function forPermissions() {

doGet()

getCalendars()

generateReport()

generatePDF()

formatDate()

}

function doGet() {

return HtmlService.createTemplateFromFile('index').evaluate().setXFrameOptionsMode(HtmlService.XFrameOptionsMode.ALLOWALL).setFaviconUrl("https://cdn-icons-png.flaticon.com/512/8327/8327839.png").setTitle("Report Generator - Google Calendar");

}

function getCalendars() {

const calendars = CalendarApp.getAllCalendars();

return calendars.map(calendar => ({ id: calendar.getId(), name: calendar.getName() }));

}

function generateReport(calendarId, startDate, endDate) {

const calendar = CalendarApp.getCalendarById(calendarId);

const start = new Date(startDate + 'T00:00:00Z');

const end = new Date(endDate + 'T23:59:59Z');

const events = calendar.getEvents(start, end);

const userUsage = [];

let totalMeetings = 0;

let totalHours = 0;

events.forEach(event => {

if (!event.isAllDayEvent()) {

const creators = event.getCreators();

if (creators.length > 0) {

const organizer = creators[0];

if (organizer) {

const [username, domain] = organizer.split('@');

const duration = (event.getEndTime() - event.getStartTime()) / (1000 \* 60 \* 60);

const userDomain = userUsage.find(usage => usage.user === username && usage.domain === `@${domain}`);

if (!userDomain) {

userUsage.push({ user: username, domain: `@${domain}`, count: 1, hours: duration });

} else {

userDomain.count += 1;

userDomain.hours += duration;

}

totalMeetings += 1;

totalHours += duration;

}

}

}

});

// Ordenar os dados por domínio e, em seguida, por nome de usuário, desconsiderando o '@'

userUsage.sort((a, b) => {

const domainA = a.domain.replace('@', '');

const domainB = b.domain.replace('@', '');

if (domainA === domainB) {

return a.user.localeCompare(b.user);

}

return domainA.localeCompare(domainB);

});

const spreadsheetId = '1R3WHwXNtyNLVYsLxX0uFyFWyDH2w6-Tw-ktwCZuTFSY';

const ss = SpreadsheetApp.openById(spreadsheetId);

const templateSheet = ss.getSheetByName('Template');

// Define the new sheet name incrementally

let reportIndex = 1;

let newName = `Report ${reportIndex}`;

while (ss.getSheetByName(newName)) {

reportIndex += 1;

newName = `Report ${reportIndex}`;

}

let sheet = templateSheet.copyTo(ss).setName(newName);

// Set the dates and summary info

sheet.getRange('A3').setValue(formatDate(new Date(startDate)));

sheet.getRange('B3').setValue(formatDate(new Date(endDate)));

sheet.getRange('C3').setValue(totalMeetings);

sheet.getRange('D3').setValue(totalHours.toFixed(2));

sheet.getRange('D1').setValue(formatDate(new Date()));

let row = sheet.getLastRow() + 1;

userUsage.forEach(usage => {

sheet.getRange(row, 1).setValue(usage.user);

sheet.getRange(row, 2).setValue(usage.domain);

sheet.getRange(row, 3).setValue(usage.count);

sheet.getRange(row, 4).setValue(usage.hours.toFixed(2));

row++;

});

const pdfUrl = generatePDF(ss.getId(), newName);

return { message: `Report generated successfully in sheet: ${newName}`, pdfUrl: pdfUrl, sheetId: spreadsheetId };

}

function formatDate(date) {

const dd = String(date.getDate()).padStart(2, '0');

const mm = String(date.getMonth() + 1).padStart(2, '0');

const yyyy = date.getFullYear();

return `${dd}/${mm}/${yyyy}`;

}

function generatePDF(spreadsheetId, sheetName) {

const ss = SpreadsheetApp.openById(spreadsheetId);

const sheet = ss.getSheetByName(sheetName);

const lastRow = sheet.getLastRow();

const printOptions = {

size: 7,

fzr: true,

portrait: true,

fitw: true,

gridlines: false,

printtitle: false,

sheetnames: false,

fitw: true,

pagenum: 'CENTER',

attachment: false,

top\_margin: 0.2,

bottom\_margin: 0.2,

left\_margin: 0.2,

right\_margin: 0.2

};

const printRange = `&r1=0&c1=0&r2=${lastRow}&c2=4`; // Adjusted to include up to column E

const queryString = Object.keys(printOptions).map(key => `${key}=${printOptions[key]}`).join('&');

const url = `https://docs.google.com/spreadsheets/d/${spreadsheetId}/export?format=pdf&${queryString}&gid=${sheet.getSheetId()}${printRange}`;

const token = ScriptApp.getOAuthToken();

const response = UrlFetchApp.fetch(url, {

headers: {

Authorization: `Bearer ${token}`

}

});

const blob = response.getBlob().setName(`${sheetName}.pdf`);

const pdfFile = DriveApp.createFile(blob);

return pdfFile.getUrl();

