TECH CHALLENGE

**Cadastro do membro do time**

<!-- Firebase App (necessário para todos os recursos do Firebase) -->

<script src="https://www.gstatic.com/firebasejs/9.x.x/firebase-app.js"></script>

<!-- Firebase Firestore -->

<script src="https://www.gstatic.com/firebasejs/9.x.x/firebase-firestore.js"></script>

// Suas credenciais do Firebase

const firebaseConfig = {

apiKey: "API\_KEY",

authDomain: "PROJECT\_ID.firebaseapp.com",

projectId: "PROJECT\_ID",

storageBucket: "PROJECT\_ID.appspot.com",

messagingSenderId: "SENDER\_ID",

appId: "APP\_ID"

};

// Inicializar Firebase

const app = firebase.initializeApp(firebaseConfig);

const db = firebase.firestore();

<form id="teamMemberForm">

<label for="name">Nome:</label>

<input type="text" id="name" required><br>

<label for="email">Email:</label>

<input type="email" id="email" required><br>

<label for="role">Função:</label>

<input type="text" id="role" required><br>

<label for="code">Código:</label>

<input type="text" id="code" required><br>

<button type="submit">Registrar Membro</button>

</form>

<div id="successMessage"></div>  
  
document.getElementById('teamMemberForm').addEventListener('submit', function(e) {

e.preventDefault();

const name = document.getElementById('name').value;

const email = document.getElementById('email').value;

const role = document.getElementById('role').value;

const code = document.getElementById('code').value;

db.collection("teamMembers").add({

name: name,

email: email,

role: role,

code: code

})

.then(() => {

document.getElementById('successMessage').innerText = "Membro registrado com sucesso!";

// Limpar formulário

document.getElementById('teamMemberForm').reset();

})

.catch((error) => {

console.error("Erro ao adicionar membro: ", error);

});

});  
  
**Verificar dados no Firestore**  
service cloud.firestore {

match /databases/{database}/documents {

match /teamMembers/{document=\*\*} {

allow read, write: if true; // Para desenvolvimento. Mude para regras mais seguras em produção.

}

}

}  
  
**Criar Formulário para Inserir Tarefas**  
<form id="taskForm">

<label for="description">Descrição:</label>

<textarea id="description" required></textarea><br>

<label for="deadline">Prazo:</label>

<input type="date" id="deadline" required><br>

<label for="priority">Prioridade:</label>

<select id="priority" required>

<option value="Baixa">Baixa</option>

<option value="Média">Média</option>

<option value="Alta">Alta</option>

</select><br>

<label for="estimatedTime">Tempo Estimado (em horas):</label>

<input type="number" id="estimatedTime" required><br>

<button type="submit">Criar Tarefa</button>

</form>

<div id="taskMessage"></div>  
  
document.getElementById('taskForm').addEventListener('submit', function(e) {

e.preventDefault();

const description = document.getElementById('description').value;

const deadline = document.getElementById('deadline').value;

const priority = document.getElementById('priority').value;

const estimatedTime = document.getElementById('estimatedTime').value;

db.collection("tasks").add({

description: description,

deadline: deadline,

priority: priority,

estimatedTime: estimatedTime,

createdAt: firebase.firestore.FieldValue.serverTimestamp()

})

.then(() => {

document.getElementById('taskMessage').innerText = "Tarefa criada com sucesso!";

// Limpar formulário

document.getElementById('taskForm').reset();

})

.catch((error) => {

console.error("Erro ao criar tarefa: ", error);

});

});  
  
**Visualização e Gestão das Tarefas**  
  
<div id="taskList"></div>  
  
db.collection("tasks").orderBy("createdAt").onSnapshot((snapshot) => {

let tasksHTML = '';

snapshot.forEach((doc) => {

const task = doc.data();

tasksHTML += `

<div class="task">

<h3>${task.description}</h3>

<p><strong>Prazo:</strong> ${task.deadline}</p>

<p><strong>Prioridade:</strong> ${task.priority}</p>

<p><strong>Tempo estimado:</strong> ${task.estimatedTime} horas</p>

</div>

`;

});

document.getElementById('taskList').innerHTML = tasksHTML;

});

**Atribuição de tarefas**  
  
{

"description": "Descrição da tarefa",

"deadline": "2024-09-30",

"priority": "Alta",

"estimatedTime": 4,

"assignedTo": "id\_do\_membro", // ID do membro a quem a tarefa foi atribuída

"createdAt": "2024-09-14T12:00:00Z"

