro_semestre']; ?>"></td>
        <td><input type="number" step="0.1" class="nota2" value="<?php echo $nota['nota_segundo_semestre']; ?>"></td>
        <td class="media"><?php echo number_format($nota['media'], 2, ',', ''); ?></td>
      </tr>
    <?php endforeach; ?>
  </tbody>
</table>
</div>
<button onclick="salvarNotas()">Salvar Notas</button>
<div class="ocorrencias-table">
```

Fonte: Autoral

Figura 18 – Funções do Javascript

```
function calcularMedia(nota1, nota2) {  
    return ((parseFloat(nota1) + parseFloat(nota2)) / 2).toFixed(2);  
}  
  
function salvarNotas() {  
    const notas = [];  
    const linhas = document.querySelectorAll("#notas-table tbody tr");  
  
    linhas.forEach(linha => {  
        const idAluno = linha.dataset.idAluno;  
        const nota1 = parseFloat(linha.querySelector(".nota1").value) || 0;  
        const nota2 = parseFloat(linha.querySelector(".nota2").value) || 0;  
        const media = calcularMedia(nota1, nota2);  
        linha.querySelector(".media").innerText = media;  
        notas.push({  
            id_aluno: idAluno,  
            nota_primeiro_semestre: nota1,  
            nota_segundo_semestre: nota2,  
            media: media  
        });  
    });  
  
    fetch("/projeto/models/alterarNotas.php", {  
        method: "POST",  
        headers: {  
            "Content-Type": "application/json"  
        },  
        body: JSON.stringify({ notas })  
    })  
  
    .then(response => response.json())  
    .then(data => {  
        if (data.status === "success") {  
            alert("Notas salvas com sucesso!");  
        } else {  
            alert("Erro ao salvar notas.");  
        }  
    })  
  
    .catch(error => console.error("Erro:", error));  
}
```

Fonte: Autoral

Figura 19 – Modelo de alteração das notas

```
<?php
include 'database.php';

class AltNota {
    private $conn;

    public function __construct() {
        $db = new Database();
        $this->conn = $db->getConnection();
    }

    public function salvarNotas($notas) {
        foreach ($notas as $nota) {
            $id_aluno = $nota['id_aluno'];
            $nota1 = $nota['nota_primeiro_semestre'];
            $nota2 = $nota['nota_segundo_semestre'];
            $media = $nota['media'];

            if ($this->validarNota($id_aluno, $nota1, $nota2, $media)) {
                $this->inserirOuAtualizarNota($id_aluno, $nota1, $nota2, $media);
            } else {
                echo json_encode(["status" => "error", "message" => "Dados inválidos para o aluno ID $id_aluno"]);
                exit;
            }
        }
        echo json_encode(["status" => "success", "message" => "Notas salvas com sucesso!"]);
    }

    private function validarNota($id_aluno, $nota1, $nota2, $media) {
        return is_numeric($id_aluno) && is_numeric($nota1) && is_numeric($nota2) && is_numeric($media);
    }

    private function inserirOuAtualizarNota($id_aluno, $nota1, $nota2, $media) {
        $alterar = "INSERT INTO notas (id_aluno_fk, nota_primeiro_semestre, nota_segundo_semestre, media)
VALUES (?, ?, ?, ?)
ON DUPLICATE KEY UPDATE
    nota_primeiro_semestre = ?,
    nota_segundo_semestre = ?,
    media = ?";
        $stmt = $this->conn->prepare($alterar);

        if ($stmt) {
            $stmt->bind_param("iddddd", $id_aluno, $nota1, $nota2, $media, $nota1, $nota2, $media);
            if (!$stmt->execute()) {
                echo json_encode(["status" => "error", "message" => "Erro ao salvar as notas para o aluno ID $id_aluno: " . $stmt->error]);
                $stmt->close();
                exit;
            }
        }
    }
}
```

Fonte: Autoral

Figura 20 – Continuação do modelo de alteração das notas

```
    }
    }
    echo json_encode(["status" => "success", "message" => "Notas salvas com sucesso!"]);
}

private function validarNota($id_aluno, $nota1, $nota2, $media) {
    return is_numeric($id_aluno) && is_numeric($nota1) && is_numeric($nota2) && is_numeric($media);
}

private function inserirOuAtualizarNota($id_aluno, $nota1, $nota2, $media) {
    $alterar = "INSERT INTO notas (id_aluno_fk, nota_primeiro_semestre, nota_segundo_semestre, media)
    VALUES (?, ?, ?, ?)
    ON DUPLICATE KEY UPDATE
    nota_primeiro_semestre = ?,
    nota_segundo_semestre = ?,
    media = ?";
    $stmt = $this->conn->prepare($alterar);

    if ($stmt) {
        $stmt->bind_param("iddddd", $id_aluno, $nota1, $nota2, $media, $nota1, $nota2, $media);
        if (!$stmt->execute()) {
            echo json_encode(["status" => "error", "message" => "Erro ao salvar as notas para o aluno ID $id_aluno: " . $stmt->error]);
            $stmt->close();
            exit;
        }
        $stmt->close();
    } else {
        echo json_encode(["status" => "error", "message" => "Erro ao preparar a query: " . $this->conn->error]);
        exit;
    }
}

}

$nota = new Altnota();
$data = json_decode(file_get_contents("php://input"), true);
if (!empty($data['notas'])) {
    $nota->salvarNotas($data['notas']);
} else {
    echo json_encode(["status" => "error", "message" => "Nenhuma nota fornecida"]);
}
?>
```

Fonte: Autoral

Por último, o calendário contendo os aniversários foi inserido utilizando o pacote FullCalendar. Ele recebe os dados da base de dados contendo todos os aniversários da turma e adiciona como eventos.

Figura 21 – Calendário

```
<script>
    var aniversarios = <?php echo json_encode($aniversarios); ?>;
    document.addEventListener('DOMContentLoaded', function() {
        var calendarEl = document.getElementById('calendar');
        var calendar = new FullCalendar.Calendar(calendarEl, {
            initialView: 'dayGridMonth',
            locale: 'pt-br',
            events: aniversarios,
            headerToolbar: {
                left: 'prev,next today',
                center: 'title',
                right: 'dayGridMonth,timeGridWeek,timeGridDay'
            }
        });
        calendar.render();
    });
</script>
```

Fonte: Autoral

4 DISCUSSÃO, CONCLUSÕES E SUGESTÕES

Finalizando, o projeto conseguiu atingir os objetivos iniciais de maneira exemplar. Não só facilitando na organização da instituição, como também trouxe uma melhoria significativa em diversos aspectos administrativos e pedagógicos. A separação de alunos por salas, notas e outros critérios permitiu um gerenciamento mais eficiente e personalizado, resultando em um ambiente de aprendizado mais estruturado e produtivo.

Contudo, alguns pontos merecem uma atenção futura. Cabe conectar o site com o resto da escola, como por exemplo: desenvolver um sistema de comunicação entre os usuários que poderia ser desenvolvido com PHP, AJAX e um banco de dados. Tal atualização facilitaria extremamente a comunicação entre setores. Outra melhoria seria a opção de personalizar o perfil, poder alterar foto, senha ou cargo. E por fim, conectar os botões que já existem mas não funcionam, como o de direcionar ao site escolar ou exibir um documento com o regimento escolar.