}

function forPermissions() {

doGet()

getCalendars()

generateReport()

generatePDF()

formatDate()

}

function doGet() {

return HtmlService.createTemplateFromFile('index').evaluate().setXFrameOptionsMode(HtmlService.XFrameOptionsMode.ALLOWALL).setFaviconUrl("https://cdn-icons-png.flaticon.com/512/8327/8327839.png").setTitle("Report Generator - Google Calendar");

}

function getCalendars() {

const calendars = CalendarApp.getAllCalendars();

return calendars.map(calendar => ({ id: calendar.getId(), name: calendar.getName() }));

}

function generateReport(calendarId, startDate, endDate) {

const calendar = CalendarApp.getCalendarById(calendarId);

const start = new Date(startDate + 'T00:00:00Z');

const end = new Date(endDate + 'T23:59:59Z');

const events = calendar.getEvents(start, end);

const userUsage = [];

let totalMeetings = 0;

let totalHours = 0;

events.forEach(event => {

if (!event.isAllDayEvent()) {

const creators = event.getCreators();

if (creators.length > 0) {

const organizer = creators[0];

if (organizer) {

const [username, domain] = organizer.split('@');

const duration = (event.getEndTime() - event.getStartTime()) / (1000 \* 60 \* 60);

const userDomain = userUsage.find(usage => usage.user === username && usage.domain === `@${domain}`);

if (!userDomain) {

userUsage.push({ user: username, domain: `@${domain}`, count: 1, hours: duration });

} else {

userDomain.count += 1;

userDomain.hours += duration;

}

totalMeetings += 1;

totalHours += duration;

}

}

}

});

// Ordenar os dados por domínio e, em seguida, por nome de usuário, desconsiderando o '@'

userUsage.sort((a, b) => {

const domainA = a.domain.replace('@', '');

const domainB = b.domain.replace('@', '');

if (domainA === domainB) {

return a.user.localeCompare(b.user);

}

return domainA.localeCompare(domainB);

});

const spreadsheetId = '1R3WHwXNtyNLVYsLxX0uFyFWyDH2w6-Tw-ktwCZuTFSY';

const ss = SpreadsheetApp.openById(spreadsheetId);

const templateSheet = ss.getSheetByName('Template');

// Define the new sheet name incrementally

let reportIndex = 1;

let newName = `Report ${reportIndex}`;

while (ss.getSheetByName(newName)) {

reportIndex += 1;

newName = `Report ${reportIndex}`;

}

let sheet = templateSheet.copyTo(ss).setName(newName);

// Set the dates and summary info

sheet.getRange('A3').setValue(formatDate(new Date(startDate)));

sheet.getRange('B3').setValue(formatDate(new Date(endDate)));

sheet.getRange('C3').setValue(totalMeetings);

sheet.getRange('D3').setValue(totalHours.toFixed(2));

sheet.getRange('D1').setValue(formatDate(new Date()));

let row = sheet.getLastRow() + 1;

userUsage.forEach(usage => {

sheet.getRange(row, 1).setValue(usage.user);

sheet.getRange(row, 2).setValue(usage.domain);

sheet.getRange(row, 3).setValue(usage.count);

sheet.getRange(row, 4).setValue(usage.hours.toFixed(2));

row++;

});

const pdfUrl = generatePDF(ss.getId(), newName);

return { message: `Report generated successfully in sheet: ${newName}`, pdfUrl: pdfUrl, sheetId: spreadsheetId };

}

function formatDate(date) {

const dd = String(date.getDate()).padStart(2, '0');

const mm = String(date.getMonth() + 1).padStart(2, '0');

const yyyy = date.getFullYear();

return `${dd}/${mm}/${yyyy}`;

}

function generatePDF(spreadsheetId, sheetName) {

const ss = SpreadsheetApp.openById(spreadsheetId);

const sheet = ss.getSheetByName(sheetName);

const lastRow = sheet.getLastRow();

const printOptions = {

size: 7,

fzr: true,

portrait: true,

fitw: true,

gridlines: false,

printtitle: false,

sheetnames: false,

fitw: true,

pagenum: 'CENTER',

attachment: false,

top\_margin: 0.2,

bottom\_margin: 0.2,

left\_margin: 0.2,

right\_margin: 0.2

};

const printRange = `&r1=0&c1=0&r2=${lastRow}&c2=4`; // Adjusted to include up to column E

const queryString = Object.keys(printOptions).map(key => `${key}=${printOptions[key]}`).join('&');

const url = `https://docs.google.com/spreadsheets/d/${spreadsheetId}/export?format=pdf&${queryString}&gid=${sheet.getSheetId()}${printRange}`;

const token = ScriptApp.getOAuthToken();

const response = UrlFetchApp.fetch(url, {

headers: {

Authorization: `Bearer ${token}`

}

});

const blob = response.getBlob().setName(`${sheetName}.pdf`);

const pdfFile = DriveApp.createFile(blob);

return pdfFile.getUrl();

}