}

<form id="taskForm">

<label for="description">Descrição:</label>

<textarea id="description" required></textarea><br>

<label for="deadline">Prazo:</label>

<input type="date" id="deadline" required><br>

<label for="priority">Prioridade:</label>

<select id="priority" required>

<option value="Baixa">Baixa</option>

<option value="Média">Média</option>

<option value="Alta">Alta</option>

</select><br>

<label for="estimatedTime">Tempo Estimado (em horas):</label>

<input type="number" id="estimatedTime" required><br>

<label for="assignedTo">Atribuir a Membro:</label>

<select id="assignedTo" required>

<!-- Opções de membros serão geradas dinamicamente -->

</select><br>

<button type="submit">Criar Tarefa</button>

</form>

<div id="taskMessage"></div>

**Preencher membros do time**

// Buscar os membros do time e preencher o dropdown

function loadTeamMembers() {

const select = document.getElementById('assignedTo');

db.collection("teamMembers").get().then((snapshot) => {

snapshot.forEach((doc) => {

const member = doc.data();

const option = document.createElement('option');

option.value = doc.id; // O ID do documento será o valor

option.textContent = member.name; // O nome do membro será o texto

select.appendChild(option);

});

});

}

// Chamar a função ao carregar a página

window.onload = loadTeamMembers;

**Salvar tarefa atribuída ao membro**

document.getElementById('taskForm').addEventListener('submit', function(e) {

e.preventDefault();

const description = document.getElementById('description').value;

const deadline = document.getElementById('deadline').value;

const priority = document.getElementById('priority').value;

const estimatedTime = document.getElementById('estimatedTime').value;

const assignedTo = document.getElementById('assignedTo').value; // ID do membro selecionado

db.collection("tasks").add({

description: description,

deadline: deadline,

priority: priority,

estimatedTime: estimatedTime,

assignedTo: assignedTo,

createdAt: firebase.firestore.FieldValue.serverTimestamp()

})

.then(() => {

document.getElementById('taskMessage').innerText = "Tarefa criada e atribuída com sucesso!";

document.getElementById('taskForm').reset();

})

.catch((error) => {

console.error("Erro ao criar tarefa: ", error);

});

});

**Exibir as tarefas atribuídas ao membro**  
  
function loadTasksForMember(memberId) {

db.collection("tasks").where("assignedTo", "==", memberId).orderBy("createdAt").onSnapshot((snapshot) => {

let tasksHTML = '';

snapshot.forEach((doc) => {

const task = doc.data();

tasksHTML += `

<div class="task">

<h3>${task.description}</h3>

<p><strong>Prazo:</strong> ${task.deadline}</p>

<p><strong>Prioridade:</strong> ${task.priority}</p>

<p><strong>Tempo estimado:</strong> ${task.estimatedTime} horas</p>

</div>

`;

});

document.getElementById('taskList').innerHTML = tasksHTML;

});

}

**5. Relacionar membros da tarefa**   
  
document.getElementById('teamMembersList').addEventListener('click', function(e) {

if (e.target.tagName === 'LI') { // Se um membro for clicado

const memberId = e.target.getAttribute('data-id');

loadTasksForMember(memberId); // Carregar tarefas atribuídas a esse membro

}

});  
  
service cloud.firestore {

match /databases/{database}/documents {

match /tasks/{taskId} {

allow read, write: if request.auth != null && request.auth.uid == resource.data.assignedTo;

}

}

}

**Cálculo de alocação de tempo**

// Função para calcular o tempo total alocado para um membro

function calculateTotalAllocatedTime(memberId) {

db.collection("tasks").where("assignedTo", "==", memberId).get()

.then((querySnapshot) => {

let totalHours = 0;

// Iterar sobre as tarefas e somar o tempo estimado

querySnapshot.forEach((doc) => {

const task = doc.data();

totalHours += parseFloat(task.estimatedTime); // Somar o tempo estimado

});

// Exibir o total de horas alocadas

document.getElementById('totalHours').innerText = `Total de horas alocadas: ${totalHours} horas`;

})

.catch((error) => {

console.error("Erro ao calcular o tempo total: ", error);

});

}

// Chamar a função para calcular o tempo total de um membro específico

// Substituir "memberId" com o ID do membro relevante

calculateTotalAllocatedTime("id\_do\_membro");

<div id="totalHours">Total de horas alocadas: 0 horas</div>

**Exibir total de horas**

// Função para calcular o tempo total de todos os membros

function calculateTotalTimeForAllMembers() {

db.collection("teamMembers").get()

.then((snapshot) => {

snapshot.forEach((doc) => {

const member = doc.data();

const memberId = doc.id;

// Para cada membro, calcular o total de horas alocadas

db.collection("tasks").where("assignedTo", "==", memberId).get()

.then((querySnapshot) => {

let totalHours = 0;

querySnapshot.forEach((taskDoc) => {

const task = taskDoc.data();

totalHours += parseFloat(task.estimatedTime);

});

// Exibir o total de horas para cada membro

document.getElementById('totalTimeForMembers').innerHTML += `

<div>

<strong>${member.name}:</strong> ${totalHours} horas alocadas

</div>

`;

});

});

});

}

// Exibir as horas de todos os membros

calculateTotalTimeForAllMembers();

<div id="totalTimeForMembers"></div>

// Ouvir em tempo real as tarefas atribuídas a um membro e atualizar o total de horas

function listenToTaskChanges(memberId) {

db.collection("tasks").where("assignedTo", "==", memberId).onSnapshot((snapshot) => {

let totalHours = 0;

snapshot.forEach((doc) => {

const task = doc.data();

totalHours += parseFloat(task.estimatedTime);

});

document.getElementById('totalHours').innerText = `Total de horas alocadas: ${totalHours} horas`;

});

}

// Chamar a função para um membro específico

listenToTaskChanges("id\_do\_membro");

// Ouvir em tempo real as tarefas atribuídas a todos os membros

function listenToAllMembersTaskChanges() {

db.collection("teamMembers").get().then((snapshot) => {

snapshot.forEach((doc) => {

const member = doc.data();

const memberId = doc.id;

db.collection("tasks").where("assignedTo", "==", memberId).onSnapshot((taskSnapshot) => {

let totalHours = 0;

taskSnapshot.forEach((taskDoc) => {

const task = taskDoc.data();

totalHours += parseFloat(task.estimatedTime);

});

document.getElementById('totalTimeForMembers').innerHTML += `

<div>

<strong>${member.name}:</strong> ${totalHours} horas alocadas

</div>

`;

});

});

});

}

// Exibir o total de horas para todos os membros

listenToAllMembersTaskChanges();

**Passos para Implementação**

**Estrutura de Dados**

Cada tarefa já possui um campo deadline que indica a data final para conclusão da tarefa. Usaremos esse campo para calcular a proximidade do prazo.

**Definir Quando Enviar as Notificações**

Vamos definir uma lógica para enviar notificações **X dias antes** do prazo final. Um intervalo comum seria 1 a 3 dias antes do prazo.

**Firebase Cloud Functions para Notificações**

Utilizaremos o **Firebase Cloud Functions** para verificar periodicamente quais tarefas estão próximas do prazo e enviar as notificações aos membros do time. Isso será feito através de uma função agendada.

**Configurar Firebase Cloud Functions**

**Instale o Firebase CLI** no seu ambiente de desenvolvimento:

npm install -g firebase-tools

firebase init functions

npm install nodemailer

**Notificação programada**

const functions = require('firebase-functions');

const admin = require('firebase-admin');

const nodemailer = require('nodemailer'); // Se estiver enviando e-mails

admin.initializeApp();

const db = admin.firestore();

// Configuração do serviço de e-mail (opcional)

const transporter = nodemailer.createTransport({

service: 'gmail',

auth: {

user: 'seu-email@gmail.com',

pass: 'sua-senha'

}

});

// Função para enviar e-mail (opcional)

function sendEmail(to, subject, message) {

const mailOptions = {

from: 'seu-email@gmail.com',

to: to,

subject: subject,

text: message

};

return transporter.sendMail(mailOptions);

}

// Função agendada que verifica tarefas e envia notificações

exports.checkTaskDeadlines = functions.pubsub.schedule('every day 00:00').onRun(async (context) => {

const now = new Date();

const threeDaysFromNow = new Date();

threeDaysFromNow.setDate(now.getDate() + 3);

try {

const tasksSnapshot = await db.collection('tasks')

.where('deadline', '<=', threeDaysFromNow.toISOString().split('T')[0])

.get();

tasksSnapshot.forEach(async (doc) => {

const task = doc.data();

const memberId = task.assignedTo;

// Buscar o e-mail do membro responsável pela tarefa

const memberSnapshot = await db.collection('teamMembers').doc(memberId).get();

const member = memberSnapshot.data();

const memberEmail = member.email; // Supondo que você tem o campo email no documento do membro

// Mensagem de notificação

const message = `A tarefa "${task.description}" está se aproximando do prazo final (${task.deadline}). Não se esqueça de concluí-la!`;

// Enviar notificação por e-mail

await sendEmail(memberEmail, 'Aviso de prazo final da tarefa', message);

// Adicionar lógica para notificação por push ou painel aqui

console.log(`Notificação enviada para ${memberEmail}`);

});

} catch (error) {

console.error('Erro ao verificar prazos de tarefas: ', error);

}

return null;

});

const payload = {

notification: {

title: 'Prazo de Tarefa Aproximando-se',

body: `A tarefa "${task.description}" está com o prazo final para ${task.deadline}.`,

icon: '/ic\_launcher.png'

}

};

// Enviar notificação via Firebase Cloud Messaging

await admin.messaging().sendToDevice(member.deviceToken, payload);

**Notificar painel de controle**

db.collection("notifications").add({

userId: memberId,

message: `A tarefa "${task.description}" está se aproximando do prazo final (${task.deadline}).`,

createdAt: admin.firestore.FieldValue.serverTimestamp(),

read: false

});

<div id="notificationsPanel"></div>

db.collection("notifications").where("userId", "==", memberId).where("read", "==", false)

.orderBy("createdAt", "desc").onSnapshot((snapshot) => {

let notificationsHTML = '';

snapshot.forEach((doc) => {

const notification = doc.data();

notificationsHTML += `<div class="notification">${notification.message}</div>`;

});

document.getElementById('notificationsPanel').innerHTML = notificationsHTML;

});